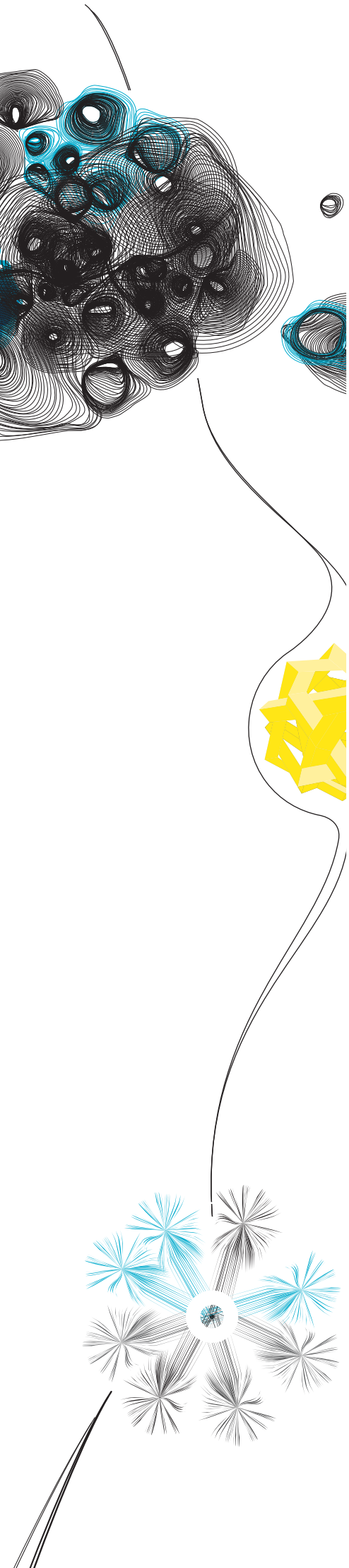

UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering,
Mathematics & Computer Science

MASTER THESIS

Predicting Blood Glucose for Type 2 Diabetes Patients

David de Meij

29th September 2018

Graduation Committee:
dr. ing. G. Englebienne
dr. M. Poel
prof.dr. M.M.R. Vollenbroek-Hutten
N. den Braber, Msc

Human Media Interaction Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede

Acknowledgements

This work would not have been possible without the active participation of type 2 diabetes patients in the cohort study at ZGT. Hopefully this thesis will contribute in alleviating the burden of this disease on these and other patients in the future.

I want to use this opportunity to thank my supervisors, especially Gwenn and Niala, for all their ideas and feedback. Our weekly progress meetings have been a great help. I also want to thank Mannes for pointing me to the computing cluster of the University, as this proved to be essential for experimenting and for providing useful feedback in the last stage of my thesis. Also many thanks to Niala and Milou for interpreting and processing all the manually written food diaries, a very time-consuming and tedious task and to the rest of the Delicate project team for the interesting and informative monthly meetings we had.

Finally, I want to thank my friends and family for all their support, feedback and enthusiasm regarding my Master thesis.

Abstract

Researchers predict 1 out of every 3 adults will get type 2 diabetes. It is important for diabetes patients to keep their blood glucose in a healthy range. However, managing blood glucose is a challenging task because there are many factors that have to be taken into account.

That is why the Delicate project aims to use data on blood glucose, food intake, physical activity and health records, collected in a large cohort study, to provide type 2 diabetics with personalized diabetes and lifestyle coaching. This will be done through an app that will give coaching and also provides blood glucose predictions based on the patient's behaviour, helping them to better manage their disease. In this research we aim to predict future blood glucose levels based on a patient's characteristics and behaviour. We also determine how such as prediction model can be deployed and how the different input features influence the predicted blood glucose.

As a baseline we use an autoregressive model that uses previous blood glucose values to make a prediction. We failed in replicating results of a study aimed at predicting blood glucose of type 1 diabetics. This might be due to some flaws discovered in the study or to the inherent differences between type 2 and type 1 diabetes. However, we were able to significantly ($p<0.1$) outperform our baseline on longer time horizons (>= 60 minutes) using a multitask long short-term memory network (LSTM). The multitask LSTM predicts blood glucose for multiple timesteps into the future at the same time. This not only improves performance (compared to a regular LSTM) but also makes it more convenient to apply in a real world application.

The trained multitask LSTM uses input features such as food intake in a consistent manner, this makes it useful in showing patients how their actions affect their predicted blood glucose.

We recommend visualizing the expected error of the predicted blood glucose in such a way that patients are aware of the limitations of the model, while still benefiting from the insight it provides.

Contents

1. Introduction

Diabetes Mellitus is a chronic condition that affects the body's ability to control the blood glucose level. In the Netherlands 1.2 million people have diabetes (1 out of 14) and researchers predict 1 out of every 3 adults will get type 2 diabetes[1].

It is important for diabetes patients to keep their blood sugar levels within a certain range, as too high blood sugar (hyperglycemia) can lead to serious long-term micro- and macrovascular complications such as kidney failure or blindness [1, 36] and too low blood sugar (hypoglycemia) can lead to blackouts, seizures and even death [2, 38].

In order to keep blood sugar in safe bounds it is important for diabetes patients to be aware of their blood glucose level and how their actions influences this during the day. However, being aware of this is challenging, as there are many factors that have to be taken into account e.g. diet, physical activity and medicine usage. This is especially a problem for type 2 diabetes patients, as they usually get their disease at a later age and are often less educated about how to manage their blood glucose levels [37]. Also most type 2 diabetes patients only measure their blood glucose a few times per day.

The University of Twente (UT) and Ziekenhuis Groep Twente (ZGT) are conducting a cohort study called "Diabetes en Lifestyle Cohort Twente" (DIALECT) with patients suffering from type 2 diabetes. In this study data is being collected about heart rate, physical activity, glucose levels and food intake for a period of two weeks.

The Delicate project {"Diabetes en leefstijl coaching Twente") aims to use this data to provide type 2 diabetes patients with personalized diabetes and lifestyle coaching, through the daily use of an app on their smartphone. This app will provide coaching and also give blood glucose predictions based on the patient's behaviour, helping them to better manage their disease.

We want to predict how the health of type 2 diabetics is influenced by their lifestyle choices. Mainly we are interested in predicting the future blood glucose levels of patients based on previous blood glucose values, patient's characteristics (such as age and gender) and actions that a patient takes (food intake and physical activity).

[1] source: https://www.diabetesfonds.nl/over-diabetes/diabetes-in-het-algemeen/diabetes-in-cijfers (retrieved at 16-4-2018)

1.1. Research goals

The main goal of this research is to create a model that takes historic data such as previous blood glucose values, step count or food intake as input and outputs an accurate blood glucose prediction. We are interested in predicting blood glucose values 30 minutes up to 120 minutes in the future (this is called the prediction horizon). This is expected to be the most useful prediction horizon, because the blood glucose during this time is most affected by actions (such as eating) that a patient takes at the time of the prediction.

A secondary research goal is to determine how the prediction model could best be deployed in a real world application and if it is prefered to train a separate model for each patient, or to use a patient independent model.

Finally, we also aim to learn which input features are important in making an accurate prediction and to find out how a prediction model behaves when we manually change these input features.

2. Background

2.1. Diabetes Mellitus

Diabetes Mellitus is a chronic condition that affects the body's ability to control blood glucose levels.

In a healthy subject the digestive system breaks down carbohydrates from food into glucose. Most of this glucose appears in the bloodstream, thus increasing blood glucose levels. This causes the pancreas to produce insulin, a hormone that signals cells to take in glucose from the bloodstream. When cells take in glucose from the bloodstream they either use it as energy or store it as glycogen which is basically a fuel reserve. This storage happens mostly in the liver and muscle cells.

As the blood glucose level decreases, the pancreas get triggered to produce another hormone called glucagon, which signals cells to degrade the stored glycogen back into glucose which can then be used as energy.[2]

Type 2 diabetes patients either can't use insulin (cells become insulin resistant) and/or their pancreas can't produce (enough) insulin (insulin deficiency), which results in high levels of blood glucose (hyperglycemia).

Hyperglycemia can damage the tiny blood vessels in the organs or the nervous system. In the long-term this can result in serious health issues such as [36, 41]:

- diabetic retinopathy (potentially causing blindness);
- nerve damage (neuropathy);
- kidney damage or kidney failure;
- peripheral artery disease (causing serious foot infections and in some severe cases even requires amputation);
- cardiovascular disease.

A too low blood sugar (hypoglycemia), which is often caused by an overdose of insulin medicine, can lead to blackouts, seizures and even death [38].

[2] Source: https://www.healthline.com/health/diabetes/insulin-and-glucagon (retrieved on 16-9-2018)

There are three types of diabetes[3]:

- **Type 1 diabetes**, also called insulin-dependent diabetes, is an autoimmune condition (that often starts in childhood) that causes the body to attack its own pancreas. The damaged pancreas stops producing insulin.
- **Type 2 diabetes**, also called non-insulin-dependent diabetes, is by far the most common form of diabetes, accounting for 95% of diabetes cases in adults. This is a milder form of diabetes that often starts later in life, although with the epidemic of obese and overweight kids more teenagers are now developing type 2 diabetes. With type 2 diabetes the pancreas usually produces some insulin. But either the amount produced is not enough for the body's needs or the body's cells are resistant to it. Insulin resistance happens primarily in fat, liver and muscle cells and results in the pancreas having to work overly hard to produce enough insulin. People who are obese are at particularly high risk of developing type 2 diabetes.
- **Gestational diabetes** is a form of diabetes that is triggered by pregnancy. In 2% to 10% of pregnancies pregnancy leads to insulin resistance. Because high blood sugar levels in a mother are circulated through the placenta to the baby, it must be controlled to protect the baby's growth and development.

2.2. Diabetes (self-)management

In order to avoid complications due to too high blood sugar as discussed in the previous section it is important for diabetes patients to control their blood sugar levels.

Type 1 diabetes patients manage their blood sugar level by using insulin. They either receive insulin via an insulin pump or by multiple daily injections. [6]

Type 2 diabetes is a slowly progressing disease with several stages that starts with cells becoming insulin resistant, causing the body to produce more insulin to keep blood sugar levels low. However, at some point the pancreas becomes too stressed; insulin production goes down and eventually no insulin is produced at all (see **figure 1**), this process can take more than 10 years. In the early stages of the disease when the patients still produce insulin, they are treated with oral antidiabetic medicine that lower insulin resistance and/or lower glucose in the blood. They are also advised lifestyle changes (mainly focussing on physical activity and diet). In further developed

[3] source: https://www.webmd.com/diabetes/guide/types-of-diabetes-mellitus (retrieved on 27-3-2018)

cases of type 2 diabetes patients may also need to use exogenous insulin to manage their blood sugar levels. [6]

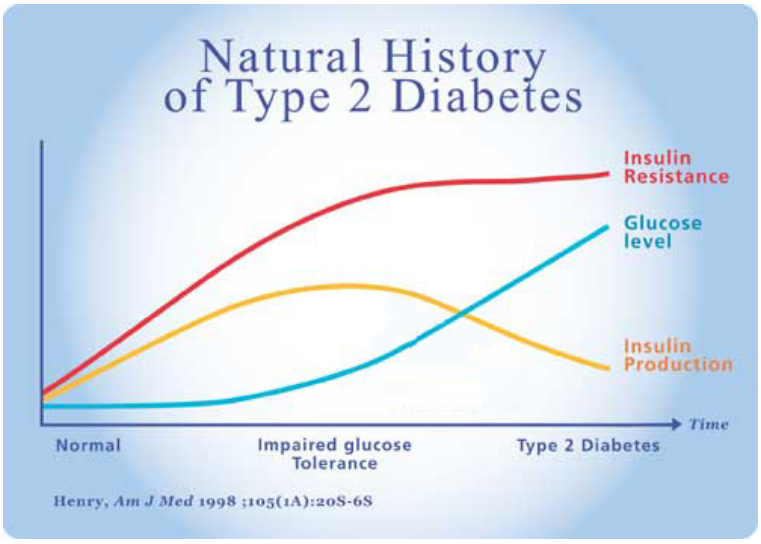

Figure 1. Progression of Type 2 Diabetes Mellitus[4]

Blood glucose level monitoring is an important aspect in diabetes self-management, it is used by diabetes patients and their families to make appropriate day-to-day treatment choices about diet and physical activity, as well as about insulin or other medication [8]. Zecchin et al. [9] showed that by predicting the future glucose levels and alerting patients when the blood sugar will go too low, patients are better able to avoid hypoglycemic events.

However, type 2 diabetes patients are often less educated about their disease and do not necessarily know what to do with their blood sugar monitoring information in order to keep their blood glucose within a healthy range. That is why an application that coaches these patients and accurately predicts and visualizes the effects of the patients' actions on their future blood glucose values can be beneficial. It can give these patients a much better insight in how their current behaviour (mainly food intake, physical activity) influences their future blood glucose levels and thus how they should adapt their behaviour. It is important that this application gives a reasonably accurate prediction, as an incorrect prediction may lead patients to use too much insulin (potentially causing hypoglycemia) or to incorrectly adjust their behaviour (potentially causing hyperglycemia). It might also make patients lose their trust in the application, making it less likely that they will adapt their behaviour accordingly.

[4] Adapted from: http://www.diabetesclinic.ca/en/diab/1basics/insulin_resistance.htm (retrieved on 12-4-2018)

2.3. Computational models

There is a variety of computational models that can be applied to the blood glucose prediction task. In this section we describe the most relevant models.

2.3.1. Autoregressive model

An autoregressive model is a regression model that uses its own past values to make a prediction about the next value. For example we might use the past 3 blood glucose values to predict the blood glucose in 60 minutes, applying a weighted sum in the form of:

$$y_{60} = a * x_{-30} + b * x_{-15} + c * x_0$$

Where y_t is the predicted blood glucose at time t and x_t is a previous blood glucose value at time t. We can easily determine the optimal parameters for a, b and c by minimizing the error between the actual blood glucose value and the predicted value for all available data points.

An important decision in training an autoregressive model is how many previous values are used as input to the model. Generally the last known value has the highest correlation with the value that has to be predicted and this correlation decreases with values further into the past. This means that as more previous values are added as additional input, the improvements in performance become smaller.

2.3.2. Support vector regression

Support vector regression (SVR) is based on support vector machines, which is a binary classification model that works by splitting two classes by a hyperplane (or line) using the largest possible margin between the closest points (the support vectors) and the hyperplane (see **figure 2**).

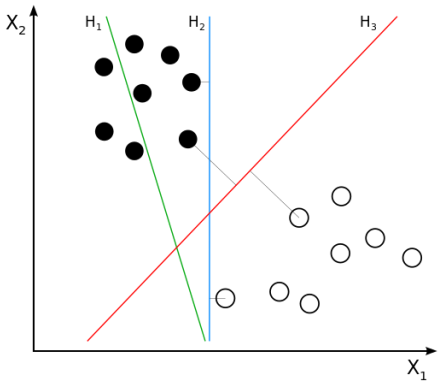

Figure 2. In this image H_2 and H_3 both separate the training examples perfectly, but intuitively H_3 seems like a better division because the margin between the line and the two classes is higher.[5]

In case the data is not directly linearly separable, a kernel function is used that transforms the data in such a way that it becomes linearly separable. For example the data can be transformed into polar coordinates to make it linearly separable (see **figure 3**).

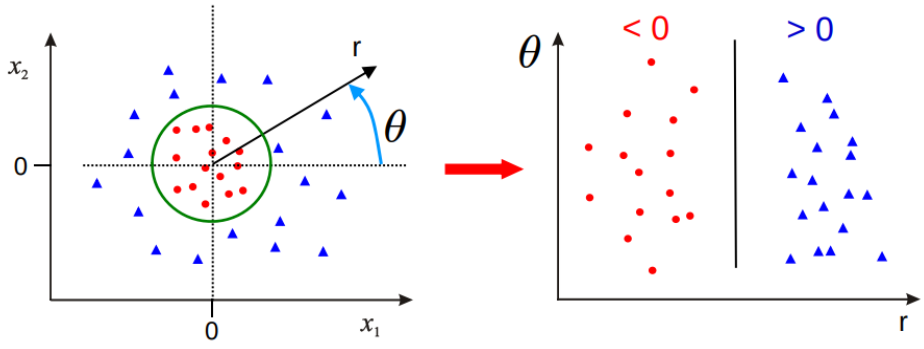

Figure 3. Transforming data from original space to polar coordinates to make it linearly separable.[6]

SVR uses the same principle, but instead of a binary classification it outputs a real number. In this case a kernel function is used to make the data linearly predictable instead of linearly separable (see **figure 4**).

[5] http://nl.wikipedia.org/wiki/Support_vector_machine#/media/File:Svm_separating_hyperplanes_(SVG).svg (retrieved at 19-9-2018)

[6] Source: http://www.robots.ox.ac.uk/~az/lectures/ml/lect3.pdf (retrieved at 17-9-2018)

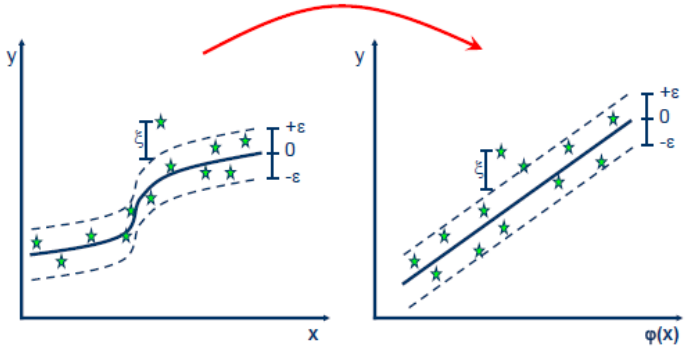

Figure 4. Use a kernel function to make the data linearly predictable.[7]

We then minimize the cost function:

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i$$

Where w refers to the weights of the linear function and ξ refers to the distance between the error margin and points that fall outside of the error margin $[-\epsilon, \epsilon]$ (see **figure 4**). C is a hyperparameter that determines the weight the algorithm will put on minimizing the cost (instead of on minimizing the weights).

2.3.3. Neural network regression

An artificial neural network is a type of computational model that is loosely inspired by how neurons in the biological brains function. A neuron in the brain receives signals from its dendrites and, if a certain threshold is met, it fires a signal across its axon which branches out to the dendrites of multiple other neurons (see **figure 5**). Each neuron can *learn* by changing how much weight it puts on the different inputs from other neurons.

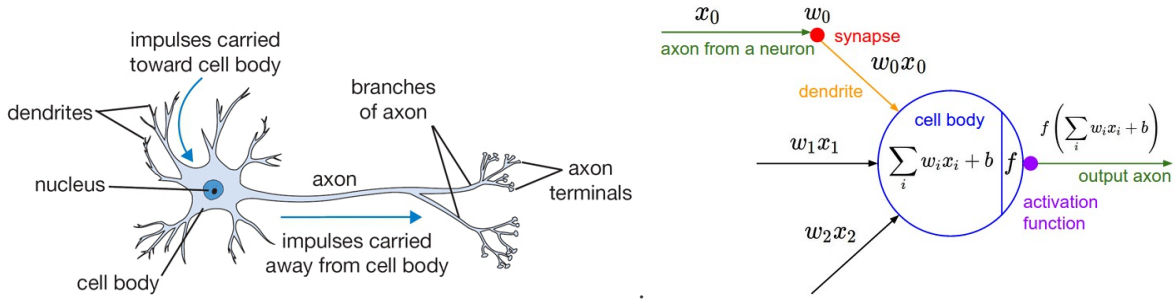

Figure 5. A visualization of the biological neuron (left) and the mathematical model of an artificial neuron (right).[8]

[7] Source: https://www.saedsayad.com/support_vector_machine_reg.htm (retrieved at 17-9-2018)
[8] Source: http://cs231n.github.io/neural-networks-1/ (retrieved at 17-9-2018)

In the computational model the analogy to this biological neuron is that each artificial neuron has one or more inputs that are weighted and summed with an added bias, there is then an activation function applied to this value which is analogous to a set threshold of when a neuron fires (see **figure 5**). Training an artificial neural network means optimizing these weights and biases for each neuron.

The computational model based on a biological neuron has already been proposed in the 1940s [43], but has only been popularized in recent years with additional algorithmic innovations (such as backpropagation [44]), increased computational power and more available data.

A neural network is typically visualized using a graph structure (see **figure 6**) where each node represents a neuron and the connections between neurons represent weights. By feeding data into the network from left to right, we get a certain output that can be compared to the expected output using a cost function. For example, using the Mean Squared Error (MSE):

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

Where N is the number of examples, y_i is the true value of example i and $\hat{y}_i$ is the predicted value of example i. We calculate the derivative of this cost with respect to the weights and biases of the model to find out in which direction to change these parameters, in order to decrease the cost (this is called backpropagation). We can then iteratively improve the model by continually feeding a batch of data into the network and updating the weights and biases based on the calculated derivatives.

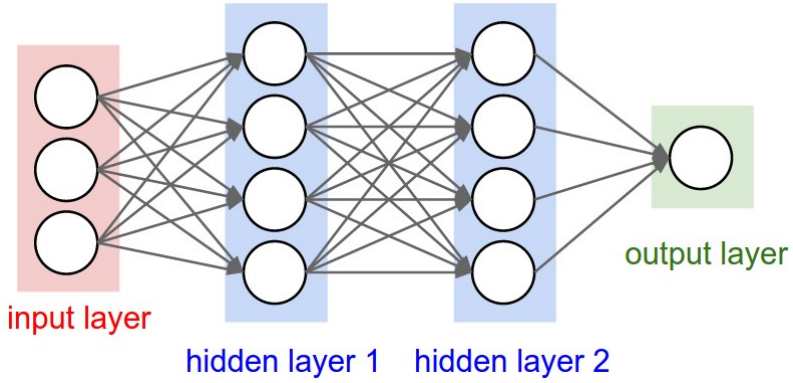

Figure 6. Graph visualization of a neural network.[9]

A common problem with neural networks is *overfitting*. This means that the network is too much adapted to the noise of the training data and thus won't perform well on unseen data (it doesn't

[9] Source: http://cs231n.github.io/neural-networks-1/ (retrieved at 17-9-2018)

generalize well). One way to try to solve this issue is with *weight regularization*. This means putting a cost on the weight parameters and thus giving the network an incentive to keep the weights low. Another more recent method to avoid overfitting is *dropout* [47]. In this method a certain percentage of randomly selected neurons is not taken into account during each training iteration. This avoids co-dependence between neurons and makes it harder for the network to overfit on the training data.

2.3.4. Recurrent Neural Network

Another issue of regular artificial neural networks is that it is impractical to apply to sequential data. Let's say that we want to predict the next word in a sentence using neural networks. This would require information about previous words in the sentence. So we could decide to use the previous five words as an input to the network and attempt to predict the next word. However, maybe information from a few sentences earlier is required to know which word comes next. For example in the text "I was born and raised in the Netherlands [...]. I speak fluent Dutch" the network could only predict the word "Dutch" using information that came earlier.

To solve this issue we can use a recurrent neural network (RNN) architecture. In this architecture the neurons in the hidden layer can receive an additional input from its own previous *state* (see **figure 7**). This previous state is also connected using weights that can be learned through backpropagation. In this way it is possible to preserve information about earlier inputs while only feeding the network data about one timestep.

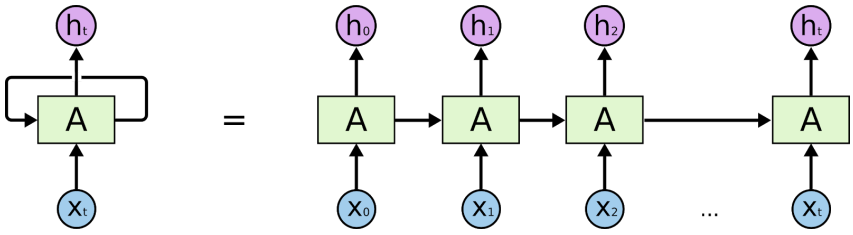

Figure 7. An unrolled recurrent neural network. [10]

A common problem experienced with this architecture is the *vanishing gradient problem*, this is the phenomena that as the network computes the gradient of the cost function based on an input many timesteps into the past, the gradient can vanish (become very small) due to a lot of computation steps between the output and an earlier input. In practice this means that it is hard for the network to learn long term dependencies.

[10] Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (retrieved at 17-9-2018)

2.3.5. Long short-term memory networks

To solve the issue of vanishing gradients and to make it easier for a neural network to learn long-term as well as short-term dependencies, an adaptation to the regular recurrent neural network (RNN) architecture has been proposed by Hochreiter et al. [26].

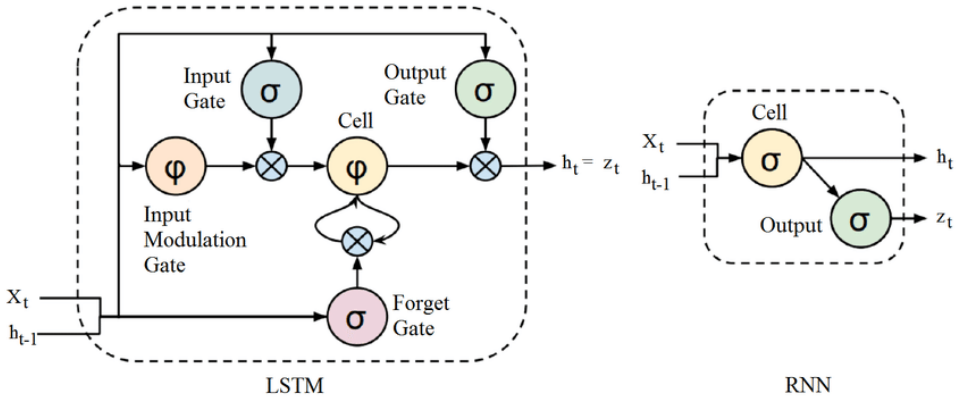

Figure 8. LSTM architecture using different *gates* (left) vs. the default RNN architecture (right).[11]

This architecture, called long short-term memory (LSTM) network, extends the default RNN by adding so-called *gates* to the hidden layer (see **figure 8**). These gates are basically just extra weight parameters that are used by the network to determine what information of the previous hidden state and of the current input to use, what information to forget and what information to remember. Since these weights are all differentiable they can be optimized using backpropagation as well.

[11] Source: https://github.com/llSourcell/LSTM_Networks/blob/master/LSTM%20Demo.ipynb (retrieved at 17-9-2018)

3. Related work

3.1. Predicting blood glucose

There are two recent studies that aim to do smartphone-based personalized blood glucose prediction [1, 11] and also aim to create an application that is able to help type 2 diabetes patients manage their disease by predicting future blood glucose levels.

S. Chemlal et al. [1] have done this by using previous blood glucose values (manually entered) and also physical activity (based on the accelerometer in the smartphone), they report an average accuracy of 90.24%. However, they don't mention any prediction horizon on this result, which is an important factor in the accuracy of these algorithms.

In "Smartphone-based personalized blood glucose prediction" [11] the blood glucose prediction is also based on manually entered blood glucose values and physical activity, but they also use nutrition and sleep. They claim to use a combination of patient-based and population-based analysis to come to a more accurate prediction of blood glucose. However they are using a small and quite artificial dataset [12] and it seems like they didn't use a separate test and validation set, making reported performance improvements by for example clustering potentially invalid (could simply be overfitting to the dataset). Also they only have a visual prototype of their application. it is not clear that results from the paper transfer to a real-world scenario.

In [48] an algorithm is developed to predict the occurence of hypoglycemia (too low blood glucose) using machine learning, but they don't attempt to directly predict blood glucose levels.

Because of the importance of insulin regulation for type 1 diabetes patients, there are more studies with the aim of accurately predicting blood glucose levels of type 1 diabetes patients. Some studies assume that the performance of models used for type 1 diabetes roughly transfer to type 2 diabetes [50], since the dynamics are very similar. Other studies don't differentiate between the two types at all [11]. But in the case of certain models, type 2 diabetes is actually harder to model than type 1 diabetes, since the model also has to take into account insulin that is still being produced by the body (which is not a factor in adult type 1 diabetes patients).

Figure 9 gives an overview of the performance of various models that aim to predict blood glucose of type 1 diabetics on varying time horizons, ranging from 15 to 180 minutes into the future, and with a variety of input variables.

Glucose prediction models based on artificial Intelligence and Autoregressive Models with time varying parameters [97]

Model	Input Space	No. of T1DM Patients (Monitoring Period)	Evaluation Results
Multilayer FNN [77]	CGMS data, Blood glucose readings, Insulin dosage, Carbohydrate intake, Hyperglycemic and hypoglycemic symptoms, Lifestyle (activities and events), Emotional states	18 (3-9 days)	PH (*min*)/ MAD (%): 50/6.7, 120/14.5, 180/18.9
FNN with 2 hidden layers [98]	**CGMS data**	9 (12 days) 6 (2 days)	PH (*min*)/RMSE(*mg/dl*): 15/10, 30/18,45/27
RBF NN [79]	Blood glucose readings, Insulin dosage, Food intake, Stress, Level of exercise	1 (77 days)	Interval/RMSE (*mg/dl*): morning/1.49, afternoon/0.92, evening/0.67, night/0.21
Wavelet NN [80]	Blood glucose readings, Insulin dosage, Food intake, Stress, Level of exercise	1 (77 days)	Interval / RMSE (*mg/dl*): morning/0.81 afternoon/0.63, evening/0.60, night/0.30
Neurofuzzy (applying wavelets as activation functions) [81]	CGMS data, Physical activity data from sensor	6 (7-15 days)	PH (*min*)/ RMSE (*mg/dl*): 15/14.42, 30/20.20, 45/24.79, 60/28.49
SOM[83]	CGMS data, Physical activity data from sensor	10 (6 days)	PH (*min*)/ RMSE(*mg/dl*):30/11.42, 60/19.58 120/31.00
SOM [83]	**CGMS data**	10 (6 days)	PH (*min*)/ RMSE(*mg/dl*): 30/12.29, 60/21.06120/33.68
SVR[82]	**CGMS data**	15 (5 – 22 days)	PH (*min*)/ RMSE(*mg/dl*):30/15.29, 60/24.19,120/33.04
Hybrid model based on the combined use of CMs and RNN [78]	CGMS data, Insulin infusion rates, Carbohydrates ingested	9 (10 days)	PH (*min*)/ RMSE (*mg/dl*): 30/18.34
Hybrid model based on the combined use of CMs and SVR[82]	CGMS data, Insulin dosages, Carbohydrates ingested, Physical Activity data from sensor, Time	15 (5 – 22 days)	PH (*min*)/ RMSE (*mg/dl*): 15/5.21, 30/6.03, 60/7.14, 120/7.62
Hybrid model based on the combined use of CMs and SOM [83]	CGMS data, Insulin Infusion rates, Carbohydrates ingested	12 (10 days)	PH (*min*)/ RMSE (*mg/dl*): 30/14.10, 60/23.19
Autoregressive models with time varying parameters [76]	**CGMS data**	28 (2 days)	PH (*min*)/ RMSE (*mg/dl*): 30/18.78

Figure 9. An overview of blood glucose prediction models for patients with type 1 diabetes [7].

It turns out that having more input features increases the accuracy of a prediction model and we expect this to also be the case for type 2 diabetes patients. As expected, the performance drops when the prediction horizon (PH) is increased. Although models in **figure 9** are not directly comparable, hybrid models using a combination of Compartmental Models (CM) and a data-driven model such as Recurrent Neural Networks (RNNs) seem to result in the lowest error [7].

3.2. Relevant features for prediction

There are many input features used in the literature to predict blood glucose levels. By far the most commonly used feature for predicting future blood glucose is (unsurprisingly) *previous blood glucose values,* which are used in practically every model [1, 11, 13-21]. After that **insulin dosage** or **insulin infusion rates** [13, 15, 16, 18, 19, 20], **carbohydrate intake** or **food intake** [11, 13, 15, 16, 18-20] and **physical activity** or the **level of exercise** [1, 11, 15-17, 20] seem to be the most commonly used features.

There are no studies that use the patient's **microbiome composition** as a feature in predicting blood sugar levels. Since Zeevi D. et al. [10] showed that there is a high correlation between the blood sugar response to specific food and an individual's microbiome composition, this could be an interesting novel feature to include. Gut microbiome compositions are relatively stable over time [22-25], so this is something that could still be collected for previous participants.

Other lesser used but perhaps helpful features are **time of day** [19], **stress** or **emotional states** [13, 15, 16] and **population statistics** [11].

Except for insulin dosage or infusion rates information from health records is not used as input in any of the studies. This is probably because most of these models are trained separately for each patient and thus don't benefit from population statistics such as BMI or age. A patient independent model could potentially benefit from this information.

3.3. Compartmental models

In compartmental modeling the human body is divided in a number of compartments which represent an organ or a body part [39]. In the case of blood glucose prediction knowledge of the human physiological processes is used to model the dynamics and transportation of insulin and glucose within the different organs (compartments) to determine the blood glucose level. In [19] two compartmental models are combined to predict glucose levels of type 1 diabetics:

1. Insulin model

In the insulin model the absorption of administered insulin is modeled by calculating the exogenous insulin flow I_{ex} at time t, using the following formula:

$$I_{\mathrm{ex}}(t) = B_d \int_{V_{\mathrm{sc}}} c_d(t, r) dV$$

(adapted from: [19])

Where B_d is a constant that defines the absorption rate, c_d is the insulin concentration in subcutaneous tissue (tissue directly below the skin), *r* is the distance from the injection point and V_{sc} is the total subcutaneous tissue volume. This exogenous insulin flow is then used to model the insulin concentration in blood plasma I_p using the following formula:

$$\dot{I}_p = \frac{I_{\mathrm{ex}}(t)}{V_d} - k_1 I_p(t) + k_2 I_h(t) + k_3 I_i(t)$$

(adapted from: [19)]

Here I_h is the concentration of insulin in the liver and I_p the concentration in the interstitial tissue. k_1, k_2 and k_3 are the rates at which insulin is eliminated from the blood plasma, liver and interstitial tissue. Because this model is aimed at type 1 diabetes patients, they don't take insulin produced by the pancreas into account. Since type 2 diabetes patients might still produce insulin themselves, a compartmental model aimed at type 2 diabetes would require some additional complexity.

2. Meal model

The meal model of Georga et al. [19] models the intestine as a single compartment in which the amount of glucose in the gut at time t after a meal that contained D carbohydrates is defined by:

$$\dot{q}_{\mathrm{gut}} = -k_{\mathrm{abs}} q_{\mathrm{gut}}(t) + G_{\mathrm{empt}}(t, D)$$

(adapted from [19])

Here k_{abs} is the rate at which glucose in the gut is absorbed and G_{empt} is the gastric emptying function, a trapezoidal function that increases or decreases with a 30 minute interval (see **figure 10**).

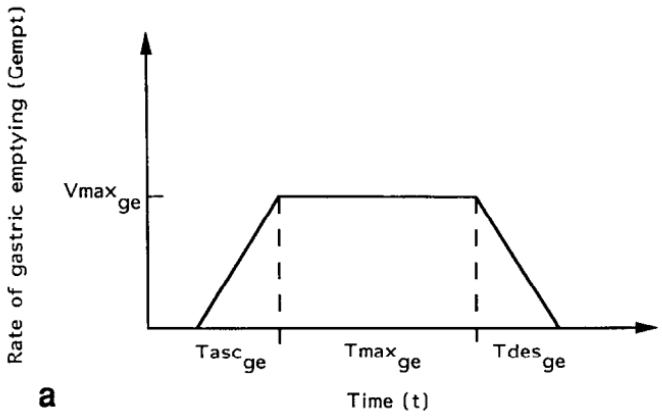

Figure 10. The gastric emptying function (adapted from [19])

The rate of appearance of meal-derived glucose in the blood is then determined by:

$$Ra(t) = k_{\mathrm{abs}} q_{\mathrm{gut}}(t)$$

(adapted from [19])

These two variables (Ra and I_p) are then used as input to the Support Vector Regression (SVR) model, besides previous blood glucose measurements.

There is also a more advanced compartmental model that models blood glucose for type 2 diabetes that also takes into account glucagon (a hormone that makes the liver and muscles release glucose) and incretin (a hormone released in the intestines after a meal intake) [39].

However, this model requires multiple blood measurements over a time of 250 minutes after an oral glucose tolerance test, in order to determine the parameters of the model for an individual patient, which is not feasible in our case.

3.4. Hybrid models

As shown in [7] hybrid models that use a combination of Compartmental Models (CMs) and a data-driven model such as an SVR, seem to outperform other models on the blood glucose prediction task.

In the best performing hybrid model [19] they use two compartmental models (described in the previous section) to "*simulate 1) the absorption and the kinetics and dynamics of s.c. administered insulin and 2) the absorption of ingested carbohydrates*". The output of these compartmental models is used as input in the data-driven Support Vector Regression (SVR) model that aims to make the prediction more patient-specific. However, since this is applied to type 1 diabetes patients they don't have to take insulin produced by the pancreas into account. In the case of type 2 diabetes this makes the CM more complex, as we don't know exactly how much insulin the pancreas is still producing and how insulin resistant the body's cells are.

3.5. Long short-term memory networks

As far as we can tell long short-term memory (LSTM) networks have not yet been applied to blood glucose prediction. LSTMs (described in section 2.3.5) are a specific kind of recurrent neural network that have been very successfully applied to a variety of tasks involving sequential data. LSTMs have achieved state-of-the-art performance in language-to-language translation [27, 28], generating image descriptions [29, 30], handwriting recognition [31, 32], protein structure prediction [33] and many more.

4. Methodology

In order to create a blood glucose prediction model that is able to accurately predict blood glucose values up to 120 minutes into the future, a variety of models are evaluated.

To set a baseline we first apply a simple autoregressive model, only using previous blood glucose values as input. We then compare this with more complex models such as compartmental models and support vector regression, as these have been shown effective in the prediction of blood glucose for type 1 diabetics [7]. However, our primary focus is on applying long short-term memory networks, because these have been very effective in related tasks dealing with sequential data [27-33] and haven't been extensively researched in the context of blood glucose prediction.

After training several models we determine if a separate model for each patient or a patient independent model is prefered based on the performance and considerations about which type would be most practical in a real world application.

Finally, we test the importance of input features by excluding a certain feature in the best performing model and observing how this affects performance. We also manually adapt the input of the model to find out how sensitive it is to changes in the input features.

4.1. Dataset description

DIALECT is a observational cohort study within ZGT that started in 2009 in which lifestyle effects such as food intake and physical activity will be monitored for 850 patients with type 2 diabetes.[12]

Currently around 700 patients have joined this cohort study and over the years the data that is recorded of these patients incrementally increased. The newest 80 participants within this study have also been equipped with a blood glucose measuring device (Freestyle Libre[13]), recording the patient's blood glucose level every 15 minutes and the patient's step count and heart rate every minute for a period of two weeks. This was done in a blind study, meaning the patients were not able to see their own blood glucose values, because this might influence their behaviour. These 80 patients have also been requested to keep a detailed log on their food intake during these two weeks. Our dataset only includes the 60 patients that joined at the start of this thesis.

[12] source: https://www.zgt.nl/patienten-en-bezoekers/onze-specialismen/wetenschap/visie-op-onderzoek/medische-disciplines/diabetes-mellitus/ (retrieved on 21-3-2018)

[13] https://www.freestylelibre.nl/ (retrieved on 21-9-2018)

Dataset Characterstics	
General	
No. features	44
No. patients	60
No. measurements	64,429
No. days	671
# days per patient	
Mean ± Std	11.2 ± 2.0
Range	5.0-13.8
Missing data (%)	
Missing heart rate	19.27%
Missing steps	5.69%
Missing food intake	24/60 (37%)
All blood glucose values - mg/dL (mmol/L)	
Mean ± Std	148.5 ± 57.7 (8.2 ± 3.2)
Range	39.6-496.8 (2.2-27.6)

Patient Characteristics	
Gender	
No. Female	21
No. Male	39
Age (years)	
Mean ± Std	64 ± 12
Range	30-82
BMI (kg/m²)	
Mean ± Std	31.7 ± 4.7
Range	21.3-44.2
Years suffering from T2DM	
Mean ± Std	9 ± 12
Range	0-36
Avg. blood glucose per patient - mg/dL (mmol/L	
Mean ± Std	149.5 ± 33.1 (8.3 ± 1.8)
Range	98.0-252.4 (5.4-14.0)

Figure 11. Dataset statistics. On the left side of the table data is shown about the entire dataset (such as the mean blood glucose over all measurements) and on the right side data about patient characteristics are shown (such as the mean of the average blood glucose level of each patient).

Figure 11 shows some interesting statistics about the data that has been collected on these 60 patients. Not all data has been successfully collected, some of the patients did not keep track of their food intake and the patients that did track their food intake, did not always do this very accurately (sometimes meals were skipped, the time was not recorded or the description was not specific enough). Also some of the steps and heart rate data is missing due to needing to charge the Fitbit or because a Fitbit without heart rate sensor was used.

To give a better insight in what this blood glucose data typically looks like, we plotted the blood glucose of a randomly selected patient over a period of three days (see **figure 12**). We also plotted the carbohydrate intake to show how this affects the blood glucose levels of this diabetes patient.

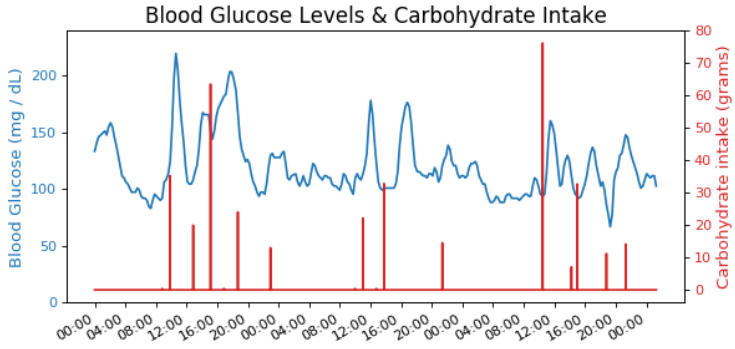

Figure 12. Blood glucose levels (blue) and carbohydrate intake in grams (red) for a randomly selected patient over a period of three days. As can be observed a carbohydrate intake is often followed by a blood glucose peak. Also, during the night blood glucose is often more stable than throughout the day.

4.2. Performance metric

To evaluate the performance of our models we use the Root Mean Squared Error (RMSE) since this is widely used in research on blood glucose prediction (making it easier to compare) and because it puts higher weight on more extreme errors of the model which is suitable to our use case. The Root Mean Squared Error is calculated as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}$$

4.3. Training scenarios

There are two scenarios that we consider for training our model.

1. The **patient dependent** scenario; in this case we train and evaluate a model for each patient separately. Training the model on the first N-100 measurements of a certain patient and evaluating the model on the last 100 measurements. We then use the average RMSE over all patients as our evaluation metric for a certain model. Because we don't have the same number of collected measurements for each patient, the amount of data used for training varies per patient.

2. The **patient independent** scenario; in this case we train a model on 54 patients and then validate the model on the remaining 6 patients. We then perform cross-validation over the other 9 folds and use the average RMSE over these folds as our evaluation metric for a certain model (see **figure 13**). We use this approach in order to get results of high significance (because we can use 9 samples to determine the RMSE) while still being able to use a separate validation set. This allows us to maximally benefit from the limited amount of available data. As there are still new patients participating in the cohort study, these could serve as an additional test set in the future.

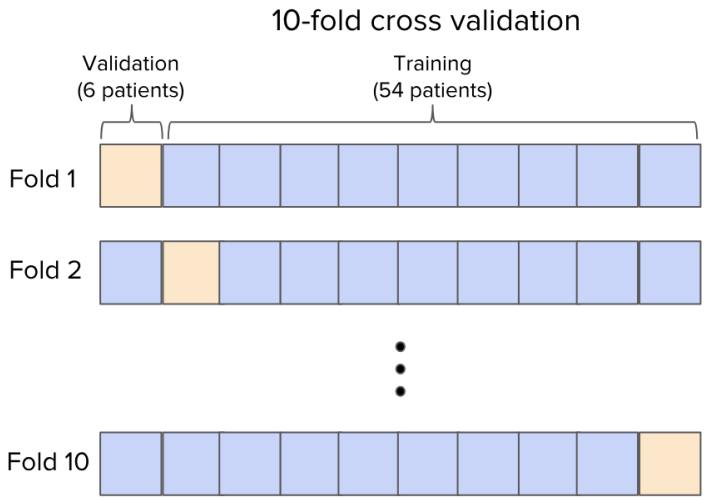

Figure 13. Performing 10-fold cross validation. We use fold 1 as validation data to choose the best hyperparameters and fold 2-10 as test data to evaluate the performance of a model.

A patient independent model has the advantage that we don't have to train a new model for each patient. However, this most likely comes at a loss of accuracy as it is harder to model patient-specific dynamics. The patient independent scenario also has the advantage that more data is available and that we can perform cross-validation (which is harder in the patient dependent case because we are using sequential data and sequential models).

4.4. Data preprocessing

In order to train different models on the available data, we first have to preprocess the data in such a way that it is easily fed to the algorithm. There are various data sources that we have to combine:

- medical records (one file that includes all patients);
- blood glucose data (a separate file for each patient);
- steps data (a separate file for each patient);
- food records (one file includes all patients that have been processed through a web app, for other patients there is a separate file from "Eetmeter" with added date and time).

Since the datasets are in different formats and separate files, we process the data in Python in order to get one file per patient that contains all available data with one row per glucose measurement. For steps and food intake data we take the sum of all data points since the previous glucose measurement, for heart rate we use the average. As a time interval we use 15 minutes, since the Freestyle Libre records a blood glucose value every 15 minutes.

4.4.1. Missing data

Lipton et al. tested several approaches for dealing with missing data in LSTMs using a variety of different models on several performance metrics. They concluded that adding a binary feature to indicate that data is missing resulted in the best performance on all metrics [34].

That is why we define an additional boolean variable for missing heart rate data that is set to "1" if heart rate is "0" or if there is no heart rate sensor on the device. We also define a boolean variable for missing steps data that is set to "1" when there is no steps data available. Finally we define a boolean variable to indicate when there is no food intake data available for a certain patient, determined by the fact that there are no food intake records for the patient.

4.4.2. Food intake data

Patients manually keep track of their food intake on paper logs, in these logs patients write down what they eat and at what date and time. It is not clear if the recorded time is the start or the end of a meal and the duration of a meal is also not taken into account.

These logs have been processed through an app called ‘Eetmeter’[14] by comparing each recorded meal to the available products in the database. However, this is an inconvenient and time-consuming process because ‘Eetmeter’ doesn’t provide the option to add a time to your input and thus the time has to be added manually to the exported file. Also ‘Eetmeter’ outputs only 27 nutritional values, while the Dutch food nutrition database called ‘NEVO’ [5] has 136 nutritional values. For example ‘Voedingscentrum’ only provides carbohydrates while ‘NEVO’ provides carbohydrates as well as ‘of which sugars’, which might be useful information in the prediction task.

To solve these issues, we developed our own food input tool[15] increasing the speed at which patient's food logs can be processed and using the nutritional information from 'NEVO'. This tool has been used to process most of the patient's food logs. It could potentially also be used by patients directly to keep track their food intake, saving researchers time a lot of time on processing food logs. However, because a significant part of the patients' food intake data was solely processed using 'Eetmeter', we were still unable to take advantage of the additional nutritional information provided by 'NEVO'.

[14] https://mijn.voedingscentrum.nl/nl/eetmeter/ (retrieved at 19-9-2018)
[15] https://daviddemeij.pythonanywhere.com (retrieved at 17-9-2018)

To preprocess the food intake data, we first load the patient's data from the appropriate file (either a file generated through 'Eetmeter' or a file exported from our own food tool). We then loop through the blood glucose measurements and sum the nutritional content of all recorded food intake since the previous blood glucose measurement and use this as input features.

4.4.3. Health records

For the health records (see **figure 14**) there are a lot of features that could potentially be included, but we will keep only it to a few basic features that we believe might have a significant correlation with blood glucose dynamics:

- Gender
- Age
- BMI
- HbA1c (glycated hemoglobin; value measured in a blood sample that represents the average blood sugar in the previous weeks)
- Fast-acting insulin (A10AB[16]) and Intermediate-acting insulin (A10AC[17] & A10AD[18]) dosage
- Metformin (A10BA02[19]) dosage
- Number of years since diagnosed with Diabetes Mellitus type 2

Subjectnr	Geslacht	Leeftijd_poli1	Jaren_DM2	Lengte_poli1	Gewicht_poli1	SerumHbA1c_1	dosA10AB	dosA10AC	dosA10AD	dosA10BA
572	1	78	15	158	80	60	62			1000
574	0	49	5	178	94	59	5			2000
616	0	79	2	173	94	54				1000
617	0	68	30	180	81	53	33			1000
618	0	48	9	187	111	74	48			3000
619	0	76	20	164	96	41	92			2000
612	0	66	10	173	93	53				
614	0	51	2	171	106	53				2000
621	0	65	6	176	110	57				1500
620	0	75	5	167	105	65				
609	0	68	18	171	88	57				1000
610	0	53	19	176	88	62				2000
622	1	50	15	154	88	61	68			1000
611	1	64	15	176	101	58	34			500

Figure 14. Sample of the available health records data.

Since the three different types of insulin that we are interested in are prescribed exclusive from each other, we sum these three variables into a single feature.

[16] https://www.whocc.no/atc_ddd_index/?code=A10AB
[17] https://www.whocc.no/atc_ddd_index/?code=A10AC
[18] https://www.whocc.no/atc_ddd_index/?code=A10AD
[19] https://www.whocc.no/atc_ddd_index/?code=A10BA02

4.4.4. Final processed dataset

The final processed dataset contains a value every 15 minutes for all patients for the following features.

Measured using a Freestyle Libre:

- **datetime** from the date and time recorded by the Freestyle Libre at each measurement;
- **blood glucose** as recorded by the Freestyle Libre;
- **seconds elapsed** since previous measurement;
- **hour of day** an integer between 0 and 24 based on the datetime.

Measured using a Fitbit:

- **missing heart rate** a boolean that is either 0 or 1 based on whether any heart rate is recorded;
- **heart rate** averaged over the period since the previous blood glucose measurement;
- **missing steps** a boolean that is either 0 or 1 based on whether step data is missing;
- **steps** summed over the period since the previous blood glucose measurement.

Retrieved from the processed food logs (all summed over the period since the previous blood glucose measurement):

- Energy (kcal)
- Fat (g)
- Saturated fat (g)
- Carbohydrates (g)
- Protein (g)
- Fiber (g)
- Alcohol (g)
- Water (g)
- Natrium (mg)
- Salt (g)
- Kalium (mg)
- Calcium (mg)
- Magnesium (mg)
- Iron (mg)
- Selenium (µg)
- Zinc (mg)
- Vit. A (µg)
- Vit. D (µg)
- Vit. B1 (mg)
- Vit. B2 (mg)
- Vit. B12 (µg)
- Nicotinic acid (mg)
- Vit. C (mg)
- Vit. B6 (mg)
- Folic acid (µg)
- Iodine (µg)
- Vit. E (mg)

Retrieved from the patients health records::

- **Gender** 0 for male and 1 for female;
- **Age** at the time of joining the cohort study;
- **Years suffering from diabetes type 2** based on the moment of diagnosis;
- **Body Mass Index (BMI)** value calculated based on weight and height;
- **HbA1c** glycated hemoglobin a value measured in the blood that indicated the average blood glucose concentration;
- **Sum dosage A10A*** a sum of the prescribed dosage for insulin types A10AB, A10AC and A10AD;
- **Dosage A10BA** the prescribed dosage of Metformin.

Using our preprocessing code described in **Appendix B** we obtain a large matrix with the dimensions:

$$number\ of\ patients \times number\ of\ timesteps \times number\ of\ features = 60 \times 1321 \times 46$$

Number of timesteps refers here to the maximum number of glucose values that has been recorded by a single patient. Since not all patients have recorded 1321 blood glucose values (this is equal to 13.75 days), the matrix is padded with zeros.

4.4.5. Normalization

For our neural network we want the input of the model to be between 0 and 1 as this has been shown to make neural networks converge faster and decrease the likelihood of getting stuck in a local optima [45]. To achieve this we normalize all the data by applying min-max normalization to each feature as well as to the output:

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)}$$

Where for each feature $x = (x_1, \ldots, x_n)$, z_i is the normalized value of x_i.

4.5. Compartmental models

4.5.1. Modeling rate of appearance (Ra)

In order to attempt replicating the compartmental model experiments as described in section 3.5, we have to model the rate of appearance (Ra) of exogenous glucose in the blood. We can model the formulas described in section 3.5 using a object-oriented Python script (see **Appendix A**). A sample of the resulting data can be seen in **figure 15**.

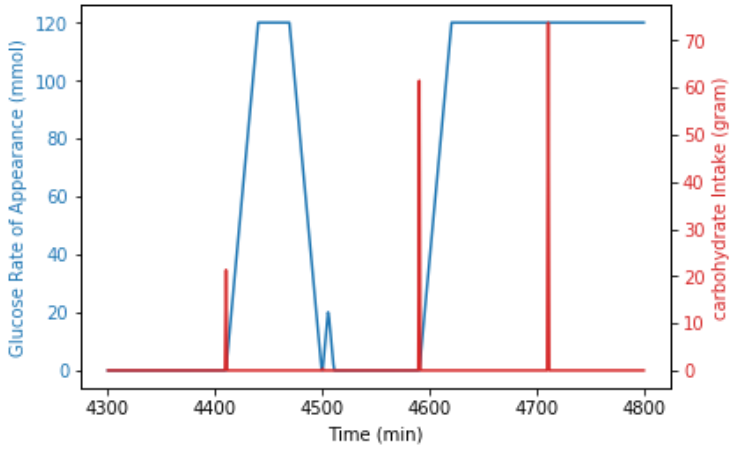

Figure 15. A typical eight hour period modeling the rate of appearance of exogenous glucose (blue) in the blood plasma based on the carbohydrate intake (red). After the Ra reaches its maximum (a fixed model parameter), it will stay there until most glucose is absorbed.

4.5.2. Modeling sum of rate of appearance (SRa)

The sum of the rate of appearance is calculated as an additional feature, as it might be useful in taking into account the amount of glucose absorbed by the blood over a longer time period [19]. This is easy to model - as we already modelled the rate of appearance (Ra) - by summing the values of Ra over the previous 90 minutes (see **figure 16**).

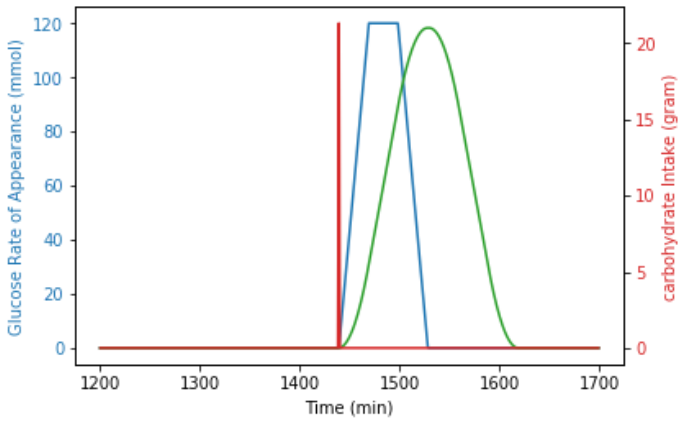

Figure 16. Modelling the sum of the rate of appearance over the previous 90 minutes (in green on a separate y-scale).

4.6. Training procedure

4.6.1. Autoregressive model

For an autoregressive model the training procedure is quite straightforward. There is only one hyperparameter that we have to set which is the number of previous values that the model uses to make a prediction. Thus we can simply optimize the parameters of the model on the training data and then evaluate the performance on the validation data for different values of this hyperparameter. In the patient dependent case this means evaluating the results on the last 100 data points and determine the average over all patients. In the patient independent case this means cross-validating the results over 9 different sets of training and validation data.

4.6.2. Support vector regression

For support vector regression the training procedure is less straightforward as there are three hyperparameters that we can tune. In this case we use the same approach as in [19], applying a Differential Evolution algorithm to the hyperparameter selection. This involves "*maintaining a population of candidate solutions subjected to iterations of recombination, evaluation, and*

selection. The recombination approach involves the creation of new candidate solution components based on the weighted difference between two randomly selected population members added to a third population member." [20] To evaluate a candidate we use a separate part of the training data and only the final candidate is evaluated on the test data. It is unclear that this is done properly in [19], meaning the positive results of this paper might be exaggerated due to overfitting.

4.6.3. Long short-term memory networks

For neural networks and LSTMs in particular there is a large amount of hyperparameters that can be set and also a few architectural choices that have to be made, among which:

- What features to use as input to the network. Using more input features gives more information that the network might be able to use to make a better prediction. But when we use more input features we also introduce more noise and increase the chance of overfitting.
- The number of layers in the network. More layers increases the computational complexity and makes it possible for the network to learn a higher level of abstraction. However, more layers can also make it harder for the network to converge to a solution.
- The number of neurons per layer. Using more neurons increases the computational complexity and memory of the network, but also makes it more susceptible to overfitting the training data and makes the network require more data to converge to a solution.
- The amount of dropout or weight regularization to apply. A higher value reduces overfitting, but also makes it harder for the network to converge to a solution.
- The learning rate (the size of the update to the weights during each iteration). A higher learning rate can increase the speed at which the network learns, but if it is set too high we might overshoot the desired weights making the network unable to converge to a solution.

We use the first fold to train and evaluate different models and we do this for many different hyperparameter settings and with different architectural set-up. The best performing set-up is then cross-validated on the other 9 folds (different combinations of training data and test data) and the results are averaged over all 9 folds.

To determine which input features are important for the network 500 LSTMs with randomly selected features are trained and evaluated. For each of the 500 experiments a feature is set to -1

[20] http://www.cleveralgorithms.com/nature-inspired/evolution/differential_evolution.html

if it is not used in the network and to 1 if it is used. Using these values the Pearson correlation coefficient between each feature and the validation RMSE can be calculated using:

$$r = \frac{\sum_{i=0}^{500}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=0}^{500}(x_i-\bar{x})(y_i-\bar{y})}}$$

Where r is the Pearson correlation coefficient and x_i and y_i represent the value of the feature for experiment i (either -1 or 1) and the validation RMSE for experiment i, respectively.

4.7. Transfer learning

A popular technique in the field of Machine Learning is training one neural network on multiple related tasks (in succession or at the same time) with the idea that the knowledge learned in one task can improve the performance of the other task.

In the case of blood glucose prediction we can apply this technique by training one network that predicts blood glucose level for 30, 60, 90 and 120 minutes at the same time, instead of training a separate LSTM for each of these prediction horizons.

4.8. Model ensembles

Another popular method to improve performance of neural networks is training several models and then combine the output of the models to obtain the final prediction. An obvious downside to this method is that it requires training multiple models which can be time consuming and also increases the computation and memory required to make a prediction (making it for example less suitable to run on a smartphone).

4.9. Testing statistical significance

In order to test the statistical significance of the correlation coefficient described in 4.6.3 we use a t-test. However, in conducting a t-test certain assumptions are made:

1) The data is measured on an interval or ratio scale (meaning it increases with a equal intervals).
2) The samples are randomly selected.
3) Samples are independent from each other.

4) Variances are approximately equal.

Assumption 2 to 4 hold in the case of correlation coefficients. However, assumption 1 does not hold and to solve this we apply a Fisher Z-transformation [49] which is defined as:

$$z = arctanh(r)$$

We can use this Z-score to calculate the probability that the correlation r of a feature is lower than zero ($P(r < 0)$) and thus useful to the model by using a one-tailed t-test. A 95% confidence interval of the Z-score is then defined by:

$$[z - 1.96 * \frac{1}{\sqrt{500–3}}, z + 1.96 * \frac{1}{\sqrt{500–3}}]$$

To obtain the 95% confidence interval of the correlation coefficient we just have to transform the obtainer Z-scores for the lower and upper bound back using:

$$r = tanh(z)$$

To evaluate if a model performs significantly better than the baseline we can perform a standard one-tailed t-test without transforming the input space, because the results are assumed to lie on a regular normal distribution with equal intervals already.

5. Results

In this section the results of the most relevant experiments are described and discussed. Models to experiment with are selected based on related work and by doing some initial experiments. If these initial experiments give any promising results we conduct a full evaluation. As a patient independent LSTM showed most promise for our use case, we decided to attempt to further improve the performance of this model by trying some popular techniques such as: adding noise, transfer learning and model ensembles. To better understand the model we test the importance of the features to model's performance by excluding a certain feature and evaluate the effect on the performance. We also experiment with the sensitivity of the model to each feature by manually adapting the input values. To get a better sense of how the trained model can be used in a real world application, we visualize the predictions from the perspective of the patient, also showing the expected error of our predictions.

5.1. Comparing models

When experimenting with various models in practice it turns out that some of the models don't work in the patient dependent case and some don't work in the patient independent case. In the patient dependent case Recurrent Neural Networks don't work because two weeks of data is simply not enough to train such a complex model. In the patient independent case Support Vector Regression does not work because the dataset is too large, fitting a SVR models exponentially increases in complexity as we have more data points. The autoregressive model can easily be applied in both cases.

5.1.1. Patient dependent models

For the patient dependent model we solely focus on 36 patients of which we have food intake data, as this allows us to better evaluate how much food intake data improves the performance of these models.

Model RMSE \ Pred. Horiz. (minutes)	**30**	**60**	**90**	**120**
Autoregressive Model (baseline)	17.43	28.53	34.70	38.32
Support Vector Regression (SVR)	18.25	29.21	34.81	37.40
SVR + Ra	19.65 ± 6.0	28.82 ± 8.1	33.77 ± 10.8	36.43 ± 12.2
SVR + Ra + SRa	19.54 ± 5.73	28.73 ± 8.3	33.60 ± 11.0	36.86 ± 13.4

SVR + Ra + SRa + time	20.70 ± 6.7	29.28 ± 10.6	33.95 ± 12.6	37.22 ± 14.8
SVR + Ra + SRa + time + steps	21.77 ± 10.9	31.47±13.2	36.04 ± 14.6	38.02 ± 15.5

Table 1. Patient dependent performance of various models on the last 100 timesteps (average over all 36 patients that recorded food intake).

For the longer time horizons Support Vector Regression seems to works slightly better than a simple autoregressive model (see **table 1**), but as there is a high standard deviation these improvements are not significant. Adding a compartmental model that models the exogenous blood glucose rate of appearance (Ra) also slightly, but not significantly, improves this performance. Adding additional information such as the time of day and step count does not seem to help at all.

We also attempted applying Recurrent Neural Networks in the patient dependent case. However, the initial results of this were not very promising. Since training and evaluating Neural Networks in the patient dependent case is very time-consuming (because we need to train and evaluate a separate Neural Network for each patient) we decided not to put any more time in this.

5.1.2. Patient independent models

In the patient independent case the regular LSTM model significantly outperforms the autoregressive model on a short time horizon (see **table 2**). A multitask LSTM using transfer learning (described in section 4.6) significantly outperforms the autoregressive baseline on longer time horizons (>= 60 minutes). This not only results in better performance, but it also saves time, since only one network has to be trained instead of four. It is also convenient that we can directly use this multitask LSTM to create a graph showing the predicted blood glucose for the upcoming two hours. An ensemble of multitask LSTMs (described in section 4.7) has the best performance on longer time horizons (>= 60 minutes), but doesn't significantly improve the regular multitask LSTM. The code (including the hyperparameters) of the best performing multitask LSTM kan be observed in **Appendix C**.

Model RMSE \ Pred. Horiz. (minutes)	**30**	**60**	**90**	**120**
Autoregressive Model (baseline)	19.53 ± 1.8	32.06 ± 3.3	39.45 ± 4.7	44.24 ± 5.8
LSTM	**18.21 ± 1.6**	30.76 ± 3.1	36.95 ± 4.4	41.47 ± 5.4
Multitask LSTM	19.33 ± 1.7	**30.01 ± 3.1**	**36.21 ± 4.6**	**40.22 ± 5.8**
Multitask LSTM ensemble	19.26 ± 1.5	**29.92 ± 3.1**	**34.98 ± 4.1***	**40.08 ± 5.7**

Table 2. Patient independent 9-fold cross-validated performance of various models. Results that are significantly better than the baseline are bolded (calculated using a one-sided t-test with p < 0.1 and * indicating p<0.05).

5.2. Feature selection

A challenging task in training our LSTM is selecting the most useful features. We evaluate this by training 500 networks on randomly selected features and observe how often certain features are used in the 20 best performing models (see **table 3**).

We also calculate the correlation between each feature and the Validation RMSE as described in section 4.6.3. In this case a negative correlation is good since this means using the feature decreases the RMSE. We can then calculate a 95% confidence interval for this correlation and calculate the probability that the correlation r of a feature is lower than zero ($P(r < 0)$) and thus useful to the model (using the methods described in section 4.9). However, we must note that the features themselves are not independent. For example the amount of fat and the amount of saturated fat are quite similar; if we have fat as an input, also having saturated fat might be less useful than if we don't have fat as an input already. What is also interesting to note, is that using gender actually decreases the accuracy on average. This might be due to a gender atypical blood glucose pattern for one or more of the patients in the validation set. Since the validation set only consists of 5 patients, the usefulness of patient characteristics such as age, gender, BMI and HbA1c might not always be accurately represented.

Feature	Top 20	Correlation	Lower bound Correlation	Upper bound Correlation	P(r < 0)
Time of day	20/20	-0.297	-0.375	-0.215	100.0%
HbA1c	6/20	-0.097	-0.183	-0.009	98.5%
Fibers	4/20	-0.079	-0.165	0.009	96.1%
Steps	8/20	-0.056	-0.143	0.032	89.5%
Time since measurement	6/20	-0.056	-0.143	0.032	89.3%
Saturated fat	3/20	-0.054	-0.141	0.034	88.7%
Energy (Kcal)	2/20	-0.043	-0.130	0.045	83.0%
Alcohol	3/20	-0.035	-0.122	0.053	78.1%
BMI	6/20	-0.030	-0.118	0.058	75.1%
Carbohydrates	4/20	-0.024	-0.112	0.064	70.6%
A10BA	1/20	-0.017	-0.105	0.071	64.8%

Fat	2/20	-0.004	-0.092	0.084	53.6%
Protein	2/20	-0.004	-0.092	0.084	53.4%
Missing food	6/20	0.008	-0.080	0.096	43.0%
Age	3/20	0.014	-0.074	0.101	38.0%
Missing HR	3/20	0.024	-0.064	0.112	29.7%
Missing Steps	1/20	0.029	-0.059	0.116	26.1%
Salt	7/20	0.031	-0.056	0.119	24.2%
Heart rate	2/20	0.057	-0.030	0.145	10.0%
Years diagnosed	0/20	0.081	-0.007	0.167	3.6%
Sum A10A	0/20	0.114	0.027	0.200	0.5%
Gender	3/20	0.136	0.049	0.222	0.1%

Table 3. Results of training 500 multi-task LSTM models on randomly selected features and validated on our validation fold. The top 20 shows how often a certain feature occurs in the 20 best performing models. The correlation column shows how much each feature is correlated to the validation RMSE. The lower and upper bounds columns show the 95% confidence interval of this correlation. The last column shows the probability that the correlation is negative (meaning it is useful to the model).

5.3. Using partial data

To test to what extend the model might improve with obtaining more data, we can observe what happens when we train the model on a percentage of the available data and gradually increase the amount of data that we feed the model.

Multitask LSTM	**30 min**	**60 min**	**90 min**	**120 min**	**Overall**
Using 25% of patients	23.43	33.05	39.11	43.14	35.67
Using 50% of patients	20.17	30.78	37.12	40.96	33.52
Using 75% of patients	19.65	30.30	36.72	40.83	33.12
Using 100% of patients	19.68	30.21	36.47	40.59	32.91

Table 4. Gradually increasing the amount of data fed to a Multitask LSTM (the reported results are the average RMSE of 9-fold cross-validation).

As expected the performance of the network improves as we use more data for training (see **table 4**). However, the increase in performance does seem to slow down as we add more data. It is not likely that the performance would improve significantly if we would have slightly more data.

5.4. Adding noise

A common way to make neural networks more robust to small perturbations to the input is by adding random noise to the input. The intuition behind this is that relative small changes to the input should generally not have very big effects to the output of a model. This should improve the generalization of the network and reduce overfitting. However, it turns out that in our case it does not improve the multitask LSTM model (see **table 5**).

Multitask LSTM	**30 min**	**60 min**	**90 min**	**120 min**	**Overall**
Noise = 0%	19.80	30.32	36.51	40.54	32.36
Noise = 1%	20.56	30.65	36.61	40.55	33.17
Noise = 2.5%	22.53	31.85	37.45	41.12	34.18
Noise = 10%	29.20	35.70	39.88	42.70	37.39

Table 5. Adding increasing amounts of noise to a Multitask LSTM model (the reported results are the average RMSE of 9-fold cross-validation).

5.5. Model ensembles

Using an ensemble of 5 models doesn't significantly improve the performance compared to the average RMSE when we apply these 5 models individually. However, this is using a simple ensemble method where we take the average prediction of the 5 models as the output. We could also use more intelligent ways to use the different models for example by training a Neural Network to weigh the output of the different models perhaps based on the age or gender of a patient.

Multitask LSTM	**30 min**	**60 min**	**90 min**	**120 min**
Average	19.51	30.31	36.56	40.57
Ensemble	19.26	29.92	34.98	40.08

Table 6. 9-fold cross-validated RMSE of using a 5 model ensemble compared to the Average RMSE of these 5 individual models.

5.6. Testing importance of features

An interesting question is how important the different input features are for the model's performance. In order to test this we can leave out a certain feature and observe how this affects the performance.

Multitask LSTM	30 min	60 min	90 min	120 min
Include carbohydrates & fat	19.22	29.49	35.01	38.73
Exclude carbohydrates & fat	19.13	30.67	35.30	38.92

Table 7. 9-fold cross-validated performance (RMSE) of the best performing multitask LSTM (solely evaluated on patients that recorded their food intake).

Even though food intake has been selected as a feature that benefits the performance, the usefulness of this feature seems to be very limited.

Multitask LSTM	30 min	60 min	90 min	120 min
Include all selected features	19.33	30.01	36.21	40.22
Exclude steps	19.89	30.27	36.43	40.46
Exclude time of day	20.12	30.85	37.43	41.84
Exclude Hba1C	19.20	29.81	35.89	39.89
Exclude all (except blood glucose)	19.16	30.91	37.76	42.19

Table 8. 9-fold cross-validated performance (RMSE) of the best performing multitask LSTM excluding certain features to see the usefulness of each feature.

Even if we exclude all features, the performance doesn't seem to be affected very much. Especially the 24-h time seems to be an important feature (and this is actually a feature that we can obtain without any additional effort).

5.7. Sensitivity analysis

Besides analyzing how much performance is improved by each feature, it is also interesting to observe how sensitive the model is to changes to the input of a certain feature. To analyze this we use a multitask LSTM and adapt the input data of a randomly selected patient from the test set. We visualize this by showing the predicted blood glucose graph for a certain day when we adapt the

input data. We also provide the mean and standard deviation of the blood glucose for different changes to the input as this also tells us something about how the prediction is affected.

5.7.1. Varying carbohydrate intake

As expected when we increase the carbohydrate intake, blood glucose has higher peaks and lower valleys and thus a higher standard deviation (see **figure 17** and **table 9**). Decreasing the carbohydrate intake to zero doesn't seem to have a large influence on the prediction. This might be because - as a lot of patients did not (accurately) keep track for their food intake - the network also learns to predict these post-meal blood glucose peaks by relying on the time of day.

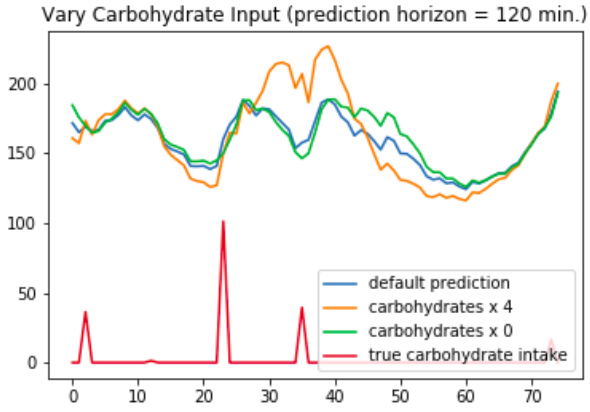

Figure 17. Varying the carbohydrate input for a random patient in the test set to observe how this influences the predicted blood glucose.

Input \ Predicted blood glucose	**Mean**	**Std. dev.**
True carbohydrate intake	150.49	22.32
Carbohydrate intake x 4	149.32	25.94
Carbohydrate intake x 0	153.76	23.53

Table 9. Effect of varying carbohydrate intake on the mean and standard deviation of the predicted blood glucose (prediction horizon = 120 minutes).

5.7.2. Varying fat intake

Varying the fat intake has a similar effect to changing the carbohydrate intake (see **figure 18**). What is interesting to note is that when we decrease the fat intake the standard deviation of the blood glucose actually increases slightly (see **table 10**). This makes sense because fat actually has been shown to slow down the glucose absorption of a meal [42].

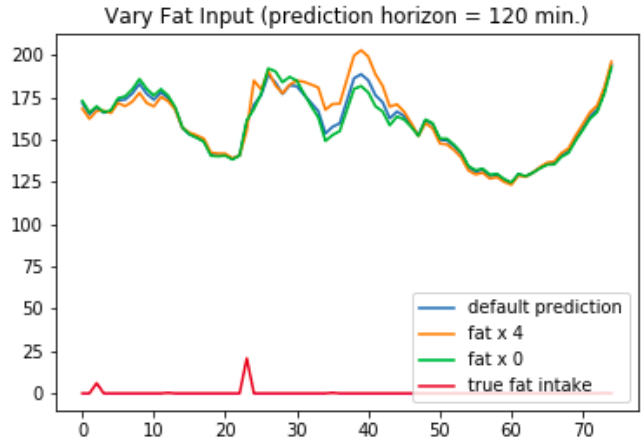

Figure 18. Varying the fat input for a random patient in the test set to observe how this influences the predicted blood glucose.

Input \ Predicted blood glucose	**Mean**	**Std. dev.**
True fat intake	150.49	22.32
Fat intake x 4	151.05	22.75
Fat intake x 0	153.76	23.53

Table 10. Effect of varying fat intake on the mean and standard deviation of the predicted blood glucose (prediction horizon = 120 minutes).

5.7.3. Varying fat and carbohydrate intake

It is also possible that the model takes certain dynamics between different features into account, so it might be interesting to see what happens if we change fat and carbohydrate intake at the same time. As expected when we increase both fat and carbohydrate intake at the same time, the mean predicted blood glucose is higher and the standard deviation is also increased (see **table 11**). When we increase carbohydrates and set fat at zero the standard deviation is also higher. When we set fat to zero and increase the fat content the standard deviation of the blood glucose goes down and blood glucose peaks are delayed. These findings are also in accordance with the research which shows that fat slows down glucose absorption of a meal [42].

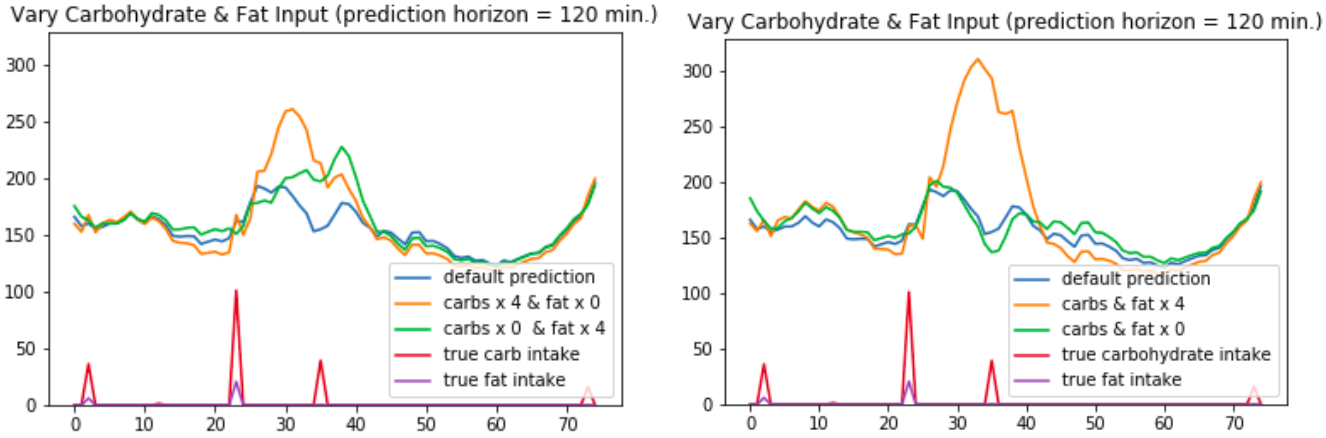

Figure 19. Varying the carbohydrate and fat input at the same time for a random patient in the test set to observe how this influences the predicted blood glucose.

Input \ Predicted blood glucose	**Mean**	**Std. dev.**
True carb & fat intake	149.24	23.35
Carbs x 4 & fat x 0	150.40	27.83
Carbs x 0 & fat x 4	151.15	24.43
Carbs x 0 & fat x 0	152.94	24.89
Carbs x 4 & fat x 4	154.29	33.45

Table 11. Effect of varying carbohydrate and fat intake on the mean and standard deviation of the predicted blood glucose (prediction horizon = 120 minutes).

5.7.4. Varying HbA1c

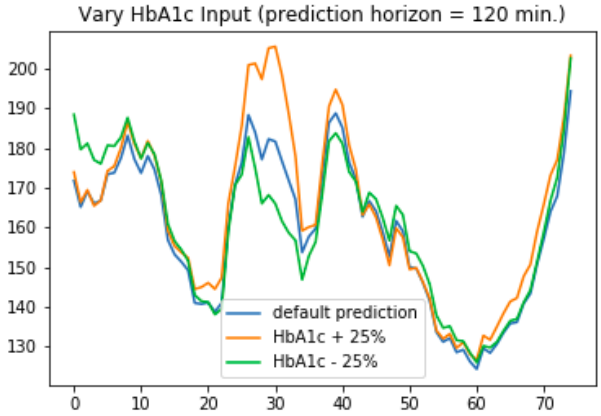

Figure 20. Varying the HbA1c for a random patient in the test set to observe how this influences the predicted blood glucose.

The model seems quite sensitive to changes in HbA1c value. As expected a higher HbA1c value translates to higher peaks and a higher mean blood glucose. What might be surprising is that a lower HbA1c value actually also has a higher standard deviation. This might be because a lower

HbA1c also increases the risk of hypoglycemia which would result in high blood glucose fluctuations or because such a low value does not occur in the training data.

Input \ Predicted blood glucose	**Mean**	**Std. dev.**
True HbA1c (53)	150.49	22.32
HbA1c + 50% (80)	163.39	28.65
HbA1c + 25% (66)	154.83	24.04
HbA1c - 25% (40)	153.74	25.88
HbA1c - 50% (27)	170.01	37.36

Table 12. Effect of varying HbA1c on the mean and standard deviation of the predicted blood glucose (prediction horizon = 120 minutes).

5.7.5. Varying steps

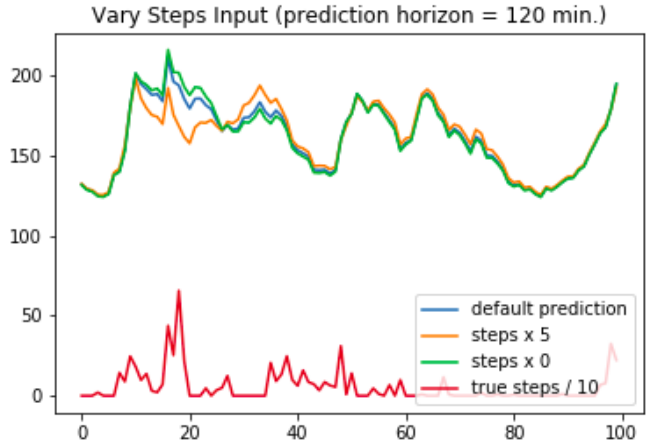

Figure 20. Varying step count input for a random patient in the test set to observe how this influences the predicted blood glucose.

Input \ Predicted blood glucose	**Mean**	**Std. dev.**
True Steps	150.49	22.32
Steps x 20	144.43	24.34
Steps x 10	148.33	21.55
Steps x 5	150.15	21.82
Steps x 0	150.31	22.56

Table 13. Effect of varying step count input on the mean and standard deviation of the predicted blood glucose (prediction horizon = 120 minutes).

Step count does not have a very large influence on the prediction, but as expected more steps results in a lower average blood glucose prediction. Increasing steps by a large factor actually results in a higher standard deviation, this might be caused by the large fluctuations and potentially unrealistic values that don't occur in the training data.

5.7.6. Varying age

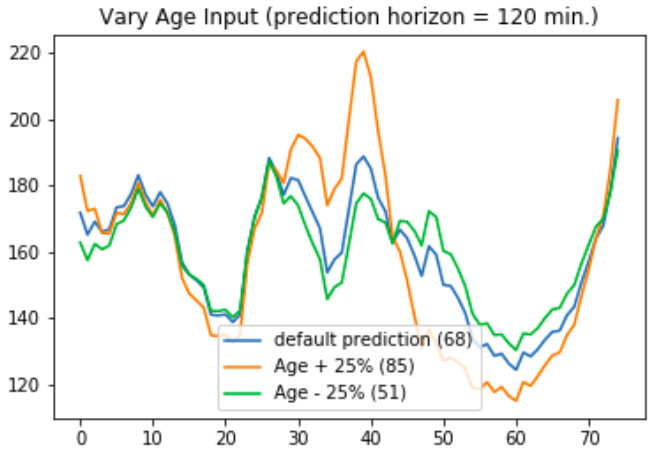

Figure 20. Varying age for a random patient in the test set to observe how this influences the predicted blood glucose.

Age seems to be an important factor for the network. As expected a lower age means lower predicted average blood glucose levels and most of all a lower standard deviation.

Input \ Predicted blood glucose	**Mean**	**Std. dev.**
True Age (68)	150.49	22.32
Age + 50% (102)	159.26	40.24
Age + 25% (85)	151.75	27.59
Age - 25% (51)	149.95	19.85
Age - 50% (34)	149.87	18.49

Table 14. Effect of varying age on the mean and standard deviation of the predicted blood glucose (prediction horizon = 120 minutes).

5.8. Visualizing predictions

In a real world application the patient should be made aware of the limitations of our model, while still benefiting from seeing the effects that their actions will have on their blood glucose. We can realize this by indicating an area in which we expect the real future blood glucose value to be in, using the cross-validated RMSE as a margin (see **figure 21** or go to

http://daviddemeij.pythonanywhere.com/static/visualizing_prediction.gif for an animation throughout the day). This margin is larger for predictions further into the future because the RMSE is also higher for longer time horizons.

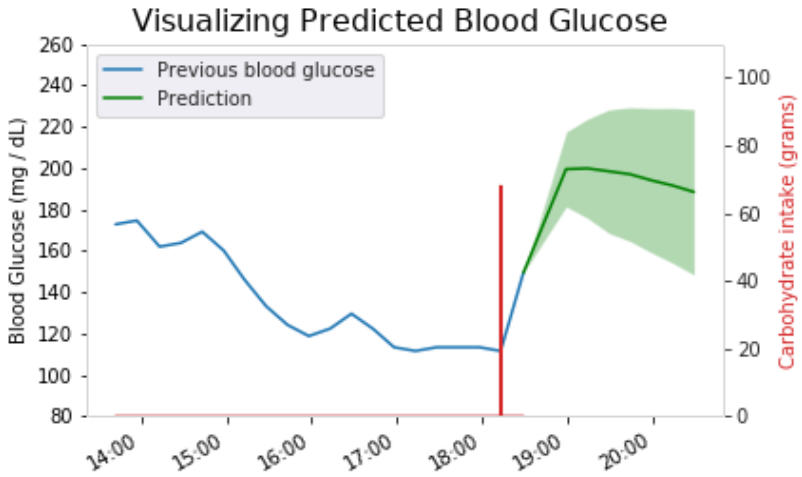

Figure 21. Predicted blood glucose for the upcoming 120 minutes for a random patient with a margin that has a width based on the RMSE.

We visualize this prediction from the perspective of a patient, meaning that we only show one prediction (consisting of 7 outputs for different time horizons). We make this prediction interactive by directly showing how certain actions affect the predicted blood glucose by altering the input of the model. For example eating an apple instead of a donut (see **figure 22**) or interactively changing the portion size of a meal (see animation at http://daviddemeij.pythonanywhere.com/static/adapting_food.gif).

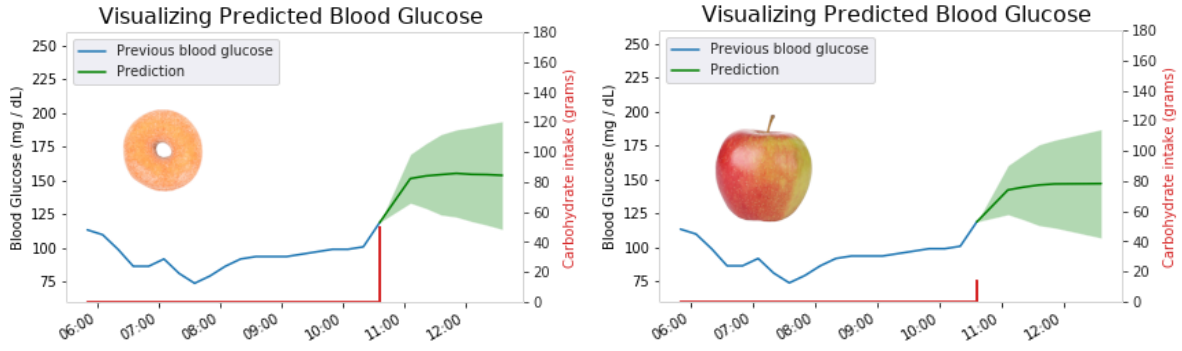

Figure 22. Predicted blood glucose for the upcoming 120 minutes when choosing to eat a donut (left) or when choosing to eat an apple (right) for a randomly selected patient by setting the respective carbohydrate and fat content as input to the model.

6. Discussion

6.1. Evaluation methods

In the patient independent case we use 10 folds of data each containing 54 patients for training and 6 patients for validation. We use 1 fold for choosing hyperparameters and determining the architecture of our model and 9 folds for evaluating each model. An alternative approach might have been to first separate a test fold of 10 patients and then use cross-validation on the other 50 patients for choosing the hyperparameters, making it easier to select the appropriate hyperparameters and features to use. However, this approach would come at the cost of significance, since you can only evaluate the final model on 10 patients instead of on 54 patients across 9 folds in our case.

In the patient dependent case we use the last 100 data points to evaluate the performance of the model for each patient and then average the results. This method resulted in a high standard deviation, giving us no significant results. This could potentially be done more thoroughly by using cross-validation. We did not choose to apply cross-validation, because this would make it impractical to experiment with recurrent neural networks, as these models rely on the entire data sequence (so you can't shuffle them).

It might have been interesting to evaluate certain simpler models before evaluating a complex LSTM. For example, evaluating the performance of a neural network with a sliding window. However, some initial experiments using neural networks with a sliding window did not give us any promising results, making us not inclined to continue in this direction.

6.2. Patient dependent vs. patient independent

In our experiments we find that patient dependent models seem to slightly outperform patient independent models. However, we cannot say this with certainty as by definition the evaluation of the two methods is performed on different test data. Most of the best performing blood glucose prediction models in the literature are patient dependent [7] and thus it is interesting that we can obtain a similar performance with a patient independent model.

A big advantage of the patient independent method is that its performance can improve as we add more patients to our dataset. This is something that won't improve with patient dependent

models unless we have longer observation periods (longer than the current two weeks). Besides the potential for improving the performance when adding additional data, another upside to using a patient independent model is the fact that we can immediately use this model on an unseen patient, without first having to collect data for two weeks. It also gives us insight in how patient characteristics such as age or HbA1c value influence the blood glucose prediction (something that a patient dependent model cannot learn).

An interesting research direction might be to combine patient dependent and independent models in some way to benefit from patient specific dynamics while also taking advantage of population-wide information. This could for example be done by training a SVR model on the first week of data for each patient and then train a LSTM using the output of these SVR models as input data together with other features such as age, hbA1c and actual blood glucose values. A disadvantage of this is that we would cut our training data for the LSTM in half.

6.3. Patient dependent models

In the patient dependent case it seems that Support Vector Regression (SVR) in combination with a compartmental model that models the Rate of appearance (Ra) of exogenous glucose in the blood, has the best performance for longer time horizons. However, we don't get close to the performance as reported by E. Georga et al. [19] and the results are not significantly better than the baseline. This may be due to one or more of the following reasons:

- It seems like E. Georga et al. don't have a separate test and validation set and thus might overfit when selecting their hyperparameters using Differential Evolution.
- E. Georga et al. model blood glucose for type 1 diabetes patients, which may be easier to model using compartmental models as the patients don't produce any insulin themselves anymore and completely rely on exogenous insulin.
- E. Georga et al. use continuous blood glucose sensors with a sample every 5 minutes, instead of every 15 minutes, giving them more data and a higher resolution which might both improve the performance.
- E. Georga et al. selectively remove data from their training and test set when a food intake occurred within the current time and the time that the prediction is made and then they use the Rate of appearance at the time of the prediction. In our opinion this is a flaw in their research method, as we wouldn't be able to do this in a real-world scenario, that is why we use the Rate of appearance at the time of the prediction instead.

- E. Georga et al. had more precise data on administered insulin and they were able to model this.

It is also possible that there is a flaw in our Differential Evolution algorithm. Perhaps a much simpler approach such as exhaustive search would result in a better performance.

Collecting data on administered insulin for the patients in the cohort study and modeling insulin (and perhaps also glucagon and incretin) in the blood using compartmental models might be an interesting direction for future research. It might also be worthwhile to try combining compartmental models with other types of models such as hidden Markov models [40] or long short-term memory (LSTM) networks [26] as this has not been attempted yet. However, our initial experiments with LSTMs were not very promising for the patient dependent case and we expect that there is not enough data available per patient to train a LSTM (as we have maximally two weeks of data for one patient). Perhaps a simpler regular neural network with a sliding window would perform better on this task, as this generally requires less data to train. However, our initial experiments using regular neural networks in the patient dependent case were also not very promising and take a long time to evaluate (because you have to train and evaluate a separate model for each patient).

6.4. Patient independent models

For patient independent models a multitask LSTM has the best performance and is also most practical, as only one network has to be trained to make blood glucose predictions for an entire time period. These predictions can be conveniently used to generate a predicted blood glucose graph to show the patient. An ensemble of multitask LSTMs performed slightly better, but does increases the computation and memory required linearly by the amount of networks that we use in the ensemble (making it for example less suitable to run on a smartphone). In the ensemble model we used the average prediction of 5 separate models to make a prediction. Another method that could be further researched is to use linear regression to weight the predictions or even use another neural network that learns to weight the model outputs based on input features such as age or HbA1c value.

In order to further improve upon the obtained performance, it might be interesting to experiment with the recently introduced attention-based algorithms [46]. It might also be worthwhile to experiment with decorrelating the input features as especially the food intake features are highly correlated, potentially making it harder for the network to learn [45]. Although with enough training data this should not be a problem. A method that could be applied to decorrelate the input

features is using principal component analysis [47]. However, this does complicate the training procedure and may introduce flaws in the model evaluation methodology.

6.5. Usefulness of features

The most beneficial feature for the LSTM seems to be time, which is a feature that basically comes for free (as each blood glucose measurement already has a timestamp) and should thus definitely be included. HbA1c also seem to be a feature helpful, but other patient characteristics such as age and gender don't seem to be helpful in our feature selection experiments. This might be due to the network overfitting on these features during training and then putting too much weight on these features during validation (for example there might be an older women with a more stable blood glucose in the validation set, while the network never encountered this during training).

Food intake data does not significantly improve the performance of the network. However, the network does seem to be sensitive to changes in food intake data in a consistent manner. This means that even though tracking food intake might not give better predictions, it can still help patients get an insight in how changing the amount they eat right now influences the predicted blood glucose, which is an important aspect of our application. The same can be concluded for steps data, which also doesn't significantly improve the performance of the prediction, but could still be used to give patients insight in how their physical activity changes the predicted blood glucose levels.

Currently we have only used nutritional information that is available when exporting data from the "Eetmeter"[21] by the "Voedingscentrum". However, we have also processed most of the data using our own food registration app[22] which uses the NEVO table [5] and provides us with a lot of additional nutritional information. An interesting future research direction is testing if this additional information - such as what kind of glucose the food contains (monosaccharide, disaccharide or polysaccharide) - can benefit the accuracy of our network.

There is only some information on medicine usage available through the health records, if we would have more precise information on insulin dosage including the time when it is administered this might significantly help the prediction model. Another novel feature that could be included in the future is the microbiome composition. This has been shown to be an effective feature in predicting postprandial blood (post-meal) glucose responses in non-diabetics [10] and it also seems to have an important role in the development of diabetes [52].

[21] https://mijn.voedingscentrum.nl/nl/eetmeter/ (Retrieved at 3-9-2018)
[22] http://daviddemeij.pythonanywhere.com/ (Retrieved at 3-9-2018)

6.6. Real world application

It is difficult to determine if the achieved accuracy is good enough to be useful in a real world application, this is probably something that has to be experienced in practice. However, what we can do, is visualize the achieved accuracy by adding a margin that shows the expected error of the model for a prediction (as described in section 5.8). This may be helpful in giving the patient insight in how reliable the prediction actually is, while still showing how the patient's actions affect their predicted blood glucose levels.

Currently we use a margin based on the evaluated RMSE as a way of visualizing the expected error for the patient. However, a more accurate way to show the expected error would be prefered. This could be done by determining the standard error of a prediction using a method such as described in [51] and using this error to determine a 95% confidence interval of the network.

Because hypoglycemia is the most dangerous possibility in the short-term and is often caused by too much insulin medicine, future work can include using insulin as an additional input to the model, attempting to model the effect of a certain insulin dosage. However, before deploying this feature in a real world application, this should be extensively validated as wrongly advising patients on insulin usage could potentially lead to dangerous situations. That's why in our opinion the initial application should probably not give advice on insulin and should also not be used by the patient to adapt their insulin dosage.

7. Conclusion

We have shown that it is possible to predict blood glucose for type 2 diabetes patients with a significantly better accuracy than using an autoregressive model by using a multitask LSTM. As expected the accuracy drops as we try to make predictions for longer time horizons. Patient dependent models seem to outperform patient independent models, although they cannot be directly compared as they are evaluated using a different evaluation method. Patient independent models have our preference and have been the main focus of this research because:

1. Patient independent models can be applied immediately to a new patient (we don't have to train a new model for each patient), making it more practical and convenient.
2. Patient independent models give us more insight in the relationship between input data (such as food intake) and the predicted blood glucose.
3. Patient independent models have more potential to improve as we add more data in the future. Patient dependent models can only gain more data by increasing the (already quite long) two week observation period.

Even though using additional features such as food intake only slightly improve the performance, the network is sensitive to the additional input and uses it in a consistent manner, making it useful for showing patients how changes in this input (for example increasing the carbohydrate intake) influences the predicted blood glucose values.

When an application is developed it is recommended to visualize the expected error of the prediction in such a way that patients are aware of the limitations of the model, while still benefiting from the insight in how their actions influence the predicted blood glucose values.

References

[1] S. Chemlal, S. Colberg, M. Satin-Smith, E. Gyuricsko, T. Hubbard, M. W. Scerbo, and F. D. McKenzie (2011). *Blood glucose individualized prediction for type 2 diabetes using iPhone application*, IEEE 37th Annual Northeast Bioengineering Conference (NEBEC), pp. 1-2, 201.

[2] Kilpatrick ES, Rigby AS, Goode K, Atkin SL. (2007). *Relating mean blood glucose and glucose variability to the risk of multiple episodes of hypoglycaemia in type 1 diabetes*, Diabetologia 50:2553–2561.

[3] B. P. Kovatchev, D. Shields, and M. Breton (2009). *Graphical and numerical evaluation of continuous glucose sensing time lag*, Diabetes Technol. Ther., vol. 11, no. 3, pp. 139–143, Mar. 2009

[4] Andrea Facchinetti (2016). *Continuous Glucose Monitoring Sensors: Past, Present and Future Algorithmic Challenges*, Sensors 16:12, 2093.

[5] NEVO-tabel; Nederlands Voedingsstoffenbestand (2011), RIVM/Voedingscentrum, Den Haag.

[6] Klaus Donsa, Stephan Spat, Peter Beck, Thomas R Pieber, and Andreas Holzinger (2015). *Towards personalization of diabetes therapy using computerized decision support and machine learning: some open problems and challenges*, Smart Health, pp. 237260. Springer.

[7] Zarkogianni K, Litsa E, Mitsis K et al (2015). *A review of emerging technologies for the management of diabetes mellitus*, IEEE Trans Biomed Eng, 62:2735–2749

[8] Benjamin EM (2002). *Self-monitoring of blood glucose: the basics*, Clin Diabetes 2002, 20:45–7.

[9] C. Zecchin, A. Facchinetti, G. Sparacino, and C. Cobelli (2013). *Reduction of number and duration of hypoglycemic events by glucose prediction methods: A proof-of-concept in silico study*, Diabetes Technol. Ther., vol. 15, pp. 66–77, Jan. 2013.

[10] Zeevi D, Korem T, Zmora N, Israeli D, Rothschild D, Weinberger A, et al. (2015). *Personalized nutrition by prediction of glycemic responses*, Cell. 2015;163:1079–94.

[11] J. Li and C. Fernando (2016). *Smartphone-based personalized blood glucose prediction," ICT Express*, vol. 2, no. 4, pp. 150–154, 2016.

[12] David Newman (2007). *UCI machine learning repository*, http://archive.ics.uci.edu/ml/.

[13] S. M. Pappada, B. D. Cameron and P. M. Rosman (2008). *Development of a neural network for prediction of glucose concentration in Type 1 diabetes patients,* J. Diabetes Sci. Technol., vol. 2, pp. 792-801, Sep. 2008.

[14] K. Zarkogianni et al. (2011). *An insulin infusion advisory system based on autotuning nonlinear model-predictive control*, IEEE Trans. Biomed. Eng., vol. 58, no. 9, pp. 2467-2477, May 2011.

[15] G. Baghdadi and A. M. Nasrabadi (2007). *Controlling blood glucose levels in diabetics by neural network predictor*, in Proc. IEEE EMBS, Lion, France, 2007, pp. 3216-3219.

[16] Z. Zainuddin, O. Pauline and C. Ardil (2009). *A neural network approach in predicting the blood glucose level for diabetic patients*, Int. J. Comp. Int., vol. 3, no. 1, pp. 72-79, Feb. 2009.

[17] K. Zarkogianni et al. (2014). *Neuro-Fuzzy based Glucose Prediction Model for Patients with Type 1 Diabetes Mellitus*, in Proc. IEEE-EMBS, Valencia, Spain, 2014, pp. 252-255.

[18] K. Zarkogianni, E. Litsa, A. Vazeou and K. S. Nikita (2013). *Personalized glucose-insulin metabolism model based on self-organizing maps for patients with type 1 diabetes mellitus*, presented at IEEE-BIBE, Chania, Greece, Nov. 10-13, 2013.

[19] E. Georga et al. (2013). *Multivariate prediction of subcutaneous glucose concentration in Type 1 diabetes patients based on support vector regression*, IEEE J. Biomed. Health Inform., vol. 17, pp. 71-81, Feb. 2013.

[20] S. G. Mougiakakou et al. (2008). *Prediction of glucose profile in children with type 1 diabetes mellitus using continuous glucose monitors and insulin pumps*, Horm. Res., pp. 22-23, 2008.

[21] G. Sparacino, et al. (2007). *Glucose Concentration can be Predicted Ahead in Time From Continuous Glucose Monitoring Sensor Time-Series*, IEEE Transactions on Biomedical Engineering, vol. 54, Issue 5, pp. 931 – 937, 2007.

[22] Costello EK, et al. (2009). *Bacterial community variation in human body habitats across space and time*, Science. 2009;326:1694–1697.

[23] Turnbaugh PJ, et al. (2009). *A core gut microbiome in obese and lean twins*, Nature. 2009;457:480–484.

[24] Caporaso JG, et al. (2011). *Moving pictures of the human microbiome*, Genome Biol. 2011;12:R50.

[25] Dethlefsen L, Relman DA. (2011). *Incomplete recovery and individualized responses of the human distal gut microbiota to repeated antibiotic perturbation*, Proc Natl Acad Sci U S A. 2011;108(Suppl 1):4554–4561.

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

[27] Auli, Michael, Galley, Michel, Quirk, Chris, and Zweig, Geoffrey (2013). *Joint language and translation modeling with recurrent neural networks*, Empirical Methods in Natural Language Proessing (EMNPL), volume 3, 2013.

[28] Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc VV (2014). *Sequence to sequence learning with neural networks*, In Advances in Neural Information Processing Systems (NIPS) 27, pp. 3104–3112, 2014.

[29] Karpathy, Andrej and Fei-Fei, Li (2015). *Deep visual-semantic alignments for generating image descriptions*, In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3128–3137, June 2015.

[30] Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru (2015). *Show and tell: A neural image caption generator*, In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3156–3164, June 2015.

[31] Liwicki, Marcus, Graves, Alex, Bunke, Horst, and Schmidhuber, Jurgen (2007), *A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks*, In Proceedings of the Ninth International Conference on Document Analysis and Recognition, volume 1, pp. 367–371, 2007.

[32] Graves, Alex, Liwicki, Marcus, Fernandez, Santiago, Bertolami, Roman, Bunke, Horst, and Schmidhuber, Jurgen (2009). *A novel connectionist system for unconstrained handwriting recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(5):855–868, 2009.

[33] Pollastri, Gianluca, Przybylski, Darisz, Rost, Burkhard, and Baldi, Pierre. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. Proteins: Structure, Function, and Bioinformatics, 47(2):228–235, 2002.

[34] Lipton, Zachary C, Kale, David C, and Wetzel, Randall (2016). *Modeling missing data in clinical time series with rnns*, Machine Learning for Healthcare, 2016.

[35] Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwi´nska, A., Hassabis, D. (2016). *Hybrid computing using a neural network with dynamic external memory*, Nature.

[36] Stratton IM, Adler AI, Neil HA, et al. Association of glycaemia with macrovascular and microvascular complications of type 2 diabetes (UKPDS 35): prospective observational study. BMJ 2000;321:405-12.

[37] N. Gul (2010). *Knowledge, attitudes and practices of type 2 diabetic patients*, Journal of Ayub Medical College, Abbottabad, vol. 22, no. 3, pp. 128–131, 2010

[38] *Diabetic Hypoglycemia*, https://www.mayoclinic.org/diseases-conditions/diabetic-hypoglycemia/symptoms-causes/syc-20371525, retrieved at 24-8-2018

[39] Vahidi O, Kwok KE, Gopaluni RB, Knop FK. (2015) *A comprehensive compartmental model of blood glucose regulation for healthy and type 2 diabetic subjects*, Med Biol Eng Comput. 2015; p. 1–16.

[40] L. R. Rabiner and B. H. Juang. (1986) *An introduction to hidden Markov models*, IEEE Acoust., Speech, Signal Processing Mag., pp. 4–16, Jan. 1986.

[41] *Hyperglycemia in Diabetes*, https://www.mayoclinic.org/diseases-conditions/hyperglycemia/symptoms-causes/syc-20373631, retrieved at 16-9-2018

[42] Cunningham, K., & Read, N. (1989). *The effect of incorporating fat into different components of a meal on gastric emptying and postprandial blood glucose and insulin responses*. British Journal of Nutrition, 61(2), 285-290. doi:10.1079/BJN19890116

[43] W. S. McCulloch and W. Pitts (1943). *A logical calculus of ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics, vol. 5, pp. 115–133, 1943.

[44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams (1986). *Learning internal representations by error propagation*, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, D. E. Rumelhart and James L. McClelland, Eds., vol. 1, ch. 8, pp. 318-362. Cambridge, MA: MIT Press, 1986.

[45] LeCun, Y. (1988). *A theoretical framework for back-propagation*, Proceedings of the 1988 Connectionist Models Summer School, pages 21–28, CMU, Pittsburgh, Pa. Morgan Kaufmann

[46] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y (2015). *Show, attend and tell: Neural image caption generation with visual attention*, CoRR, abs/1502.03044.

[47] Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan (2014). *Dropout: A simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research, 15:1929–1958, 2014

[48] Sud harsan B, Peeples M, Shom ali M (2015). *Hypoglycemia prediction using machine learning models for patients with type 2 diabetes*, J Diabetes Sci Technol an2015;9(1):86–90.http://dx.doi.org/10.1177/1932296814554260

[49] Fisher, R. A. (1915). *Frequency distribution of the values of the correlation coefficient in samples of an indefinitely large population*, Biometrika. Biometrika Trust. 10 (4): 507–521. doi:10.2307/2331838

[50] Gani, A.; Gribok, A. V.; Lu, Y.; Ward, W. K.; Vigersky, R. A. & Reifman, J. (2010). *Universal Glucose Models for Predicting Subcutaneous Glucose Concentration in Humans*, IEEE Transactions on Information Technology in Biomedicine, Vol. 14, No. 1, (January 2010) 157-165, 1089-7771

[51] Zhu, L., Laptev, N. (2017). *Deep and Confident Prediction for Time Series at Uber*. ArXiv 170901907 Stat

[52] Hartstra, A. V., Bouter, K. E., Bäckhed, F., and Nieuwdorp, M. (2015). *Insights into the role of the microbiome in obesity and type 2 diabetes*, Diabetes Care 38, 159–165. doi: 10.2337/dc14-0769

Appendix

Appendix A. Modelling the Rate of Appearance

We first model a carbohydrate intake class as follows:

```
class carb_intake:
    def __init__(self, carbs_mmol):
        self.initial_carbs = carbs_mmol
        self.available_carbs = carbs_mmol
        self.Ra = 0.0
        if self.initial_carbs <= 60:
            self.small_intake = True
        else:
            self.small_intake = False

    def update_rate(self):
        if self.small_intake:
            if self.available_carbs > 0.5 * self.initial_carbs:
                self.Ra += 4.0 / 60
            elif self.available_carbs < 0.5 * self.initial_carbs:
                self.Ra -= 4.0 / 60
        else:
            if self.available_carbs > 30:
                self.Ra += 4.0 / 60
            elif self.available_carbs < 30:
                self.Ra -= 4.0 / 60

        # Keep Ra within boundaries
        self.Ra = max(0.0, self.Ra)
        self.Ra = min(120.0, self.Ra)

    def get_carbs(self, carbs):
        if carbs <= self.available_carbs:
            self.available_carbs -= carbs
            return carbs
        else:
            available = self.available_carbs
            self.available_carbs = 0
            return available
```

We then create a carbohydrate processor class using the previously created carbohydrate intake class.

```
class carb_processor:
```

```
    def __init__(self, time_interval = 1):
        self.time_interval = time_interval
        self.max_Ra = 120
        self.carb_intakes = []

    def add_carb_intake(self, carbs_mmol):
        self.carb_intakes.append(carb_intake(carbs_mmol))
        keep_idx = []
        for i in range(len(self.carb_intakes)):
            if self.carb_intakes[i].available_carbs > 0:
                keep_idx.append(i)
        self.carb_intakes = [self.carb_intakes[idx] for idx in keep_idx]

    def update_rates(self):
        for carb_intake_obj in self.carb_intakes:
            carb_intake_obj.update_rate()

    def get_Ra(self):
        Ra = 0.0
        for carb_intake_obj in self.carb_intakes:
            Ra_obj = carb_intake_obj.Ra
            if Ra + Ra_obj <= self.max_Ra:
                carb_intake_obj.get_carbs(Ra_obj / 3600)
                Ra += Ra_obj
            else:
                remaining_carbs = self.max_Ra - Ra
                carb_intake_obj.get_carbs(remaining_carbs / 3600)
                Ra = self.max_Ra
        return Ra
```

Now we can process the rate of appearance for a patient by going through the patient's data from start to end:

```
carb_processor_obj = carb_processor()
for s in range(start, end):
    if carb_intake[s] > 0:
        carb_processor_obj.add_carb_intake(carb_intake[s])
    carb_processor_obj.update_rates()
    Ra[s] = carb_processor_obj.get_Ra()
```

Appendix B. Preprocessing Data

To preprocess the data we first have to load the health records and food intake file (including all processed data of food intakes that are processed through https://daviddemeij.pythonanywhere.com/).

```
from dateutil import parser
import numpy as np
parse = lambda x: parser.parse(x)
import pandas
import datetime
import os.path

patients = ['1001', '596', '604', '609', '614', '619', '624', '629', '634',
'639', '644', '649', '1002', '597', '605', '610','615', '620', '625',
'630', '635', '640', '645', '650','572', '598', '606', '611', '616', '621',
'626', '631', '636', '641', '646', '651', '574', '600', '607', '612',
'617', '622', '627', '632', '637', '642', '647', '652', '595', '601',
'608', '613', '618', '623', '628', '633', '638', '643', '648', '653']


health_records = pandas.read_csv('DIALECT 23-02-2018.csv', sep=';')
health_records = health_records[health_records['Subjectnr'].isin(patients)]
health_records_features = health_records[['Subjectnr', 'Geslacht',
'Leeftijd_poli1', 'Jaren_DM2', 'Gewicht_poli1', 'SerumHbA1c_1', 'dosA10AB',
'dosA10AC', 'dosA10AD', 'dosA10BA']]

# Import food intake that was processed through the food tool
food = pandas.DataFrame(pandas.read_csv('all_food_records.csv', sep='\t',
parse_dates=['datetime'])).fillna(0.0)
# We have to calculate salt based on the value for natrium (this is how
voedingscentrum also calculates salt because this is not given in the NEVO
table)
food['salt'] = (food['field_09006']/1000.0)/0.4
# List of patients that have their food intake logs processed through
daviddemeij.pythonanywhere.com
patients_tool = list(food.patient_id.unique()[:])

headers = ["datetime", "glucose", "seconds_elapsed", "hour_of_day",
"missing_hr", "hr", "missing_steps", "steps", 'missing_food',
'Energie (kcal)', 'Vet (g)', 'Verz. vet (g)', 'Koolhydr (g)', 'Eiwit (g)',
'Vezels (g)', 'Zout (g)', 'Alcohol (g)', 'Water (g)', 'Natrium (mg)',
```

```
'Kalium (mg)', 'Calcium (mg)', 'Magnesium (mg)', 'IJzer (mg)', 'Selenium
(µg)', 'Zink (mg)', 'Vit. A (µg)', 'Vit. D (µg)', 'Vit. E (mg)', 'Vit. B1
(mg)', 'Vit. B2 (mg)', 'Vit. B6 (mg)', 'Foliumzuur (µg)', 'Vit. B12 (µg)',
'Nicotinezuur (mg)', 'Vit. C (mg)', 'Jodium (µg)',
'Geslacht', 'Leeftijd_poli1', 'Jaren_DM2', 'BMI', 'HbA1c',
'dosA10AB', 'dosA10AC', 'dosA10AD', 'dosA10BA']
```

We then process each patient individually and store in a separate CSV file.

```
for patient in patients:
    print(patient)
    patient_dir = os.curdir + "/data/" + patient + "/"
    heart = os.path.isfile(patient_dir + patient + "-heart.xlsx")
    food_manual = os.path.isfile(patient_dir + patient + "-voeding.xlsx")

    # Preprocess glucose data
    table = pandas.read_table(patient_dir + patient + "-glucose.txt")
    table = table.where(getattr(table, "Type vastlegging") == 0)
    table = table[pandas.notnull(table.ID)]
    table = table[['Tijd', 'Historie glucose (mmol/L)']]
    times_parsed = np.array([parse(time) for time in
table['Tijd'].values.tolist()])
    glucose_values = np.concatenate(
        (times_parsed.reshape(-1, 1), table['Historie glucose
(mmol/L)'].values.reshape(-1, 1)), axis=1)

    # preprocess HR data
    if heart:
        hr_table = pandas.read_excel(patient_dir + patient + "-heart.xlsx")
        datetimes, hr_values = [], []
        for date in hr_table.columns:
            if date != "time":
                data = hr_table[['time', date]].values
                for i in range(data.shape[0]):
                    datetimes.append(parse(date + " " + data[i, 0]))
                    hr_values.append(data[i, 1])
        hr_data = np.concatenate((np.array(datetimes).reshape(-1, 1),
np.array(hr_values).reshape(-1, 1)), axis=1)

    # Preprocess STEPS DATA
    steps_table = pandas.read_excel(patient_dir + patient + "-steps.xlsx")
    datetimes = []
    steps_values = []
    for date in steps_table.columns:
        if date != "time":
```

```
            data = steps_table[['time', date]].values
            for i in range(data.shape[0]):
                datetimes.append(parse(date + " " + data[i, 0]))
                steps_values.append(data[i, 1])
        steps_data = np.concatenate((np.array(datetimes).reshape(-1, 1),
np.array(steps_values).reshape(-1, 1)), axis=1)

    # Preprocess FOOD INTAKE DATA
    if food_manual:
        food_table = pandas.read_excel(patient_dir + patient +
"-voeding.xlsx", header=None)
        df = pandas.DataFrame(food_table)
        df = df[df.iloc[:, 0].notna()]
        df = df.fillna(0)
        food_intake_data = df.values[:, [0] + list(range(5, 32))]
    else:
        food_records_filtered = food[(food.patient_id == int(patient)) &
(food.missing_time == False)].fillna(0.0)

    # Preprocess health records data
    health_records_data = list(
        health_records_features[
            health_records_features['Subjectnr'] == patient
            ].replace(' ', '0').values[0, 1:].astype(float))
    bmi = float(health_records_data[4]) / ((float(health_records_data[3]) /
100.0) ** 2)
    health_records_data = np.array([health_records_data[:3] + [bmi] +
health_records_data[5:]])

    # Combine data
    prev_date = glucose_values[0][0] - datetime.timedelta(minutes=20)
    data = np.array([headers])
    for row in glucose_values:

        steps = steps_data[(steps_data[:, 0] < row[0]) & (steps_data[:, 0] >
prev_date) & (steps_data[:, 1] != -1), 1]
        if food_manual:
            food_intake = food_intake_data[(food_intake_data[:, 0] < row[0])
& (food_intake_data[:, 0] > prev_date), 1:]
            sum_food_intake = np.array([np.sum(food_intake,
axis=0).astype(float)])
        else:
            # The fields refer to the NEVO table fields corresponding to the
nutritional values that we use
            food_records_values =
food_records_filtered[(food_records_filtered.datetime <= row[0]) &

(food_records_filtered.datetime > prev_date)][[
```

```
                'field_01001', 'field_03001', 'field_03004',
                'field_05001', 'field_02002', 'field_06001', 'salt',
'field_08001',
                'field_07001', 'field_09006', 'field_09007', 'field_09001',
                'field_09008', 'field_09003', 'field_10001', 'field_09009',
                'field_11002', 'field_11009', 'field_11010', 'field_11005',
                'field_11006', 'field_11007', 'field_11013', 'field_11008',
                'field_11014', 'field_11011', 'field_10003']].values
            sum_food_intake = np.zeros((1, 27)) +
np.sum(food_records_values, axis=0)

        missing_steps = 0
        if heart:
            hr = hr_data[(hr_data[:, 0] < row[0]) & (hr_data[:, 0] >
prev_date) & (hr_data[:, 1] != -1), 1]
            if len(hr) > 0:
                avg_hr = np.sum(hr) / float(len(hr))
                missing_steps = 0
            else:
                avg_hr = 0
                missing_steps = 1
        else:
            avg_hr = 0

        if len(steps) > 0:
            sum_steps = np.sum(steps)
            if sum_steps > 0:
                missing_steps = 0
        else:
            sum_steps = 0
            missing_steps = 1

        # Store current date for next iteration
        if prev_date != datetime.datetime(2000, 1, 1, 0, 0):
            seconds = (row[0] - prev_date).seconds
        else:
            seconds = 0

        prev_date = row[0]
        print(row[0], (row[0] - datetime.datetime(1970, 1,
1)).total_seconds())
        data_row = np.array([[(row[0] - datetime.datetime(1970, 1,
1)).total_seconds(), float(row[1].replace(",", ".")),
                              seconds, row[0].hour, int(not heart), avg_hr,
                              missing_steps, sum_steps, int(not food_manual
and not (int(patient) in patients_tool))]])

        data_row = np.concatenate((data_row, sum_food_intake), axis=1)
```

```
            data_row = np.concatenate((data_row, health_records_data), axis=1)
            data = np.concatenate((data, data_row), axis=0)
        print(data.shape)

        df = pandas.DataFrame(data[1:, :], columns=data[0, :])
        df.to_csv("/test_data_processing/" + patient + "-processed.csv")
```

Finally we combine these separate files in one large matrix using the following code:

```
# Create matrix that includes all data
datasets = []
nr_missing_food, nr_food = 0, 0
max_length = 0
patient_ids = []
for filename in os.listdir(os.curdir + "./processed_data_per_patient"):
    # if np.genfromtxt(os.curdir + "./processed_data_per_patient/" +
filename, delimiter=',')[1,9] == 0:
    data = np.genfromtxt(os.curdir + "./processed_data_per_patient/" +
filename, delimiter=',')[1:, :]
    data[:, 2] = data[:, 2] * 18
    patient_id = np.ones((data.shape[0], 1)) * int(filename.split("-")[0])
    patient_ids.append(patient_id)
    datasets.append(data)
    if data.shape[0] > max_length:
        max_length = data.shape[0]
    if int(data[0, 9]) == 1:
        nr_missing_food += 1
    else:
        nr_food += 1
    # print(int(filename.split("-")[0]))
print("# food", nr_food)
print("# missing food", nr_missing_food)
# x_train.append(np.genfromtxt(os.curdir+"/all_patients-processed.csv",
delimiter=',')[1:,3])
all_data = np.zeros((len(datasets), max_length, datasets[0].shape[-1]))
for i in range(len(datasets)):
    patient_id = patient_ids[i]
    all_data[i, (max_length - datasets[i].shape[0]):, :] =
np.concatenate((patient_id, datasets[i][:, 1:]), axis=1)

np.save("all_data_zeros_first", all_data)
```

Appendix C. Multitask LSTM network

The code below shows the architecture and hyperparameters of the best performing multitask LSTM network.

```
import tensorflow as tf
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense, LSTM, Dropout, Activation
from tensorflow.python.keras.optimizers import RMSprop, Adam
from tensorflow.python.keras.callbacks import EarlyStopping, TensorBoard
import random
import os
import numpy as np

dataset = np.load('all_data_zeros_first.npy')

# Experiment Parameters
lr = 0.01
dropout = 0.3
hidden_state = 85
# [glucose, time of day (0-24), steps, saturated fat, carbohydrates,
#  salt, age, HbA1c)
features = [2, 4, 8, 12, 13, 16, 38, 41]
nr_features = len(features)

ix_test = [44, 34, 2, 1, 27, 36]
ix_train = [i for i in range(60) if (i not in ix_test)]

# Dataset preparation
if len(features) == 1:
    x_data = dataset[:, :-10, :][:, :, 2:3]
else:
    x_data = dataset[:, :-10, :][:, :, features]
y_data = np.concatenate((dataset[:, 8:-2, 2:3], dataset[:, 7:-3, 2:3],
dataset[:, 6:-4, 2:3], dataset[:, 5:-5, 2:3], dataset[:, 4:-6, 2:3],
dataset[:, 3:-7, 2:3], dataset[:, 2:-8, 2:3]), axis=2)

x_train = x_data[ix_train, :, :]
x_test = x_data[ix_test, ::, :]

y_train = y_data[ix_train, :, :]
```

```
y_test = y_data[ix_test, :, :]

num_x_signals = x_train.shape[2]
num_y_signals = y_train.shape[2]

# Scaling data to [0, 1]
x_min_scaler = np.min(x_train, axis=(0, 1))
x_max_scaler = np.max(x_train, axis=(0, 1))

x_train_scaled = np.zeros_like(x_train)
for i in range(x_train.shape[2]):
    x_train_scaled[:, :, i] = (x_train[:, :, i] - x_min_scaler[i]) /
(x_max_scaler[i] - x_min_scaler[i])

y_min_scaler = np.min(y_train, axis=(0, 1))
y_max_scaler = np.max(y_train, axis=(0, 1))

y_train_scaled = (y_train - y_min_scaler) / (y_max_scaler - y_min_scaler)

y_test_scaled = (y_test - y_min_scaler) / (y_max_scaler - y_min_scaler)

x_test_scaled = np.zeros_like(x_test)
   for i in range(x_test.shape[2]):
       x_test_scaled[:, :, i] = (x_test[:, :, i] - x_min_scaler[i]) /
(x_max_scaler[i] - x_min_scaler[i])

validation_data = (x_test_scaled, y_test_scaled)

model = Sequential()

model.add(LSTM(hidden_state, return_sequences=True, stateful=False,
input_shape=(x_train.shape[1], num_x_signals)))
model.add(Dropout(dropout))

model.add(LSTM(hidden_state, return_sequences=True,
input_shape=(x_train.shape[1], hidden_state), stateful=False))

model.add(Dropout(dropout))

model.add(Dense(num_y_signals))


def rmse_120(y_true, y_pred):
```

```
    y_true_120 = y_true[:, :, 0:1]
    y_pred_120 = y_pred[:, :, 0:1]
    return rmse(y_true_120, y_pred_120)

def rmse_90(y_true, y_pred):
    y_true_90 = y_true[:, :, 2:3]
    y_pred_90 = y_pred[:, :, 2:3]
    return rmse(y_true_90, y_pred_90)

def rmse_60(y_true, y_pred):
    y_true_60 = y_true[:, :, 4:5]
    y_pred_60 = y_pred[:, :, 4:5]
    return rmse(y_true_60, y_pred_60)

def rmse_30(y_true, y_pred):
    y_true_30 = y_true[:, :, 6:7]
    y_pred_30 = y_pred[:, :, 6:7]
    return rmse(y_true_30, y_pred_30)

def rmse(y_true, y_pred):
    y_true_unscaled = tf.add(tf.scalar_mul(y_max_scaler[0] -
y_min_scaler[0], y_true), y_min_scaler[0])
    y_pred_unscaled = tf.add(tf.scalar_mul(y_max_scaler[0] -
y_min_scaler[0], y_pred), y_min_scaler[0])
    y_pred_unscaled = tf.where(tf.less(y_true_unscaled, 0.01),
tf.zeros_like(y_true_unscaled), y_pred_unscaled)

    diff =
tf.reduce_sum(tf.square(tf.keras.backend.flatten(y_true_unscaled) -
tf.keras.backend.flatten(y_pred_unscaled)))
    loss = tf.sqrt(diff / tf.cast(tf.count_nonzero(y_pred_unscaled),
dtype=tf.float32))
    return loss


def loss(y_true, y_pred):
    y_true_unscaled = tf.add(tf.scalar_mul(y_max_scaler[0] -
y_min_scaler[0], y_true), y_min_scaler[0])
    y_pred = tf.where(tf.less(y_true_unscaled, 0.01),
tf.zeros_like(y_pred), y_pred)
    loss = tf.reduce_sum(tf.square(tf.keras.backend.flatten(y_true) -
tf.keras.backend.flatten(y_pred)) /
                         tf.cast(tf.count_nonzero(y_pred),
```

```
dtype=tf.float32))
    return loss


optimizer = Adam(lr=lr)
model.compile(loss=loss, optimizer=optimizer, metrics=[rmse, rmse_30,
rmse_60, rmse_90, rmse_120])
model.summary()

early_stopping = EarlyStopping(monitor='val_loss', min_delta=0,
patience=10, mode='auto')
callback_tensorboard = TensorBoard(log_dir='./log/')
callbacks = [early_stopping, callback_tensorboard]

x_batch = x_train_scaled
y_batch = y_train_scaled

def batch_generator():
    """
    Generator function for creating random batches of training-data.
    """
    while True:
        idx = np.random.permutation(x_train_scaled.shape[0])
        x_batch = x_train_scaled[idx]
        y_batch = y_train_scaled[idx]
        yield (x_batch, y_batch)


num_epochs = 200
model.fit_generator(generator=batch_generator(),
                                epochs=num_epochs,
                                steps_per_epoch=10,
                                validation_data=validation_data,
                                verbose=True, callbacks=callbacks)
```