Bern University
of Applied Sciences

UNIVERSITÄT
BERN

Master of Science in Biomedical Engineering

Data Driven Diabetes Management

Project 3: Blood glucose prediction with deep learning

Matthias Pracht, Patrick von Raumer, Matteo Tagliabue & Yves Moerlen

December 2023

Abstract

Diabetes, a common chronic metabolic disorder, is associated with elevated blood glucose levels. Current treatments rely on insulin injections, which has led to the development of automated insulin delivery (AID) systems. Challenges remain in optimizing insulin dosing, considering factors such as food intake and physical activity. Using the Ohio dataset, this report explores machine learning algorithms to improve the accuracy of predicting blood glucose levels. The goal is to advance AID systems by reducing the risk of hyper- or hypoglycaemia caused by incorrect insulin doses.
Data preprocessing involved addressing missing values and regularizing the dataset. The selection of key parameters for model development was informed by correlation analysis. The use of one of two neuronal network models was decided and the determination of optimal parameters (k = 72, PH = 4) were achieved through systematic evaluation. The final model was assessed using two methods resulting in an average mean squared error of 1.23% with one and 3.16% with the other method across all patients.

Contents

Introduction

Diabetes is a chronic metabolic disease characterized by chronically elevated blood glucose because the body cannot produce or use insulin. Diabetes is categorized as Type 1 (T1D), when Insulin is not produced in sufficient amounts, or Type 2 Diabetes (T2D, when the body becomes resistant to insulin. Both types result in unstable blood sugar levels known as either hyperglycemia or hypoglycemia. Hyper- or hypoglycemia can induce a range of complications, from acute to chronic, leading to 6.7 million deaths attributed to diabetes (International Diabetes Federation, 2021) worldwide per year.

Diabetes is not only a highly prevalent disease, with over 500 million people affected worldwide, but also without any effective treatment method available. Currently the main treatment for diabetes is regular insulin injections, where the estimated dose heavily depends on the physician's and patient's experience. Thus, automated insulin delivery (AID) systems have been developed, but they still require user input for optimal control. In such almost closed-loop systems, the sensory as well as the correcting part, are important for optimal control.

Considering the amount of currently available efficient blood sugar detection methods there are still some problems for optimal control of the insulin dose, as some factors, for example digested food, and activity, are not considered. However, the combination of these parameters with the measured blood glucose level can be used as an input for a machine learning algorithm, which then can be implemented in the closed loop system to predict the future blood glucose levels. Thus, the regulation system can be further improved, which could reduce the risk of possible hyper- or hypoglycaemia induced through the wrong dose of insulin.

In this project, different types of machine learning algorithms have been built, trained, and tested with the Ohio data set [1] to determine the accuracy of predicted blood glucose levels.

Related work

The topic under analysis has already been addressed in these three documents, which deal with the same issue.

- *"A Deep Learning Algorithm For Personalized Blood Glucose Prediction"* [2]
- *"An autonomous channel deep learning framework for blood glucose prediction"* [3]
- "Blood Glucose Prediction in Type 1 Diabetes Using Deep Learning on the Edge"[4]

All three reports are based on predicting glucose levels in patients with type 1 diabetes using machine learning-based approach. What distinguishes the papers is the method used to achieve the goal.

The first report is based on a CNN (convolutional neural network) model based on WaveNet [5]. It transforms the problem into a classification, using CNN dilated layers and fast WaveNet algorithms to predict future glucose levels. Finally, the model is evaluated on the OhioT1DM dataset.

The second paper is based on an autonomous deep learning-based framework with a channel network for learning. The framework uses an autonomous channel network, learning representations from input variables with reasonable sampling periods and sequence lengths based on time-domain knowledge. Finally, the model is evaluated on the OhioT1DM clinical dataset.

The third study employs a learning framework using a microcontroller. It uses a long-term memory based RNN (LSTM). It is evaluated on a clinical dataset obtained from 12 subjects with T1D.

This work is situated among those mentioned above, allowing the prediction of blood glucose in the patient with diabetes of type 1 using deep learning but maintaining a simple structure.

Material and Methods

Material

Ohio Dataset

The "OhioT1DM Dataset" is a dataset used for research in the field of type 1 diabetes (T1D) management. This dataset contains detailed information on patients with type 1 diabetes, including their blood glucose levels, continuous glucose readings, amount of insulin administered, carbohydrate intake, and other relevant parameters.

Information in the dataset is collected at regular intervals over time, often every 5 minutes, to provide a detailed view of blood glucose dynamics. The data include readings from continuous glucose sensors, glucose values obtained from measurements using test strips or other monitoring devices.[6]

During the period from 2018 to 2020, data were collected from six patients in each year. Each patient contributed two sets of data, obtained at different time periods during an interval of eight weeks each, allowing each subject to be used for both training and testing of customized models.

The CSV file is organized with the following columns:

- *5minute_intervals_timestamp:* Time recorded at five-minute intervals.
- *missing_cbg:* Absence of Continuous Blood Glucose data.
- *cbg:* Continuous Blood Glucose levels.
- *finger:* Blood glucose readings from finger-stick measurements.
- *basal:* Basal insulin rate, indicating the dosage of long-acting insulin.
- *hr:* Heart rate.
- *gsr:* Galvanic Skin Response, a measure of skin conductance.
- *carbInput:* Estimated carbohydrate intake.
- *bolus:* Insulin bolus injection, representing the administration of short acting insulin.

All patients, of the used Ohio data sets 2018 and 2020, have received a further simplified identification as listed in Table 1.

Table 1: Ohio Dataset as Patient ID's. Each dataset has an individual training and testing data set.

Dataset	Ohio 2018						Ohio 2020					
Patient Nr:	0	1	2	3	4	5	6	7	8	9	10	11
Dataset ID:	559	563	570	575	588	591	540	544	552	567	584	596

Methods

The libraries and frameworks TensorFlow and Keras were utilized to develop the Recurrent Neural Network (RNN) and Self-Attention Network (SAN). NumPy, SciPy, MatPlotLib, Pandas, and built-in modules from Python were used for everything else. These models were trained using a supervised learning approach, where the model is provided with the data. The models were trained on the Training data set and tested on the Test data set.

Libraries and Frameworks

TensorFlow (version 2.13.0) is an open-source machine learning framework, which we used in combination with Keras (version 2.13.0) to build and train the models.
Numpy (version 1.20.3), Pandas (version 1.3.4), and SciPy (version 1.10.1) were used to manipulate and transform the data, perform various mathematical operations and analyses. Matplotlib (version 3.7.4) has been used to create graphs and visualize results.
The whole environment has been built with Python (version 3.8.18) using conda (version 4.10.3) on the University of Bern Linux Cluster (UBELIX).

Pipeline structure

The ground structure of the code is structured as follows:

Global parameters: Definition of global parameters to control visualization, model training and evaluation.

Loading data: loading data in CSV format, organized by year, divided into training and test sets.

Data preparation: The *prepareData*-function interpolates missing data and normalizes the specified columns. Optional visualization shows graphs of original and interpolated data for each patient.

Correlation matrix: The *correlationMatrix*-function calculates the correlation between variables, highlighting the correlation with the expected glucose level in the future.

Creation of sequences to train the model: The *sequences*-function creates input (X) and output (Y) sequences from patient data, using specified time windows and prediction horizons.

Neural model construction: the **SAN** and **RNN** get constructed, and the summaries get displayed.

Model training: The code trains a neural network using an optimizer, the loss "*mean_squared_error*". Through a *scheduler*, the learning rate adjusts during model training.

Model Evaluation: If enabled, the code loads the best saved model and evaluates performance, generating graphs for loss, learning rate, and comparing predicted results with actual results.

Saving Results: Saves results and graphs in a directory specific to the current experiment.

Data preprocessing

The Ohio data set has been preprocessed and normalized before using it for the training and testing of the models. The preprocessing consisted of replacing every “NaN” entry by zero and interpolating missing cbg values using the Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) 1-D monotonic cubic interpolation. Finally, the inverse of the highest value of each data type of the whole Ohio data set has been used as a normalization factor for each individual type of data, except for "*5minute_intervals_timestamp*" and "*missing_cbg*", thus resulting in data values in a range of [0,1] for each data type as seen in Figure 1 and Table 2. Hence, the retransformation of the predicted normalized cbg value was possible using this global normalization factor of cbg.

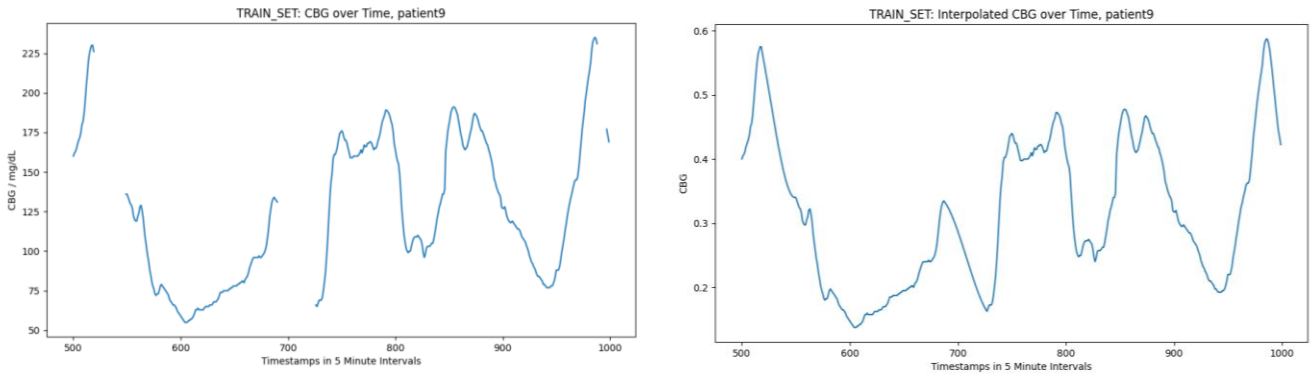

Figure 1: Example for a pre-processed and normalized part of the cbg data of patient 9. A more detailed overview for each patient can be found in the image folder "img" on our GitHub.

Table 2: Example, extracted from patient 2 with index from 78 to 80, of raw and prepared data.

Raw data								
5minute intervals timestamp	**missing cbg**	**cbg**	**finger**	**basal**	**hr**	**gsr**	**carbInput**	**bolus**
5474670.84336	0.0	294.0	NaN	1.35	94.0	0.00006	NaN	NaN
5474671.84392	0.0	296.0	NaN	1.35	100.0	0.00005	NaN	NaN
5474671.84448	1.0	NaN	359.0	1.35	NaN	NaN	NaN	NaN

Prepared data								
5minute intervals timestamp	**missing cbg**	**cbg**	**finger**	**basal**	**hr**	**gsr**	**carbInput**	**bolus**
5474670.84336	0.0	0.73500	0.00000	0.57692	0.49735	0.00000	0.00000	0.00000
5474671.84392	0.0	0.74000	0.00000	0.57692	0.53439	0.00000	0.00000	0.00000
5474671.84448	1.0	0.76253	0.61263	0.57692	0.00000	0.00000	0.00000	0.00000

Parameter selection and correlation matrix

The data types cbg, basal, carbInput, and bolus have been used as input parameters for the prediction of the future cbg value according to the project guidelines. The correlation between all parameters and the current or future cbg values have been computed to quantify their relationship for each patient.

Sequencing and Training Modes

Finally, the training data is cut into overlapping segments of a defined window length "k" for each chosen parameter. Thus, after cutting a segment, the window is shifted by one index and the next segment is cut and represents the past k values, which will be used to predict the future cbg value.
For this corresponding future cbg ground-truth, the indices of the segment are shifted by a defined prediction horizon "PH", which defines the temporal difference between the current and the predicted value. Thus, a data set of e.g., length = 100, with k = 10 and PH = 2, yields 88 input parameter segments and ground-truth segments pairs.

Two different modes have been implemented: "Separately" and "Generalized". The training data set and testing data set of "Separately" trained models consist of only of one individual patient and thus, creates a personalized trained model. On the other hand, the "Generalized" mode trained a model using the training data sets from all patients except for a defined patient, whose testing data set will be used for testing the "Generalized" model, which will be representing a "one fits all" method.

Architecture

The architecture of the proposed models is designed to effectively capture complex patterns and dependencies within sequential data.

RNN

The Recurrent Neural Network (RNN) architecture is structured with three Long Short-Term Memory (LSTM) layers, each configured to process input sequences of length "k". The progressive increase in the number of units 64 to 256 allows the RNN to capture smaller dependencies. Incorporated dropout layers contribute to the prevention of overfitting. Two Dense layers are finishing to fit the data first into the positive range using a Rectified Linear Unit (ReLU) and afterwards into the range between zero and one using a Sigmoid function. These layers sum up to a total of 527'233 trainable parameters for this model.

SAN

The Self-Attention Network (SAN) employs a mechanism by integrating multi-head attention layers. The model begins by processing the input sequences of length "k" through a multi-head attention mechanism, allowing it to dynamically weigh the importance of different elements in the sequence. This process is iterated, with an additional normalization layer. A pooling layer condenses the sequence followed by a dropout to prevent overfitting and 2 Dense layers. These fit the data first into the positive range using ReLU and afterwords between zero and one using a Sigmoid Function. These layers sum up to a total of 15'001 trainable parameters for this model.

Optimization

For the optimization, the algorithm Adaptive Moment Estimation (Adam) was chosen. Adam optimization is a stochastic gradient descent algorithm that uses adaptive learning rates and momentum to optimize the model parameters. It is often used as the default optimizer in neural network libraries such as Keras and TensorFlow.

Learning Rate

The process of training a machine learning model involves setting a learning rate, which determines how much the model adjusts its parameters during each epoch. However, it is often difficult to find the optimal learning rate. In this case, a constant learning rate was determined to be ineffective, as such a variable learning rate was implemented. The learning rate was set to be initially high and then exponentially decrease as visible in Figure 2.

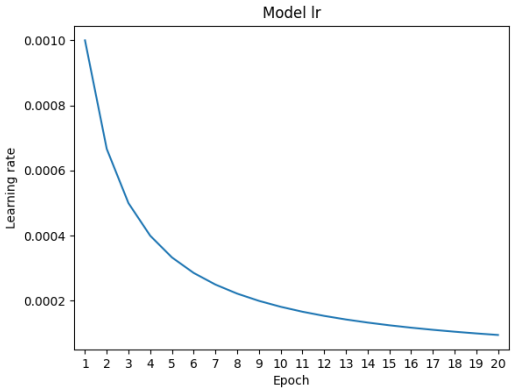

Figure 2: The used learning rate across the epochs

Evaluation

The evaluation of the models and the chosen parameters is conducted through a visual comparison a quantitative metric. The visual comparison involves an examination of graphs depicting the predicted and original data sequences superimposed. Noticeable deviations and the overall alignment are essential in this visual assessment. In parallel, the quantitative metric to measure the average differences between the predicted and actual values is the mean squared error (MSE). A lower MSE indicates better accuracy and precision in predicting the sequential data. This metric is particularly valuable in quantifying the overall performance of the models. The comparisons to be made are:

- RNN vs SAN
- Window evaluation
- Prediction horizon evaluation
- Separately vs Generalized

Results

Correlation matrix

The correlation matrix quantifies the relationship between parameters. In Figure 3 correlation between cbg and all other parameters for each patient has been visualized for the current cbg values and for the cbg values of 20 min in the future.

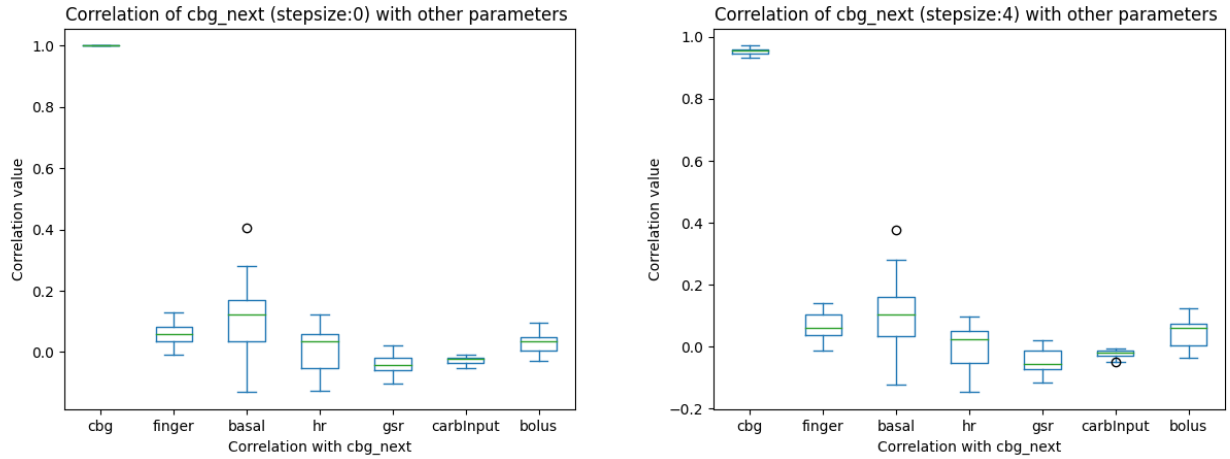

Figure 3: (left) boxplot of correlation between the parameters per patient and the current cbg value (step size: 0); (right) boxplot of correlation between the parameters per patient and the 20 min-future cbg value (step size: 4).

RNN vs SAN

In the first step, an evaluation was conducted through both visual comparison and the MSE to compare the performance of the recurrent neural network RNN and the self-attention network SAN.

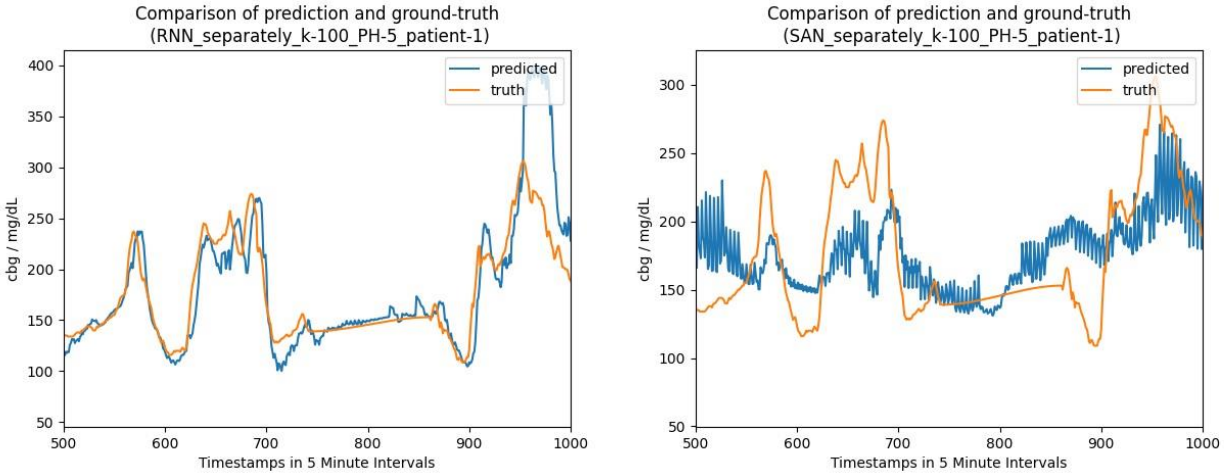

Figure 4: Comparison between predicted cbg and ground truth cbg values using a RNN as well as a SAN with k=100 and a prediction horizon of 5

In the chosen specific case of patient 1, with a k of 100 and a PH of 5, it was evident in the graphs of Figure 4, that the RNN outperformed the SAN. The RNN exhibited better adaptability to significant value changes, while the SAN struggled to maintain accuracy even in the face of larger fluctuations.
The RNN achieved a lower error percentage compared to the SAN, as visible in the Table 3. Thus, we decided on the use of the RNN for further experiments.

Table 3: Resulting MSE values in percent in the comparison between the different models.

Model	Method	Patient Nr	k	PH	MSE
RNN	Separately	1	100	5	4.2%
SAN	Separately	1	100	5	10.2%

Window

We evaluated the influence of different k values on the performance of the RNN using both visual comparison and the MSE.

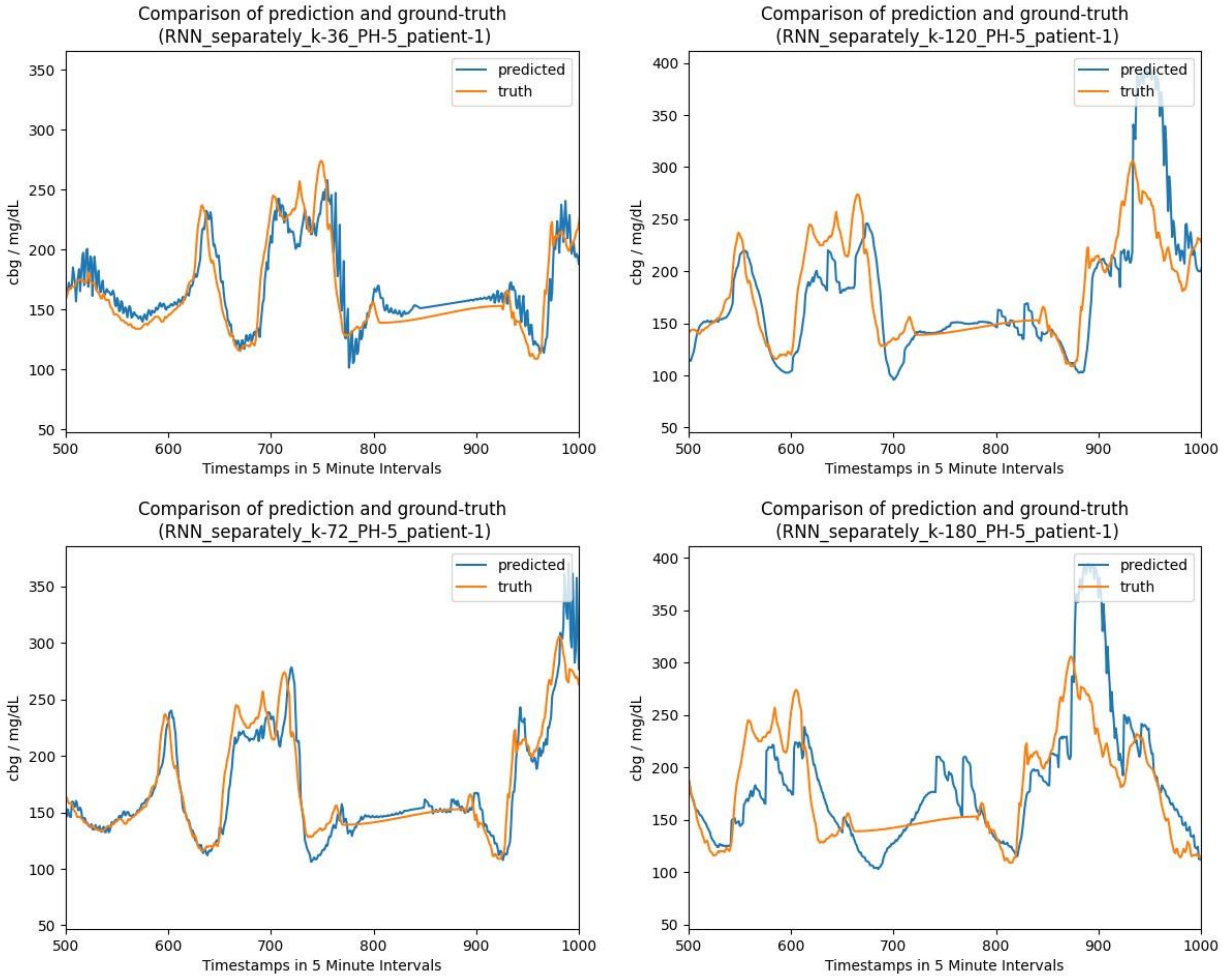

Figure 5: Comparison between predicted cbg and ground truth cbg values using RNNs with different window sizes. The windows were of size k=36, k=72, k=120 and k=180

Notably, with k values of 36, 72, 120, and 180, it was observed that the RNN exhibited varying degrees of prediction accuracy. Specifically, misbehaviour in predictions was evident in the graphs of Figure 5 corresponding to k values of 120 and 180.

The misbehaviour was also visible in the MSE values in Table 4. The lowest error percentage was shown with a k value of 72, which corresponds to the data of 6 hours. Hence, we decided on the use of k = 72 for further evaluations.

Table 4: Resulting MSE values when using different window sizes k.

Model	Method	Patient Nr.	k	PH	MSE
RNN	Separately	1	36	5	3.1%
RNN	Separately	1	72	5	2.5%
RNN	Separately	1	120	5	7.4%
RNN	Separately	1	180	5	9.9%

Prediction horizon

We evaluated the influence of different PH values on the performance of the RNN using both visual comparison and the MSE.

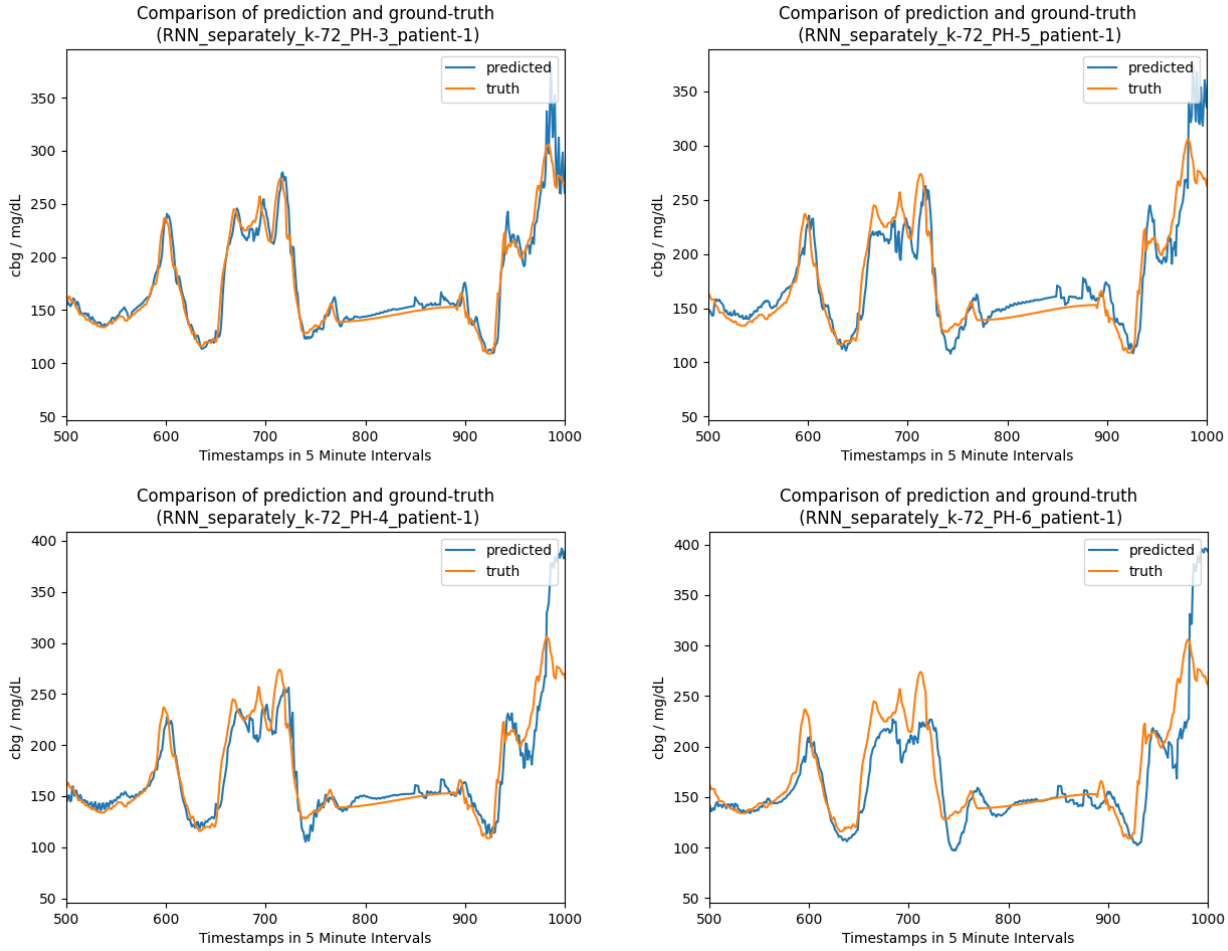

Figure 6: Comparison between predicted cbg and ground truth cbg values using RNNs with different prediction horizons. The prediction horizons were considering the time step of 5 minutes between datapoints: 15 minutes, 20 minutes, 25 minutes and 30 minutes.

The visual assessment involved comparing graphs in Figure 6 depicting the predicted and original data sequences for different prediction horizon values. Specifically, PH values of 3, 4, 5, and 6 were considered. The visual analysis revealed a trend indicating that as the PH value increased, the performance of the RNN decreased, with less accurate predictions.

The MSE values in the Table 5 show the same trend. For further evaluations we decided on the use of PH 4.

Table 5: Resulting MSE values when using different prediction horizons PH.

Model	Method	Patient Nr	k	PH	MSE
RNN	Separately	1	72	3	1.2%
RNN	Separately	1	72	4	2.1%
RNN	Separately	1	72	5	2.5%
RNN	Separately	1	72	6	6.0%

Separately vs Generalized

To choose one of the methods we analysed their performance using both visual comparison and the MSE.

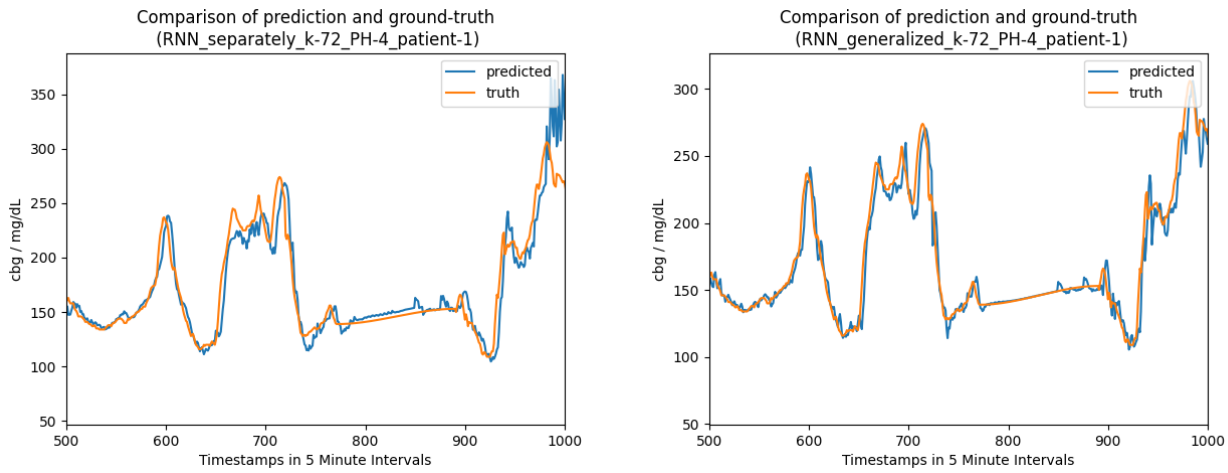

Figure 7: Comparison between predicted blood glucose level and ground truth cbg values using the separately approach and using the generalized method.

Although, the visual analysis in Figure 7 suggested relatively equal performance for both methods used, slight issues were observed in the separately method, particularly in accurately predicting smaller changes. The MSE values of Table 6 show a better accuracy for the generalized method as well.

Table 6: Resulting MSE values when using different methods.

Model	Method	Patient Nr	k	PH	MSE
RNN	Separately	1	72	4	2.1%
RNN	Generalized	1	72	4	1.3%

Figure 8 shows a direct comparison of the methods for each of the patients.

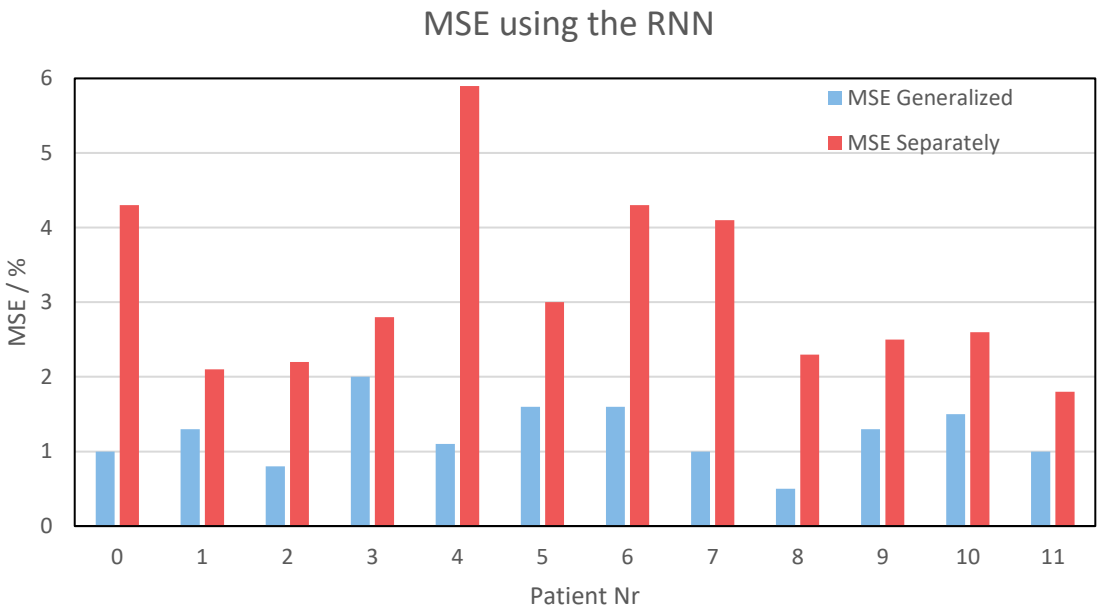

Figure 8: MSE of the whole dataset trained with RNN, k: 72 and PH: 4.

Discussion

The following can be observed from the results. It was not possible to predict blood glucose levels reliably with the SAN model, while more promising results were obtained with the RNN model. Further experiments with the SAN model were discontinued because no solution was found after investigating possible issues with the SAN model.

The RNN model showed promising results right from the start. Hence, it has been optimized. To obtain a result as close to the ground truth as possible the parameters k and PH have been adjusted. This has been initially done using Patient 1's data. Initialized using PH = 5, it was found that with a value of k = 72 (6 hours of past values) the model had the best response with an MSE = 2.5% from the truth (Table 4).
In Table 5 with PH = 3, the model had the best behavior resulting in an MSE = 1.2%. However, by increasing the prediction horizon from 15 minutes to 20 minutes, with PH = 4, the MSE increased only by 0.9%.
In comparison to the required onset time for different insulin administration forms [7] we deemed 15 minutes to be on the lower end for a useful prediction time, regarding dangerous cases of hyperglycemia. On the other hand, e.g. glucose tablets can be digested fast to prevent cases of hypoglycemia [8].
Thus, this mode had the best performance with the values k = 72 and PH = 4 translating into 6 hours of previously recorded data for a prediction 20 minutes into the future.

These parameters, k = 72 and PH = 4, were also applied for the generalized mode. As seen in Table 6, the generalized mode has a smaller error than the separate one, 1.3% compared to 2.1%. Due to this, it is concluded that the generalized mode is better. However, the generalized mode takes much more time and computational cost: 10 h compared to 1 h for the separate mode.
In Figure 8 we see in some cases there is a smaller difference (patient 1 ,2, 11) while in some cases there is a clear difference (patient 0, 4, 6, 7). This may be due to the amount of missing data in the dataset of the individual patient. Finally, the generalized mode is chosen because it has a better response for each patient and the fact that more time is needed for training does not justify choosing the separate mode.

If we consider the speed of the system, which predicts blood glucose levels in real time, the system brings an important advantage for therapy management and is possibly capable of warning the patient in case of an imminent hypo- or hyperglycemia. However, a problem of the approach in using carbInput and insulin values of the patient, considering the individual’s metabolism through which it is possible that his response to treatment may vary. Given the low correlation and its high variance, as seen in the correlation matrices (Figure 3), the limited influence of the parameter carbInput could also be considered a safety feature for cases of wrongly estimated carbohydrate counts by the patient. Subsequently, the model can be adapted to a specific patient depending on his condition.

Conclusion

The quick abandonment of the SAN model prompts reflection on potential enhancements that might have been overlooked. A more exhaustive exploration of variations in window size (k) and prediction horizon (PH) could have enriched our understanding of model sensitivity.

The extended runtime of our models, particularly with a large k parameter, highlights the need for more efficient computational resources, such as GPU acceleration. While we used UBELIX, challenges with using the TensorFlow library on its GPUs limited our optimization efforts as we were limited to the CPUs.

In hindsight, a more robust approach to handling missing Continuous Blood Glucose (cbg) values, perhaps by filtering out instances with large data gaps, could have improved model performance. The reliance on pchip interpolation might benefit from a better strategy to ensure the integrity of the data.

To streamline model training, pre-evaluating the correlation matrix (Figure 3) before running the RNN could be a valuable practice. This proactive step might help identifying and prioritizing highly correlated parameters, contributing to more focused and efficient model training. In our case as we used four fitting parameters, utilizing perhaps only parameters with higher correlations, such as only the CBG itself, could have optimized computational resources and in some cases even improved the results.

Future work should explore advancements in GPU compatibility for UBELIX, enabling seamless integration with TensorFlow. Additionally, a more exhaustive exploration of model variations and data preprocessing techniques could refine predictions, ultimately advancing the effectiveness of automated insulin delivery systems in diabetes management.

References

[1] C. Marling and R. Bunescu, “The OhioT1DM Dataset for Blood Glucose Level Prediction: Update 2020”, Accessed: Dec. 22, 2023. [Online]. Available: http://www.jdrf.org/impact/research/artificial-

[2] T. Zhu, K. Li, P. Herrero, J. Chen, and P. Georgiou, “A Deep Learning Algorithm For Personalized Blood Glucose Prediction”.

[3] T. Yang, X. Yu, N. Ma, R. Wu, and H. Li, “An autonomous channel deep learning framework for blood glucose prediction,” *Appl Soft Comput*, vol. 120, p. 108636, May 2022, doi: 10.1016/J.ASOC.2022.108636.

[4] T. Zhu, L. Kuang, K. Li, J. Zeng, P. Herrero, and P. Georgiou, “Blood glucose prediction in type 1 diabetes using deep learning on the edge,” *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 2021-May, 2021, doi: 10.1109/ISCAS51556.2021.9401083.

[5] “WaveNet: A generative model for raw audio - Google DeepMind.” Accessed: Dec. 22, 2023. [Online]. Available: https://deepmind.google/discover/blog/wavenet-a-generative-model-for-raw-audio/

[6] “OhioT1DM Dataset.” Accessed: Dec. 22, 2023. [Online]. Available: http://smarthealth.cs.ohio.edu/OhioT1DM-dataset.html

[7] “Types of Insulin | Diabetes | CDC.” Accessed: Dec. 22, 2023. [Online]. Available: https://www.cdc.gov/diabetes/basics/type-1-types-of-insulin.html

[8] A. B. Evert, “Treatment of Mild Hypoglycemia,” *Diabetes Spectr*, vol. 27, no. 1, p. 58, 2014, doi: 10.2337/DIASPECT.27.1.58.

Figures

Tables

Appendix

- GitHub repository with the code: https://github.com/Scherzkeks/Blood-Glucose-Prediction