Final Report - SugaRNN: Blood Glucose Predictions

Asa Adomatis Andrew Plattel
Tucker Mackie

August 1, 2025

1 Introduction

Diabetes affects approximately 11.6% of the U.S. population and is a chronic, lifelong disease that impacts individuals across all age groups. It arises when blood glucose levels become elevated due to the body's inability to produce or effectively respond to insulin. Managing diabetes requires continuous monitoring of glucose levels to appropriately adjust insulin dosage. Accurate forecasting of blood glucose could empower individuals to make more informed decisions regarding insulin administration, dietary choices, and physical activity.

Currently, continuous glucose monitors (CGMs) are widely used to track blood glucose levels at five-minute intervals and provide general short-term forecasts, typically in the form of high/low risk assessments for the next 15 minutes. However, they do not offer precise, quantitative predictions over longer horizons.

This study aims to evaluate and compare several machine learning models for forecasting blood glucose levels in individuals with diabetes. Specifically, we focus on using time-series models, convolutional neural networks (CNNs), and recurrent neural networks (RNNs) with a 30-minute prediction horizon to assess each model's effectiveness in making accurate, clinically useful predictions.

1.1 Exploratory Data Analysis (EDA)

This study utilizes the Hall (2018) dataset, which features a variety of 57 individuals of mixed health, including healthy, pre-diabetic, and diabetic states, who wore continuous glucose monitors (CGMs) over two to four weeks that checked glucose levels on average every 5 minutes.

BG levels also vary highly among individuals within the data set. The minimum standard deviation for BG is 9.62 mg/dL, while the maximum is 39.29 mg/dL. The range of BG also varied greatly. The minimum BG range among participants was 66.0 mg/dL while the maximum range among participants was 122.06 mg/dL.

In addition to BG time series data, the Hall dataset also includes biometric data and BG summary statistics for each individual. Example biometric data includes Age, BMI, and Height. Participants varied highly in age, ranging from 25-76, and BMI, ranging from 26 to 40 kg/m^2.

1.2 Related Work

For our related work, we focus on three separate regions: CNNs, RNNs, and Time Series modeling. One of the largest takeaways was the type of modeling that was beneficial to use, with that being an LSTM model. [6] provides a great example that illustrates the LSTM model and its constraints within this methodology. One of the largest issues with an RNN model is its struggle with long-range dependencies due to the vanishing gradient problem, which makes training it difficult. To overcome this, the authors implement a long short-term memory (LSTM) architecture. The final LSTM output was passed into a fully connected neural network with two hidden layers (512 and 256 neurons) using ReLU activations and dropout (20% and 30%) to prevent overfitting. The output layer consists of two neurons—one with linear activation and one with exponential activation—used to model BG as a univariate Gaussian distribution. The study also experimented with a glucose-specific physiological loss function that combines mean squared error with a penalty for predictions that could lead to harmful clinical interventions; however, it did not improve model performance. Data from each

patient was split into 60% for training, 20% for validation (for early stopping), and 20% for testing, with hyperparameters tuned using a grid search based on RMSE. While the model incorporated additional physiological signals such as sleep, stress, and heart rate, it focused solely on predicting future BG levels, up to 60 minutes ahead. Notably, training on combined data from all patients yielded the best results, suggesting the presence of generalizable patterns in BG variability across individuals.

For our project, one of the differences made differently from the *Blood Glucose Prediction* [6] article was the setup for the LSTM modeling. Where their methods consisted of using two layers of 512 and 256 neurons, ours consisted of using 4 different sequences: 70, 60, 24, 12, 6, totaling 228 neurons. This layout of sequences we found provided different results, but was more tailored to what we were looking for. Secondly, they were looking at an hour of data for the prediction, whereas we focused on thirty minutes. After studying this, we found that going past thirty minutes, the error increases, and predictions become more skewed.

Our next article [7] presented a novel method for predicting blood glucose levels using a stacked LSTM-based deep recurrent neural network. To correct errors from continuous glucose monitors (CGMs), particularly the older and less accurate Medtronic CGM, the authors incorporated a modified Kalman smoothing technique. Their model considered four hyperparameters: CGM values, meal information, insulin doses, and step count, and they distinguished between increasing and decreasing carbohydrate absorption phases based on meal timing. The LSTM model used a stacked architecture with 128 hidden states, and results showed that this deeper design outperformed shallow models with a single layer.

When looking at Time Series modeling, this can be a crucial step to accurately predict where the next blood glucose level will be. Using [4], we were able to find some valuable information and testing methods to assist us. [4] showed one of the largest limitations in that every human is different. When creating a model, on average, a person may fit into the model, but breaking that model out into an individual aspect is not always going to provide great results. With every human being being different, they had to enlist a lag model to provide more accurate results. As every lag length is going to be different, you fall into two separate buckets. *In bespoke analysis, either optimal lag values should be found for each individual separately, or a globally suboptimal lag value should be used for all. The former approach degenerates the analysis's congruency and imposes extra perplexity. With the latter, the fine-tuned lag is not necessarily the optimum option for all individuals. To cope with this challenge, this work suggests an interconnected lag fusion framework based on nested meta-learning analysis that improves the accuracy and precision of predictions for personalized blood glucose level forecasting.* [4] Using this knowledge of lag fusion, we will be able to build better models to fit people or adjust our overarching model.

For some of the time series testing, and even with the RNN models, we saw how different people affected our results. One of the strongest issues that we would run into is data quality because of how much of data was given. ID 26 was our best result consistently, and in doing so, we saw that it had the best RMSE in multiple models. This person also had a lot more data and more consistent data, where there weren't significant gaps. On the opposite side of the spectrum, we saw that a few different people kept performing badly across our modeling, and their given data was a little more sparse or sporadic in the BG levels. With people having different BG levels, weight, BMI, and eating habits, some errors come into the data without trying. We saw this very easily, and tried to counter some of this by removing a lot of the NaNs. By removing these and cleaning our data, I gave a better outlook on our results. For our time series model specifically, we could see this error more so than our CNN and RNN models. For our persistence model, which derives its results from the previous data points, if there are gaps or errors, it skews our results to have a worse result. Due to these issues, we believe that is why our time series model performed the worst because of the gaps and outliers. When we tried the Random Forest and ARIMA models, we ran into several issues due to our dataset. With both models, NaNs and large skews or variability drastically impact the data to where these models don't work.

[5] provides a different twist to time series modeling, and runs a model on the nocturnal aspects for people with type one diabetes. Predicting the BG levels overnight is a different task compared to during the day. Unless you work third shift, you are sleeping where you have a constant rise and fall to your levels, but the impacts of food and exercise are not there. *The Ward algorithm in hierarchical clustering works by iteratively merging the two clusters that result in a minimum increase of the total within-cluster variance after merging.* [5] Using this Ward algorithm provides a new step to time series

modeling and provides a clustering model. This method provided a lot of insight into how clustering the data impacts the result, Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). A clustering method allowed for our errors to be more ideal, and to find the underlying patterns that are within the given data.

One of the portions we observed during our testing of not just time series models, but the data itself, is that we noticed changes closer to night and overnight. With BG levels, they drop as you enter your "nighttime routine." This is due to lower activity levels, changes in diet as you don't eat as much, and other physiological or medical changes at nighttime. As we started to see this, we tried to keep our data uniform and stay away from the nighttime portions of the data for more consistent results. With our model, we wanted to keep things as consistent but also as accurate as we were able to. With a lot of volatility and variability, we chose to stick closer to the normal daytime routines.

Jaloli and Cescon [3] present a hybrid CNN-LSTM model for predicting blood glucose (BG) levels at 30-, 60-, and 90-minute horizons, using glucose measurements, meal information, and insulin intake as inputs. The model architecture includes multiple convolutional layers: two 1D convolutional layers with a kernel size of 7, followed by a 1D max pooling layer, then another pair of 1D convolutional layers with a kernel size of 5, and a final max pooling layer. The output from the convolution and pooling layers is flattened and passed to an LSTM layer, which is followed by two fully connected dense layers, ultimately producing a sequence of future predictions up to the desired prediction horizon. Dropout layers were added between LSTM layers to mitigate overfitting. Model parameters were optimized by minimizing the mean absolute error (MAE). The final model achieved an MAE of 2.69 mg/dL, an RMSE of 9.73, and an R^2 score of 91.67%. The study highlighted two key findings warranting further investigation: (1) adjusting the input sequence length based on the prediction horizon, and (2) modeling data on a per-patient basis.

2 Methods

2.1 RNN

Our models are built in the *Gluco Bench* [8] repository using the **Hall** dataset. This dataset provides BG details every five minutes for 57 different people. It provides many good use cases for all three model types: RNN, CNN, and Time Series. To start to build our model, we needed to clean the dataset. The data came in a fairly clean format and did not have extreme issues. We did have several "NAs" that had to be removed to make the models more accurate, resulting in slight data loss. The next step was that we had to adjust the time variable format to be uniform. To ensure accurate time frames, we did this by transforming it into a datetime format. Without the removal of NAs and adjusting the datetime format, our models would not run correctly or would've produced false results. When building our RNN model before the midway report, we originally extracted the blood glucose variable as our target variable to train and predict, normalized the data using the `MinMaxScaler` function from `tensorflow`, and flattened it to use it in our model. After which, we created sequences and created our train/test split using an 80/20 split. Since many of the studies we researched found LSTM models performed the best, we decided to approach the problem using the same method. After initializing the model, we fit it to the history, created predictions, inverted the predictions back to the normal scale, and calculated our RMSE.

To give ourselves a starting point, our first step was building out a baseline model. For this model, we wanted to see a few different things without a lot of legwork or alterations to the modeling process. We made a model that shows our BG levels, a prediction, and the true data results. We decided to look 6 steps into the future, which is around 30 minutes, as each step is five minutes. For this baseline, our model was made to predict without any change, and our results, to start with, were good. We calculated an RMSE of 10.12 mg/dL and an MAE of 6.12 mg/dL as a baseline. Using this baseline, this was our target goal to beat for all of our models, but we also used this as a comparison to the *Gluco Bench* [8] results. Within their paper, they had two separate models, a CNN-LSTM [3] and a LSTM model [7]. Their results for the CNN-LSTM model came out to a 9.81 RMSE and a 5.65 MAE, whereas the LSTM model had a 6.45 RMSE result.

Since the midway report, we have improved the model greatly. The first step we implemented was reorganizing the data by creating a dictionary that stored a data frame for each of the 57 IDs in the dataset. This data frame stored the respective information that used to be grouped. After creating the

dictionary, we then extracted the gl variable (BG value), removed any NaN values, and normalized the data using `MinMaxScaler` from `scikit-learn` and `fit_transform`. From her, we were able to design the model. The RNN model consisted of 2 hidden LSTM layers consisting of 50 units with ReLU activation, from which we then compiled using the ADAM optimizer and MSE for loss. The designed model had two inputs: the number of sequences in and the number of sequences out. Knowing that each sequence in our data is about 5 minutes, this allowed us to control how much time we trained the model's "history" on and how far into the future we want to predict.

After designing the model, the next step was to implement it. We knew that we wanted to try multiple history lengths to predict our window, which we decided to predict 30 minutes into the future (6 sequences). To make training, running, predicting, visualizing, and benchmarking each model easier, we designed a function that inputs the dictionary of subsets, the `seq_length_in` and the `seq_length_out`. After inputting those values, the function creates the model, creates and reshapes the sequences, and splits the data into the 80/20 train/test split. It then trains the history using 35 epochs, a batch size that is equal to the length of the subset divided by 100 (this seemed to work the best with our data size), a validation split of 0.2, and a verbose of 0. From her, the function makes predictions for each ID in the dataset, inverts the predictions back to the original scale, calculates the testing RMSE and MAE for each iteration of the sequences (6 for each in our use case of 30 mins), and an overall test RMSE and MAE for that ID. After which, it then visualizes the accuracy by plotting the predictions against the test values. In short, the function runs a model for each ID (57 models each time you run) in one cell rather than copying/pasting code multiple times.

Now that we had an easy way to test for the best training time we decided to go with `seq_length_in` values of 70 (350 mins), 60 (300 mins), 24 (120 mins), 12 (60 mins), and 6 (30 mins) with a `seq_length_out` value of 6 (30 mins) for each.

2.2 Time Series

For our time series modeling, we went down a few different routes on what we were testing for. Our main model, a Persistence model, was the focal point of the time series piece. Since the Random Forest and ARIMA models were unsuccessful, we chose against them. We got results for these models, but the end results were RMSE values of 60 mg/dL or more. Considering our baseline models were less than 10 mg/dL, this significant difference shows that these models won't fit our data set without extensive changes. With a Persistence model, there are several differences compared to our CNN and RNN models. We still have to create a training and test split, but we do not have training parameters within this model. For our training and test data, it his to be in chronological order so the results are accurate. If our data is not in chronological order, then the results come back skewed and distorted. Since we are only using the data we give it, and the parameters we set, there is no training for this model. Along with this, we also haven't learned behaviors or added layers. Most of the functions used for testing and putting our model together are functions based on he `numpy` package. To feed the data into our model, we created a loop that looks at the given time frame that we need, and gives it the `window_size` and `horizon`. These two functions give you how much past data and future data to use. Once we decided our parameters, we fed the model our desired ranges, calculated our RMSE, MAE, and output our time series plot as shown in Appendix **??**.

2.3 Convolutional Neural Network (CNN)

2.3.1 Data Processing

The CNN model used initial preprocessing performed using the `IrinaStatsLabpackage`. Continuous glucose monitoring (CGM) data were segmented into sequences of 60 measurements, representing 5 hours of blood glucose (BG) readings per sequence. These sequences were paired with biometric information from the corresponding participant. An additional characteristic, termed volatility, was calculated to quantify the number of times the CGM signal changed by more than 8 mg/dL between consecutive measurements.

For the prediction targets, sequences of 6 CGM measurements were extracted, corresponding to a 30-minute prediction horizon. These output sequences were further transformed to represent the delta (change) in mg/dL from the final value in the input sequence, allowing the model to focus on relative changes in glucose levels rather than absolute values.

2.3.2 Model Architecture

The CNN model incorporated a split input architecture, in which CGM sequences and biometric data were processed separately before being merged. The CGM input branch began with a one-dimensional convolutional layer comprising 32 filters and a kernel size of 5, followed by a 1D max-pooling layer with a pool size of 3. The biometric data branch consisted of a fully connected dense layer with 128 units.

The outputs of both branches were concatenated and passed through two fully connected dense layers, each containing 256 units. The final output layer consisted of 6 dense units, corresponding to the 6 predicted CGM values over the 30-minute horizon.

2.3.3 Model Training

Model parameters were optimized by minimizing the mean squared error (MSE) between predicted and actual outputs, defined as:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$$

The model was trained for 30 epochs. This limited number of training iterations was chosen to mitigate overfitting, which was observed when training exceeded 50 epochs. In such cases, the model achieved root mean squared error (RMSE) values as low as 4.01 mg/dL on the training set, but RMSE values exceeding 12 mg/dL on the test set. A 74/26 validation split was employed during training. This split was predefined from the `IrinaStatsLabpackage`.

2.3.4 Hyperparameter Selection

Hyperparameters were selected through grid search, exploring combinations of the number of filters and kernel sizes for the convolutional layers, pool size for max pooling, number of units in the biometric data branch, and batch size. This systematic approach ensured optimal performance within the defined model architecture.

2.3.5 Hyperparameters

Hyperparameters were chosen for the CNN using a grid search on the number of filters of the convolution layer, the kernel size of the convolution layer, the pool size of the max pooling layer, the number of dense units in the biometric side of the split input, and the batch size.

2.4 Comparison Metrics

To compare the models, we use two metrics: root mean squared error (RMSE) and mean absolute error (MAE). They are derived as follows:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|Y_i - \hat{Y}_i\right|$$

3 Results

3.1 Recursive Neural Network (RNN)

For the RNN, we obtained very promising results using the sequence values of 70, 60, 24, 12, and 6. Returning root mean squared error (RMSE) values of 12.82 mg/dL, 12.70 mg/dL, 12.84 mg/dL, 12.26

mg/dL, and 12.87 mg/dL, respectively. And mean absolute error (MAE) values of 8.45 mg/dL, 8.32 mg/dL, 8.44 mg/dL, 8.01 mg/dL, and 8.53 mg/dL. In both error metrics, the model that used 12 sequences, or 1 hour, for training history performed the best. Contextually, this makes sense because BG is going to be affected more by what's happened semi-recently compared to hours ago. What's interesting is that ID 26 had the minimum RMSE of 6.43 mg/dL. For all sequence values and the minimum MAE value for 70, 60, and 6. To explore this, we compared ID 26 with ID 1, which had the maximum RMSE for a sequence of 12 of 18.20 mg/dL. When comparing the IDs, we can see the visual difference in predictions clearly, and when looking at the underlying data from the study, it seems ID 26 had fewer spikes/dips and smoother slopes as shown in Appendix 5.

3.2 Time Series

For our time series model, we saw an RNSE of 10.59 mg/dL and an MAE of 6.14 mg/dL. This is a very small downgrade compared to our baseline model. This RMSE and MAE calculation was based on the future six intervals. The biggest constraint was factoring in the historical data without being able to reduce the outliers and data that skews our results. Clearing the NaNs helped make the RMSE better to reduce the number of gaps in the data. While testing multiple models, the time series was difficult to get a great model to beat the baseline model, since two of the better models do not fit well with our data.

3.3 Convolutional Neural Network (CNN)

For general rather than individual results, the CNN model produced a validation RMSE of 9.88 mg/dL and a validation MAE of 6.23 mg/dL. This is a large improvement over initial,l models, which could not pass the naïve persistence model benchmark. Overall, the CNN model faced challenges in producing accurate predictions independently. Overfitting was a significant concern, and achieving a low validation RMSE proved difficult. Model performance appeared highly sensitive to initialization, making the validation RMSE difficult to reproduce consistently.

3.4 Models Compared

Among the models examined, the CNN achieved the best overall performance for generalized blood glucose prediction, with a root mean squared error (RMSE) of 9.88 mg/dL. In contrast, the RNN demonstrated superior performance in personalized predictions, achieving a minimum RMSE of 6.43 mg/dL for individual participants. This low error rate is highly encouraging and highlights the potential for combining personalized and generalized modeling approaches to further enhance predictive accuracy.

4 Conclusion

During this project, we accomplished creating multiple models that were up to par, and in some cases better, than those in research papers we found. We successfully built and trained models that successfully predicted blood glucose values in a reasonable error range. We expanded on the knowledge of models that we have previously used and learned new techniques to get the models that were ideal for our goal. Using these models gives us the confidence to claim we can provide valuable insights and suggestions on implementing future models and improving the accuracy of said models.

4.1 Future Improvements

These models establish a strong baseline for blood glucose prediction; however, there are several promising avenues for improvement. Future work will focus on enhanced hyperparameter tuning via a more comprehensive grid search. Notably, in our RNN model, we observed a recurring pattern in the accuracy plots that formed a parallelogram-like shape characterized by underprediction at high glucose values and overprediction at low values. Although this behavior warrants further investigation, it was beyond the scope of the current study.

For the CNN model, we aim to incorporate batch normalization and dropout layers to refine the training process and reduce overfitting. Additional improvements may be achieved by implementing

early stopping with model checkpoints. We also plan to explore ensemble methods, such as stacking, averaging, or mixture of experts, to combine the strengths of different architectures we explored here. This is particularly relevant for integrating the RNN, which excelled in predicting consistent, patterned glucose levels for individual participants, with the CNN, which showed greater generalizability across varying patterns.

Expanding the dataset is another priority. The `IrinaStatsLabrepository` offers a wide range of additional data, which could be integrated to support more comprehensive training and improve model robustness.

To facilitate more meaningful model comparisons, we intend to standardize the preprocessing pipelines and output formats across architectures. While our current research varied these elements to probe each model's potential, unification will enable more direct and fair performance evaluations. Additionally, [2] has introduced continuous glucose–error grid analysis (CG-EGA) as a framework for evaluating the clinical acceptability of CGM prediction models. We aim to further develop our methods to align with this standard of clinical acceptance.

Finally, we envision potential deployment of this system in continuous glucose monitors (CGMs) to aid diabetic patients in real-time glucose forecasting. In the longer term, such predictive systems could be extended to inform or automate insulin delivery decisions. To support such applications, future work will also focus on quantifying prediction uncertainty, such as through confidence intervals or probabilistic outputs.

Additional Figures

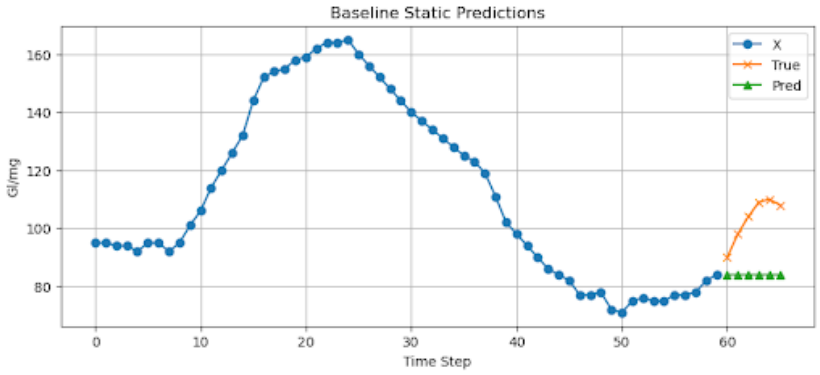

Figure 1: Baseline Prediction

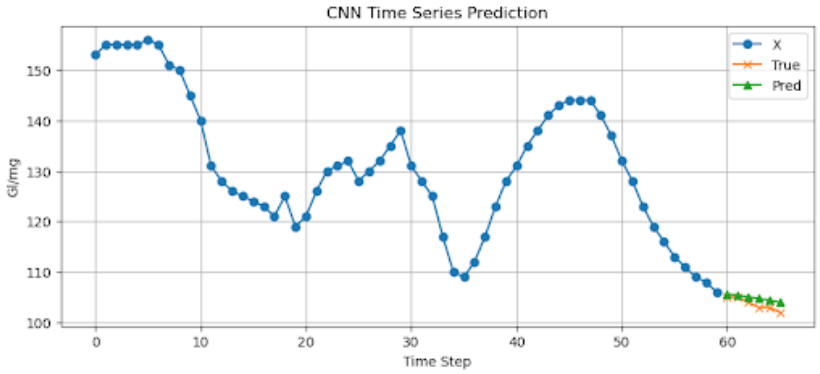

Figure 2: CNN Prediction

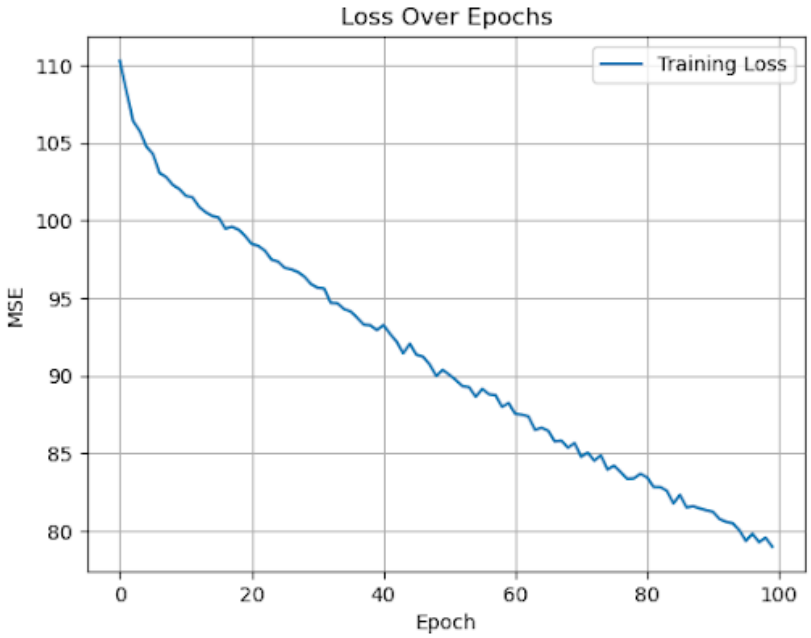

Figure 3: CNN Loss

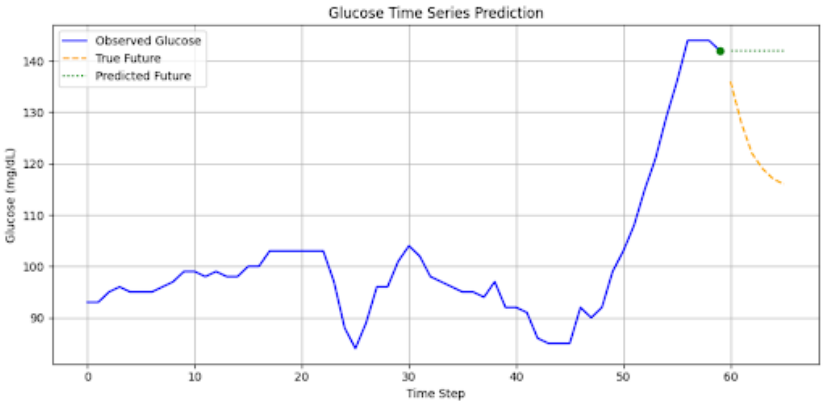

Figure 4: Glucose Time Series Prediction

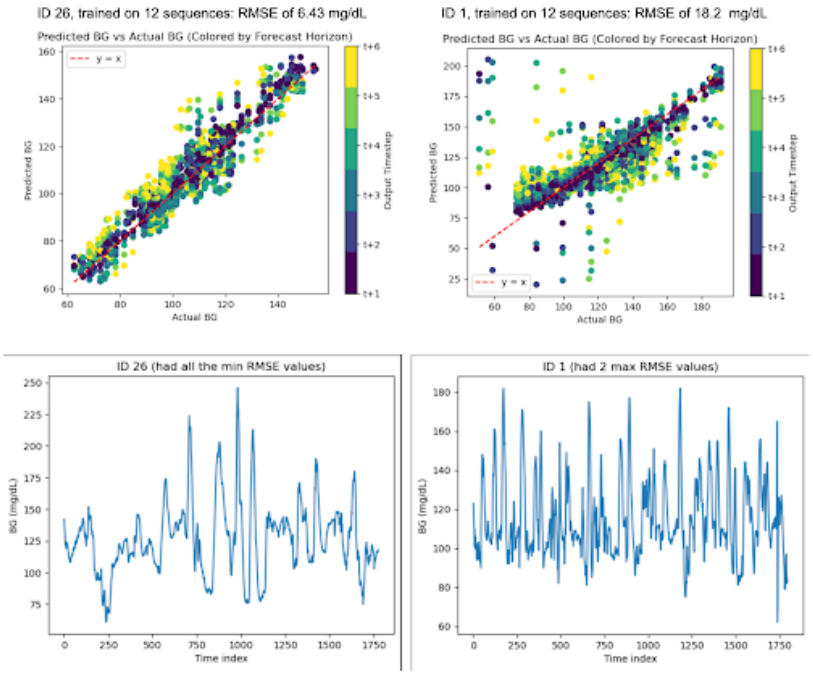

Figure 5: RNN Results

References

[1] Asa Adomatis, Andrew Plattel, and Tucker Mackie. *DS780-Project*. https://github.com/AsaAdomatis/DS780-Project. Version 2.0. 2025.

[2] PHD Boris P. Kovatchev et al. "Evaluating the Accuracy of Continuous Glucose-Monitoring Sensors: Continuous glucose–error grid analysis illustrated by TheraSense Freestyle Navigator data Free". In: *Diabetes Care* (2004).

[3] Mehrad Jaloli and Marzia Cescon. "Long-Term Prediction of Blood Glucose Levels in Type 1 Diabetes Using a CNN-LSTM-Based Deep Neural Network". In: *Journal of Diabetes Science and Technology* (2023).

[4] Heydar Khadem et al. "Blood Glucose Level Time Series Forecasting: Nested Deep Ensemble Learning Lag Fusion". In: *Journal Article* (2023).

[5] Danil E. Kladov et al. "Machine Learning Algorithms Based on Time Series Pre-Clustering for Nocturnal Glucose Prediction in People with Type 1 Diabetes". In: *Diagnostics* (2024).

[6] John Martinsson et al. "Blood Glucose Prediction with Variance Estimation Using Recurrent Neural Networks". In: *Journal of Healthcare Informatics Research* (2020).

[7] Md Fazle Rabby et al. "Stacked LSTM based deep recurrent neural network with kalman smoothing for blood glucose prediction". In: *BMC Medical Informatics and Decision Making* (2021).

[8] Renat Sergazinov et al. "GlucoBench: Curated List of Continuous Glucose Monitoring Datasets with Prediction Benchmarks". In: *arXiv* (2023).