Final Project Report for CS 184A/284A, Fall 2024

Project Title: BrisT1D Blood Glucose Prediction Competition
Project Number: Group 18

Student Name(s)

Pagnapech Chamroeun, 59143046, pjchamro@uci.edu

Visa Touch, 43257988, vtouch@uci.edu

General Instructions:

- *Your report should be 6 to 8 pages long in PDF*
- *If you want to add more details (e.g., additional graphs, examples of your system's output, etc) beyond the 8-page limit, feel free to add an Appendix to your report for additional results*
- *The sections where I expect to see mostly new material, and where I will focus on the most for grading, are Sections 4, 5 and 6, since these are the sections where you should have the most new material to report relative to earlier reports. Don't ignore the other sections, but pay particular attention to Sections 4, 5, and 6.*
- ***What the Team submits to Canvas (one student submits the items below on behalf of the team)***
    - *Your report entitled FinalReport.pdf*
    - *A zip file called project.zip that contains the following in 1 directory called project/*
        - *A README file that contains a 1 line description of each file in project/*
        - *A Jupyter notebook (called project.ipynb) that can be run directly and that demonstrates your project. Your notebook can import a sample of the data that you used, import 1 or more models that you built, and generate examples of the types of predictions or simulations your model can make. The notebook should not take any longer than 1 minute to run in total (if you have models that require a lot of training time, train them offline and just upload the models and some sample data to illustrate them). Feel free to generate examples of your model(s) in action, e.g., for reviews you could generate examples of reviews where the models work well and reviews where the models work poorly.*
        - *Also save a .html version of your notebook called project.html, showing the outputs of all the cells in the notebook.*
        - *Upload any data files needed to run project.ipynb – keep your data sets to 5MB in total or or less.*
        - *Also include a subdirectory called src (within the zipped project/ directory) with all of the individual code (scripts, modules) for Python (or equivalent*

for other languages) that your team wrote or adapted– these don't need to be called by the project.ipynb notebook but need to be in the src/ directory

- *Note that we don't necessarily plan to run all your code, but may want to look and run parts of it.*

1. Introduction and Problem Statement

The BrisT1D Blood Glucose Prediction Competition focuses on forecasting future blood glucose levels in individuals with type 1 diabetes. The problem is to predict the blood glucose levels one hour ahead of time using the data from the previous six hours of participant data. The system's inputs have five-minute intervals including blood glucose readings, insulin dosages, carbohydrate intake, and smartwatch activity metrics such as heart rate, steps, calories, and activity performed. The output is to predict the blood glucose within one hour of the end of the six hours. This project will use the Transformer for time-series analysis to predict future blood glucose, evaluating Root Mean Square Error (RMSE), Mean Squared Error (MSE), and Mean Absolute Error (MAE).

2. Related Work

Source: Khadem H, Nemat H, Elliott J, Benaissa M. Blood Glucose Level Time Series Forecasting: Nested Deep Ensemble Learning Lag Fusion. Bioengineering (Basel). 2023 Apr 19;10(4):487. doi: 10.3390/bioengineering10040487. PMID: 37106674; PMCID: PMC10135844.

According to the National Library of Medicine, this type of blood glucose level prediction was implemented using the dataset from 2018 and 2020 of OhioT1DM (*Khadem H et al*). The techniques are MultiLayer Perception (MLP), Long Short Term Memory (LSTM), and Ensemble. By applying the ensemble techniques, there are stacking and non-stacking of MLP and LSTM within the specific layers. The evaluations are root mean square error (RMSE), standard deviation (SD), mean absolute error (MAE), mean absolute percentage error (MAPE), coefficient of determination (r^2), Matthew's correlation coefficient (MCC), surveillance error (SE). average surveillance error (ASE).

3. Data Sets

Source: Sam Gordon James, Miranda Elaine Glynis Armstrong, Aisling Ann O'Kane, Harry Emerson, and Zahraa S. Abdallah. BrisT1D Blood Glucose Prediction Competition. https://kaggle.com/competitions/brist1d, 2024. Kaggle.

The dataset is provided from the Kaggle competition (*Sam Gordon James et al*). These are the CSV format in the train and test dataset, including the description of the data.

- id - row id consisting of participant number and a count for that participant
- p_num - participant number

- time - time of day in the format HH:MM:SS
- bg-X:XX - blood glucose reading in mmol/L, X:XX(H:MM) time in the past (e.g. bg-2:35, would be the blood glucose reading from 2 hours and 35 minutes before the time value for that row), recorded by the continuous glucose monitor
- insulin-X:XX - total insulin dose received in units in the last 5 minutes, X:XX(H:MM) time in the past (e.g. insulin-2:35, would be the total insulin dose received between 2 hours and 40 minutes and 2 hours and 35 minutes before the time value for that row), recorded by the insulin pump
- carbs-X:XX - total carbohydrate value consumed in grammes in the last 5 minutes, X:XX(H:MM) time in the past (e.g. carbs-2:35, would be the total carbohydrate value consumed between 2 hours and 40 minutes and 2 hours and 35 minutes before the time value for that row), recorded by the participant
- hr-X:XX - mean heart rate in beats per minute in the last 5 minutes, X:XX(H:MM) time in the past (e.g. hr-2:35, would be the mean heart rate between 2 hours and 40 minutes and 2 hours and 35 minutes before the time value for that row), recorded by the smartwatch
- steps-X:XX - total steps walked in the last 5 minutes, X:XX(H:MM) time in the past (e.g. steps-2:35, would be the total steps walked between 2 hours and 40 minutes and 2 hours and 35 minutes before the time value for that row), recorded by the smartwatch
- cals-X:XX - total calories burnt in the last 5 minutes, X:XX(H:MM) time in the past (e.g. cals-2:35, would be the total calories burned between 2 hours and 40 minutes and 2 hours and 35 minutes before the time value for that row), calculated by the smartwatch
- activity-X:XX - self-declared activity performed in the last 5 minutes, X:XX(H:MM) time in the past (e.g. activity-2:35, would show a string name of the activity performed between 2 hours and 40 minutes and 2 hours and 35 minutes before the time value for that row), set on the smartwatch
- bg+1:00 - blood glucose reading in mmol/L an hour in the future, this is the value you will be predicting **(not provided in test.csv)**

There are **508** features in the training datasets including:

- ➢ 4 features: id, p_num, time and bg+1:00
- ➢ 504 features: bg-X:XX, insulin-X:XX, carbs-X:XX, hr-X:XX, steps-X:XX, cals-X:XX, and activity-X:XX
    - 7 parameters
    - Each parameter has 72 features: (6 hours x 60 min/hr) / 5 min = 72

In addition, here is the list of provided activities:

- ➔ Indoor climbing
- ➔ Run
- ➔ Strength training
- ➔ Swim
- ➔ Bike

- ➔ Dancing
- ➔ Stairclimber
- ➔ Spinning
- ➔ Walking
- ➔ HIIT
- ➔ Outdoor Bike
- ➔ Walk
- ➔ Aerobic Workout
- ➔ Tennis
- ➔ Workout
- ➔ Hike
- ➔ Zumba
- ➔ Sport
- ➔ Yoga
- ➔ Swimming
- ➔ Weights
- ➔ Running

Based on the dataset description, the “first three months of study” data was collected from nine participants. The data is in chronological order and overlapping. The train and test dataset contains 177,024 rows and 3,644 data, respectively. In addition, this medical data is raw data, which contains missing values and noise.

4. Description of Technical Approach

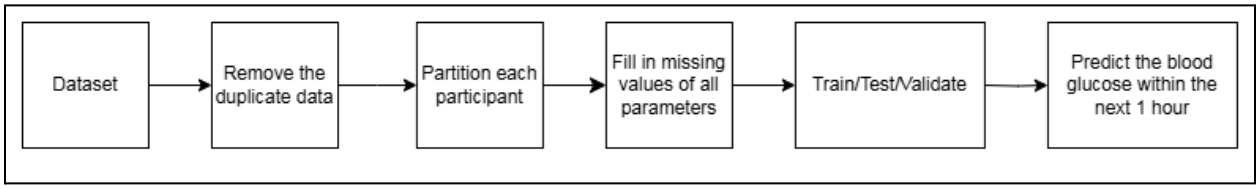

Figure 1: The Overview Process of Blood Glucose Prediction

Figure 1 shows the overview process of blood glucose prediction within the next 1 hour. This pipeline involves four major steps: data cleaning, handling missing data, data pre-processing, and evaluation, which handle the complexities of time-series data and ensure accurate predictions.

Data Cleaning

The first step in the process involves cleaning the raw dataset to ensure data quality and reliability. Duplicate and overlapping entries are removed to avoid bias or redundancy in the training phase. This step ensures that inconsistent or irrelevant entries are eliminated, which reduces the noise of the dataset. The second step is to partition the

entries based on the participant before filling in the missing values, avoiding the mixing bias of prediction for the participant.

Handling Missing Data

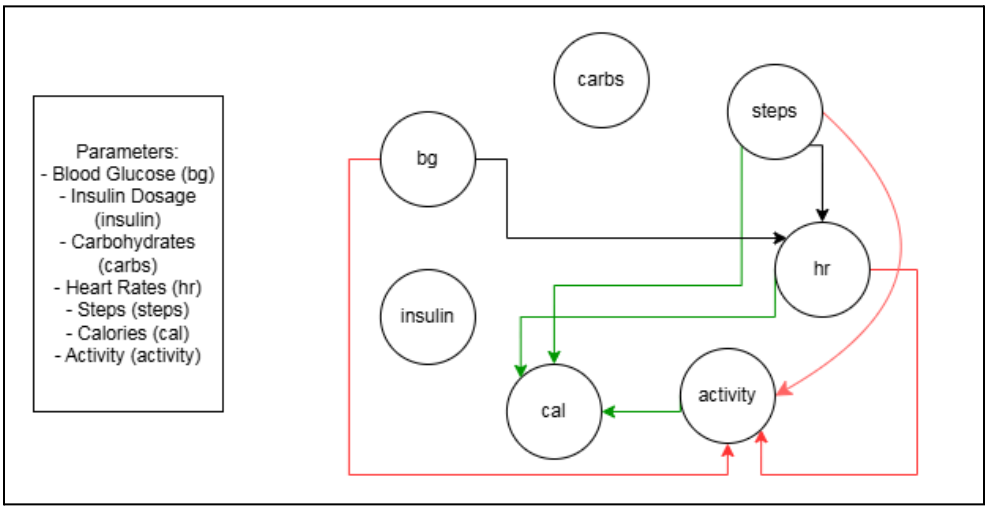

Figure 2: Relationship of Parameters for Filling Missing Values

Missing values are a common issue in healthcare datasets. Techniques such as filling in with zeros, KNNImputer, and XGB Regressor are used to preserve temporal data consistency. The graph in Figure 2 uses topological order sorting to fill in the parameters' missing values. Below are the descriptions of each technique and application to the parameters.

- Filling with zeros is a straightforward imputation method where missing values in the dataset are replaced with zeros.
    - Parameters: Steps and Carbohydrate Intake are assumably zeros
- KNNImputer (K-Nearest Neighbors Imputer) is a method that uses similarities between data points to fill in missing values. The process is as follows.
    - First, identify the k nearest neighbors (based on a distance metric like Euclidean distance) for the data point with missing values.
    - Second, it calculates the median of the neighbors' values for the feature with missing data.
    - Third, it assigns that value to the missing entries.
    - Parameters: Blood Glucose and Insulin contain most of the data, with only a few missing entries
- XGB Regressor is a machine learning-based approach that uses gradient boosting to impute missing data by predicting the missing values as a regression problem. The process is as follows.
    - First, the dataset is split into two parts: rows with complete data (used for training) and rows with missing values (used for testing).

    - Second, a regression model is trained using features from the complete data to predict the features with missing values.
    - Third, the trained model is applied to predict missing values for rows in the dataset.
    - Parameters: required the other parameters to predict its missing values
        - Heart rate (required parameters: blood glucose and steps)
        - Activity (required parameters: blood glucose, steps, and heart rate)
        - Calories Burnt (required parameters: steps, heart rate, and activity)

Model Architecture: Transformer

A clean and preprocess data is used to train a Transformer model. Transformers are particularly suitable for this task due to their ability to capture long-term dependencies and temporal patterns in sequential data. Key features of the model include:

- **Self-Attention Mechanism:** This mechanism allows the model to focus on relevant portions of the input sequence, capturing critical variations in blood glucose levels.
- **Positional Encoding:** Temporal dependencies are encoded within the data using positional encoding to ensure the model understands the order and timing of events.
- **Multi-Layer Architecture:** The model employs multiple layers of attention and feed-forward networks to enhance its learning capacity.

Hyperparameters such as the number of layers, attention heads, and learning rate are optimized to achieve the best predictive accuracy.

Model Training and Testing

The dataset is divided into training and testing sets. The training phase involves optimizing the model weights using a loss function like Root Mean Squared Error (RMSE). These metrics ensure the model minimizes prediction errors during training.

The testing phase evaluates the model's performance on unseen data. Performance metrics such as RMSE, Mean Absolute Error (MAE), and Mean Squared Error (MSE) are computed to measure the accuracy and reliability of predictions.

Evaluation and Validation

After the training phase, the validation dataset is employed to simulate real-world predictions from the Kaggle competition. Unlike supervised testing, validation is unsupervised, allowing the model to predict blood glucose levels for the next hour without prior labels. This step assesses the model's ability to generalize beyond the training data and adapt to new scenarios.

4. Software

The project software is divided into two main categories: custom code specifically written for this project and third-party libraries that were leveraged to enhance functionality. The custom code encompasses several critical components, including scripts for data preprocessing tasks such as cleaning, normalization, and cyclical encoding to handle time-series features like timestamps. It also includes methods for addressing missing data using advanced techniques like KNNImputer and XGB Regressor, as well as the implementation of a Transformer model tailored for time-series prediction. These scripts are designed to take raw time-series healthcare data as input and produce outputs like cleaned datasets, imputed values, or predictive insights about blood glucose levels.

On the other hand, third-party libraries provided robust tools that streamlined various aspects of the project. Python libraries like **pandas** and **numpy** were used extensively for data manipulation, while **scikit-learn** and **xgboost** enabled efficient imputation and gradient boosting methods. **Matplotlib** was employed for data visualization to better understand trends and preprocessing outcomes. The project relied on **PyTorch** for constructing and training the Transformer model, leveraging its powerful framework for handling neural networks and GPU acceleration. These third-party tools not only accelerated the development process but also contributed to the model's high accuracy and efficiency in processing complex healthcare time-series data.

Written code in the zip inside src folder. There are:

- dataGrouping.ipynb
- DataImputation.ipynb
- PreProcessingTestingData.ipynb
- PreProcessingTrainingData.ipynb
- TrainTestEvaluate.ipynb

5. Experiments and Evaluation

Data Imputation

As described in the technical approach section, for data imputation the main setup of experiments are KNNImputer and XGB Regressor.

For KNNImputer, before the process, the data is separated into two parts: sufficient and non-sufficient data. Apply K = 3 for the sufficient data and K = 5 for the non-sufficient data.

XGB Regressor parameters:

- n_estimators=100

- learning_rate=0.1
- max_depth=10
- random_state=42

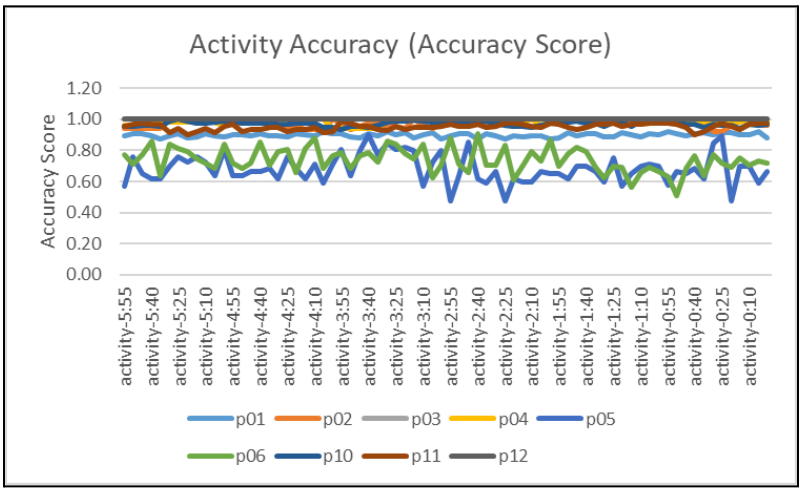

Figure 3: Activity Imputation using XGB Regressor

Figure 3 shows the activity imputation of missing data using the XGB Regressor measured by accuracy score from sklearn. Participant 5 and 6 have a fluctuating accuracy between 0.5 to 0.85, while the other participants have a good accuracy of around 0.9 to 1.

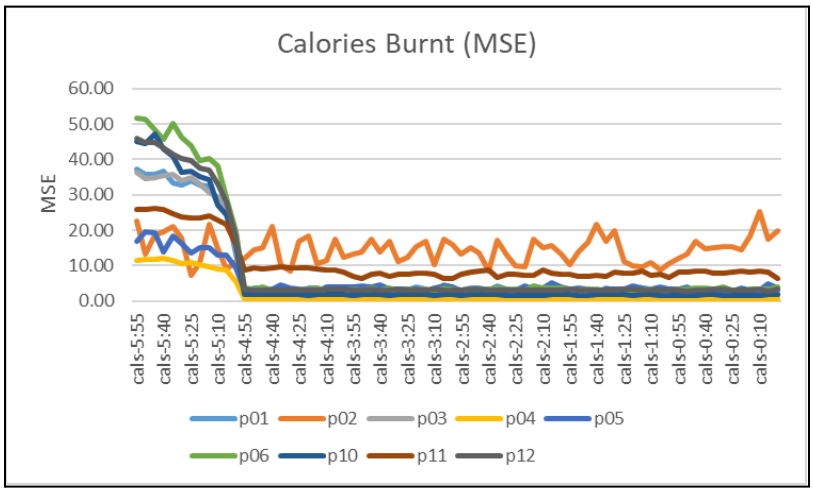

Figure 4: Calories Burnt Imputation using XGB Regressor

Figure 4 illustrates the calories burnt imputation of the missing data using the XGB Regressor measured by mean squared error (loss function). Only participant 2 has a fluctuating error, while the other participants between cals-5:55 and cals-4:55 have a high error, but the error starts to go down after cals-4:55.

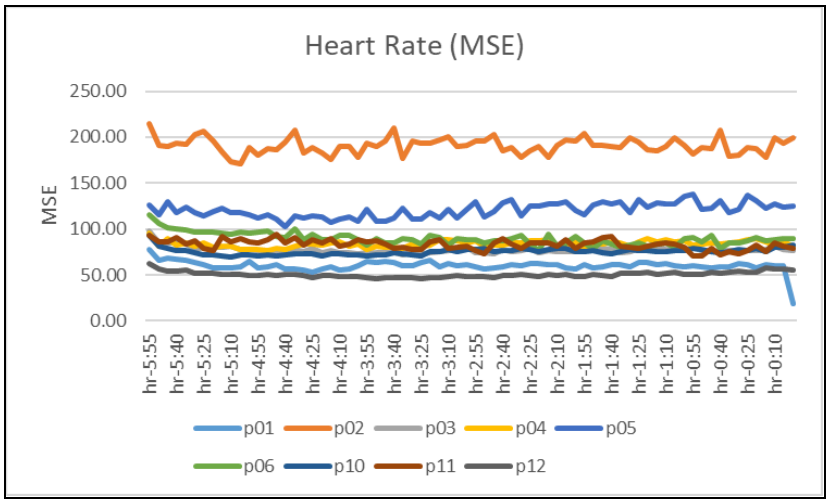

Figure 5: Heart Rate Imputation using XGB Regressor

Figure 5 displays the heart rate imputation of the missing data using the XGB Regressor measured by mean squared error (loss function). Participant 2 has the highest error among the other participants at around 200 points. There seems a large margin of error above 50 points among the participants.

Data Pre-Processing

As described in the technical approach section, once the data is cleaned and filled, the next step is data pre-processing. There are two methods applied to the parameter selection: Data Normalization and Time Cyclical Encoding. For data normalization, numeric normalization has been applied to blood glucose, insulin, carbohydrate intake, heart rate, number of steps, and calories burnt. Furthermore, the blood glucose normalization scaler is stored for the future prediction of the blood glucose, which is used to convert back to the readable value. For categorical normalization, the only parameter is activity. In addition, time-cyclical encoding has been used for time parameter to ensure that it is consistent with its data input. It is applied to the hour and minute of the time column by encoded sine and cosine functions. In the beginning, there were 508 features, but with the addition of sine and cosine functions of hour and minute, and the removal of the time feature, then the total is 511 features.

Transformer Hyperparameters

After the data is pre-processed, the transformer model is used to train the data. There are 3 main stages of the transformer: the embedding stage, transformer encoder stage, and fully connected output stage.

Hyperparameters for the Transformer Model with 12 attention heads:

- Input dimension: 511

- Embedding dimension: 192
- Number of attention heads: **12**
- Number of layers: 3
- Dropout rate: 0.1
- Criterion: Mean Squared Error
- Optimizer: Adam
- Learning rate: 0.1

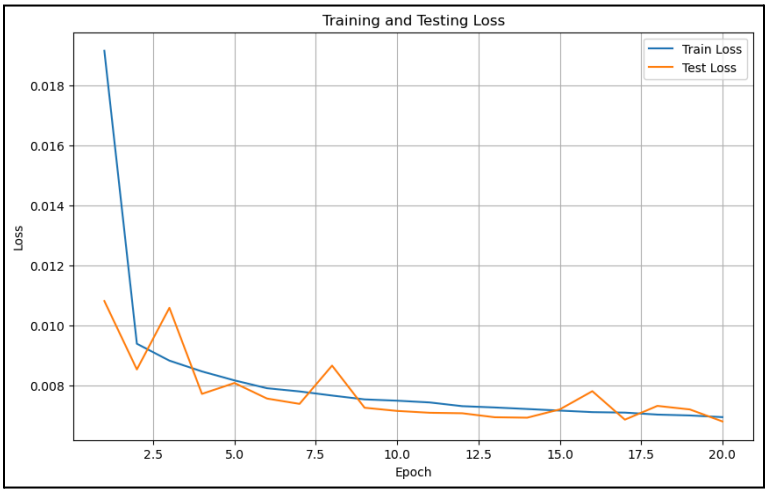

Figure 6: Transformer Loss function with the Number of Head=12 and Epoch=20

Figure 6 shows the transformer training and testing loss function with number of head is 12. The loss values for both training and testing are below 0.008.

To be able to read the blood glucose prediction values, use the blood glucose scaler from the normalization to convert it. With these hyperparameters, when submitting to Kaggle 3,644 test data for validation, the result is **6.4281** using **RMSE** evaluation.

Hyperparameters for the Transformer Model with 8 attention heads:

- Input dimension: 511
- Embedding dimension: 192
- Number of attention heads: **8**
- Number of layers: 3
- Dropout rate: 0.1
- Criterion: mean squared error
- Optimizer: adam
- Learning rate: 0.1

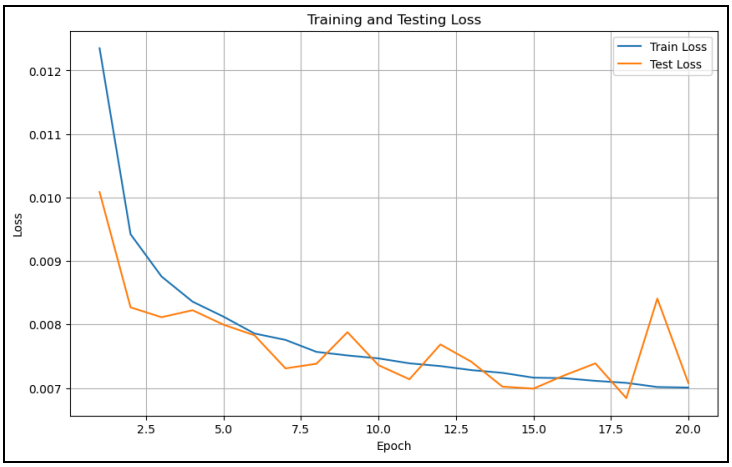

Figure 7: Transformer Loss function with the Number of Head=8 and Epoch=20

Figure 7 shows the transformer training and testing loss function with number of head is 8. The loss values for training is slightly above 0.007, while there is a fluctuation of loss function for testing at epoch 19 and 20.

Hyperparameters for LSTM Model:

- Input_size: 511
- hidden_size: **100**
- Output_size: 1
- Number of layers: 3
- Dropout_rate = 0.1

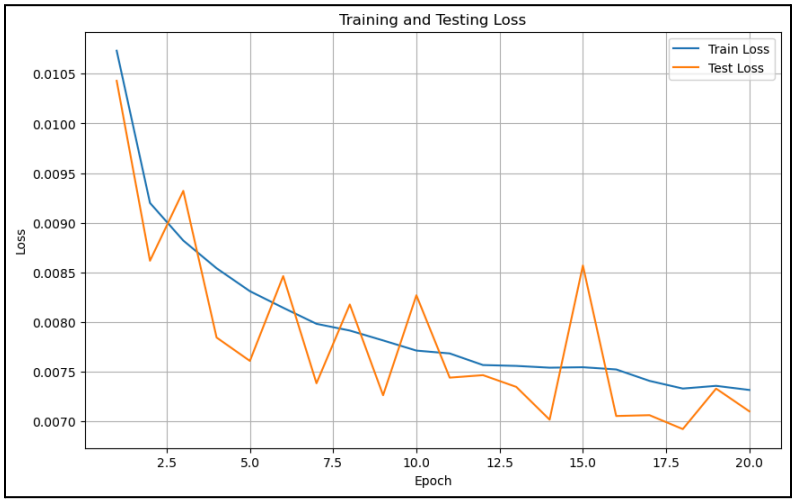

Figure 8: LSTM Loss Function (hidden_size=100)

Figure 8 shows the LSTM training and testing loss function with hidden size is 100. The loss values for training is slightly below 0.0075, while there is a fluctuation of loss function for testing among the epochs.

Hyperparameters for LSTM Model:

- Input_size: 511
- hidden_size: **200**
- Output_size: 1
- Number of layers: 3
- Dropout_rate = 0.1

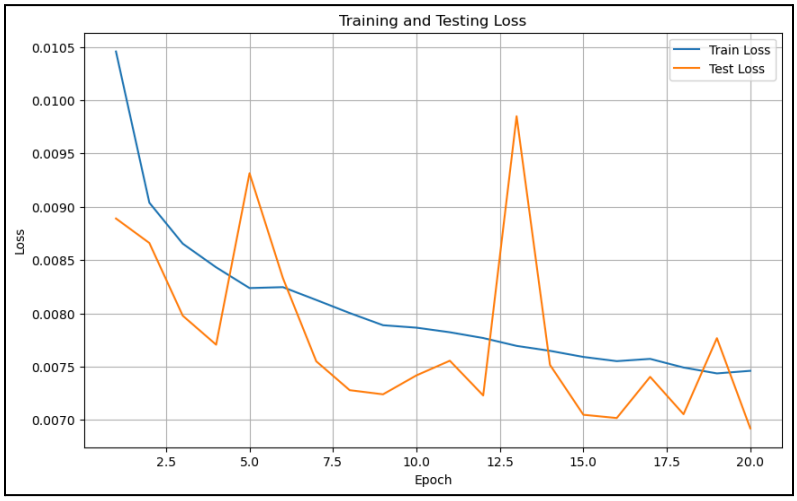

Figure 9: LSTM Loss Function (hidden_size=200)

Figure 9 shows the LSTM training and testing loss function with hidden size is 200. The loss values for training is slightly above 0.0075, while there is a fluctuation of loss function for testing at epoch 12, 13 and 14.

Transformer Train Loader

Hyperparameter Changes	Test Loss (MSE criterion)	Mean Squared Error (MSE)	Mean Absolute Error (MAE)	Root Mean Squared Error (RMSE)
Number of Head=12	0.0066	0.0066	0.0599	0.0810
Number of Head=8	0.0069	0.0069	0.0628	0.0830

Transformer Test Loader

Hyperparameter Changes	Test Loss (MSE criterion)	Mean Squared Error (MSE)	Mean Absolute Error (MAE)	Root Mean Squared Error (RMSE)
Number of Head=12	0.0068	0.0068	0.0610	0.0824
Number of Head=8	0.0071	0.0071	0.0638	0.0841

With 12 number of attention head, the transformer model performs better than 8 number of attention head.

LSTM Train Loader

Hyperparameter Changes	Test Loss	Mean Squared Error (MSE)	Mean Absolute Error (MAE)	Root Mean Squared Error (RMSE)
Hidden_Size=100	0.0186	0.0069	0.0618	0.0833
Hidden_Size=200	0.0202	0.0067	0.0606	0.0820

LSTM Test Loader

Hyperparameter Changes	Test Loss	Mean Squared Error (MSE)	Mean Absolute Error (MAE)	Root Mean Squared Error (RMSE)
Hidden_Size=100	0.0188	0.0071	0.0627	0.0843
Hidden_Size=200	0.0203	0.0069	0.0617	0.0832

With 200 number of hidden size, the LSTM model performs better than 100 number of hidden size.

Between the transformer and LSTM model, the transformer has a greater performance than LSTM when comparing the evaluation metrics of the error rate.

6. Discussion and Conclusion

This project demonstrated the transformative potential of Transformer models in predicting blood glucose levels, particularly their ability to capture long-term dependencies and temporal

patterns within sequential healthcare data. The self-attention mechanism proved highly effective in discerning meaningful relationships across time steps, a crucial feature for identifying trends in blood glucose fluctuations. A key insight was the critical role of preprocessing techniques, such as imputing missing values, standardizing inputs, and handling outliers, which maintained data integrity when working with messy, real-world healthcare datasets. Positional encoding further enhanced the model's ability to understand temporal order, while techniques like layer normalization and dropout stabilized training and reduced overfitting, especially given the small dataset size. While the model excelled in capturing periodic trends, such as diurnal variations, it struggled with abrupt changes caused by factors like exercise or stress, highlighting the limitations of purely data-driven approaches in handling domain-specific nuances without explicit contextual inputs. Small datasets also amplified noise sensitivity, occasionally leading to erratic predictions. Major limitations of current approaches include the scarcity of healthcare data, lack of contextual domain knowledge integration, high computational requirements, and challenges with interpretability.

Looking ahead, future research could focus on hybrid models that integrate domain knowledge with Transformers to improve interpretability and handle rare events more effectively. Addressing the small dataset challenge through data augmentation, synthetic data generation, or transfer learning could enhance model robustness. Real-time adaptive systems that adjust based on continuous feedback may enable personalized and dynamic predictions, while lightweight Transformer architectures could make deployment feasible on edge devices like smartwatches or continuous glucose monitors. Efforts to enhance explainability, such as using attention-based heatmaps or model distillation, are crucial for building trust in healthcare applications. Additionally, integrating multi-modal data sources, such as physical activity, diet, and stress levels, could provide richer contextual inputs for better predictions. Collaborative learning frameworks, such as federated learning, could also enable access to larger, diverse datasets while preserving patient privacy. These directions aim to bridge the gap between state-of-the-art predictive performance and real-world applicability, advancing personalized healthcare solutions.

7. A separate page on *Individual Contributions*

Pagnapech Chamroeun - I contributed to data cleaning and data pre-processing, which ensured the dataset was properly structured and free of inconsistencies for effective analysis. I dedicated significant effort to identifying and handling missing values, removing outliers, and encoding categorical variables. Additionally, I do the training of transformer and LSTM models, experimenting with hyperparameters. While the initial project framework and evaluation metrics were developed collaboratively with my partner, I took the lead in implementing the models and analyzing their results, ensuring our approach aligned with the project goals. In addition, I worked on the project proposal, presentation, and project report to ensure that it was submitted before the deadline.

Visa Touch - I contributed to several aspects of this project, starting with drafting the project proposal and ensuring a clear problem statement and methodology. I took the lead on writing

significant portions of the project report, particularly in the sections covering the introduction, dataset description, and technical approach. I also supported the training and tuning of the Transformer model, helping optimize hyperparameters and evaluate its performance using various metrics. Additionally, I managed the final project submission, ensuring all deliverables were complete and met the requirements. Many aspects of the project, such as project proposal, project report, project submission, data preprocessing and imputation strategies, were tackled jointly with my teammate.