Continuous Glucose Forecasting via Deep Learning Methods

Hamza Shaikh
Georgia Institute of Technology
hshaikh8@gatech.edu

Zafir Lari
Georgia Institute of Technology
zlari3@gatech.edu

Abstract

We performed a comparative analysis of four distinct forecasting models for use on continuous glucose measurement data. We used a Linear model, an LSTM based model, and a Transformer based model. Each were compared to each other and an ARIMA baseline model. We demonstrate advantages with the deep-learning models, however results showed ARIMA performed the best in general with LSTMs as a close second.

1. Introduction

Understanding a person's glucose level gives actionable insight into their diabetic health. Traditional glucose sensing methods require a finger-prick and can therefore only sense glucose at a single point-in-time. [1] Continuous Glucose Measurement (CGM) devices have experienced significant improvements over the last decade and can now provide accurate and frequent glucose information for patients throughout the day without requiring finger-pricks.[1]

Using time-series forecasting methods on CGM data, it may be possible to anticipate glucose out-of-range events early in-order to inform the user to take proactive action.

Today, many traditional applications of time-series forecasting rely on a single, long running time-series. This limits learning that could happen between multiple shorter time-series that measure similar processes. This limitation to traditional forecasting methods initially prevents accurate predictions for a new user. [2] Since modern CGM devices are limited to 5-14 days at a time, there are numerous short-term timeseries measuring glucose data available.[1]

We will attempt to improve glucose forecasting via deep learning methods. Since each time-series is fundamentally reading outputs from the same physiological process, we postulate that deep learning methods may be advantageous over traditional forecasting methods. With deep-learning methods we can maximize both within-patient learning and between-patient learning as depicted in Figure 1. Ideally the between-patient learning will teach the model fundamental aspects of CGM data, whereas the within patient learning will teach the model personalized aspects of the patient's data.

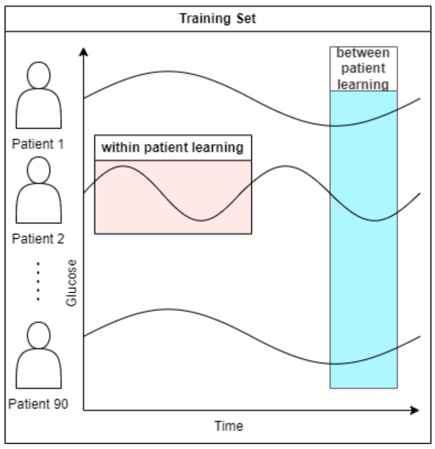

Figure 1: Between vs within patient learning

If we are successful, we will have a more reliable method to alert diabetic users of potential out of range glucose events, which will help diabetics manage their glucose levels.

The dataset is leveraged from a separate study performed by Zhao et al. The referenced study contains CGM data of diabetic patients in Shanghai, China. [3] We extracted 480 consecutive data points sampled every 15 minutes, or 5 days of data per patient, for 96 Type 2 diabetic patients to perform the forecasting. This is equivalent to 11,520 hours of CGM data. The y-value readings are the glucose levels in mg/dL. The time-axis is the time since the first reading, the actual time of day was unknown to us. There is one timeseries given per patient.

2. Approach

To investigate the performance of deep-learning models applied to forecasting CGM data we chose four different model architectures to test. 1.) ARIMA, 2.) Linear Model, 3.) LSTM-based Model, 4.) Transformer-based Model. For each deep-learning model we had 90 patients' data used for training and 6 patients' data used for testing.

2.1 ARIMA model

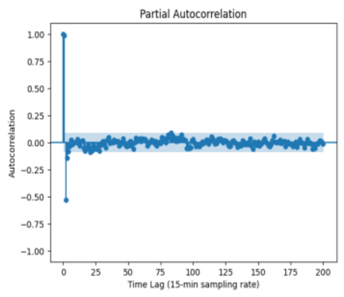

Figure 2: Partial Autocorrelation of Mean Time-Series

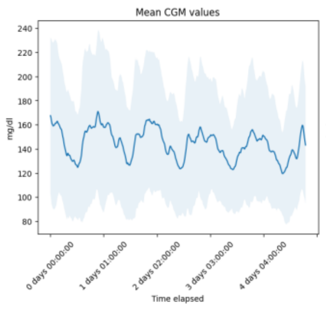

Figure 3: Mean CGM Values

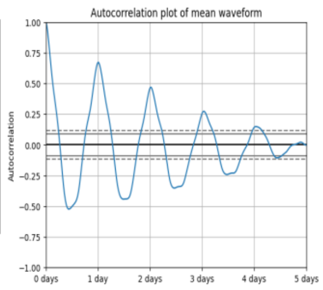

Figure 4:Autocorrelation

First is the traditional ARIMA approach, which is our baseline. The ARIMA model is an Auto-Regressive Integrated Moving Average model that is widely used for time-series forecasting. It is a non-deep-learning approach that uses multiples of past data points to predict the next datapoint. To use ARIMA the number of autoregressive terms (p), the number of differencing terms (d), and the size of the averaging window (q) needs to be chosen. Based off Figure 2 the Partial Autocorrelation shows that 3 is sufficient for p. Figure 3 shows that the time-series' rolling mean is stationary therefore a d=0. Figure 4 shows a sinusoidal autocorrelation with peaks every day; since there is no clear cut off q=0. The parameters (3,0,0) for the ARIMA model align with the findings by Otoom et al. [4]

2.2 Linear model

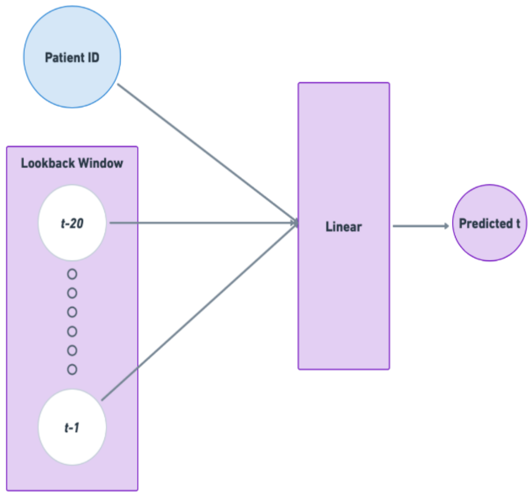

Figure 5: Linear Model

Previous literature has shown simple linear networks are effective in other forecasting applications. [5] Since linear networks cannot take in variable-sized sequences, we created lookback windows of 20 past points (5 hours), as seen in Figure 5. We chose this lookback window size via hyperparameter tuning, choosing the value where the loss was minimized and the forecasts appeared the most sensical. Since this is a forecasting problem and not a classification problem SoftMax is not applicable. Therefore, the output dimension needed to be fixed at 1; we are predicting a single glucose value for each input window. New to our approach is adding the patient ID as another input. The network was then trained to predict the next data point. Initially we did not include the patient ID as an input, which caused the network to perform poorly. This is likely because each patient likely has unique characteristics and if patient ID is not provided then those characteristics end up being averaged by the network.

2.3 LSTM Model

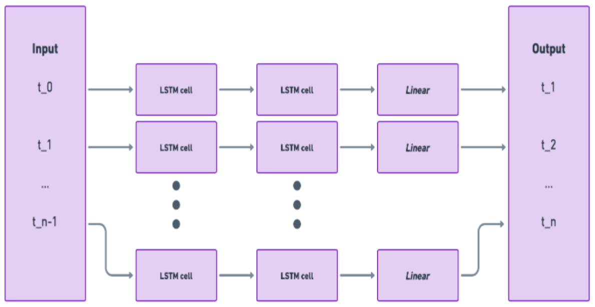

Figure 6: LSTM Network

As shown in figure 6, we created a two-layer LSTM network followed by a linear layer. This architecture was landed upon by starting from a single LSTM layer, then adding a second, then adding the linear layer. Adding the linear layer allows us to increase the hidden dimensions within the LSTM cells, while still outputting a single number for each timestep. Without the second LSTM layer the network exhibited poor performance, as it always predicted the average of the time-series data. The hidden dimension in the LSTMs and the input dimension for the linear layer was set to 50. This value arrived upon after iteratively performing hyperparameter tuning. Again, the output dimension for the linear layer was set to 1 because we are forecasting a single value for each time-point. Increasing the complexity of the architecture in these ways allowed the network to have improved forecasting.

2.4 Transformer Model

As seen in figure 7, our Transformer architecture begins with an embedding layer, followed by a positional encoder, and a transformer encoder with multiple self-attention heads and layers. The layers are proceeded by a feedforward network, and the model has a final linear layer for output. We anticipate this transformer network design will perform well at the sequential data processing due to the self-attention mechanism.

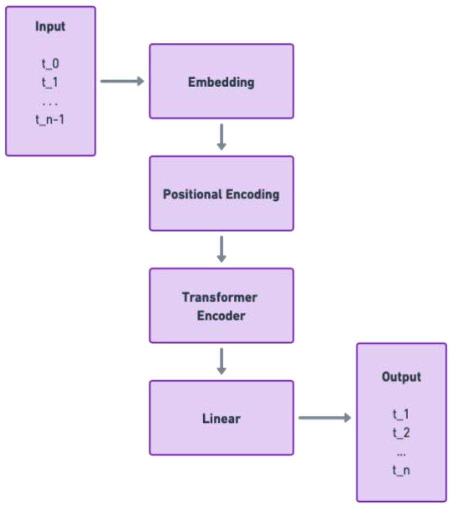

Figure 7: Transformer Network

Initially, we attempted to use this network without a positional encoding layer. This resulted in a constant predicted output. The inclusion of a positional encoder, which relies on sine and cosine functions, allows our model to properly utilize the sequence order. Even though the input sequences are numerical, an embedding layer was still required. This is because there is only one feature per time-point at the input, and the positional encoder needs to perturb multiple features for the network to understand the ordering. After adjusting hyperparameters, we found increasing the feature dimension from 1 to 4 via the embedding layer let the network understand the ordering of the input. Unfortunately, the embedding layer in pytorch does not accept float inputs. Inputs were rounded to integers, which immediately leads to some loss of accuracy in our transformer model.

Unlike our LSTM architecture, the transformer model's computational expense posed a significant challenge, considering the number of self-attention calculations and the variable sequence lengths. This potential downside, among other results, will be examined later in the report.

2.5 Loss Metric

For all models, mean absolute percentage error (MAPE) was chosen as a loss metric, optimized via Adam. This loss metric was chosen because it is a relative error and therefore the associated loss can be compared to other time-series that have different ranges. A downside with MAPE is the instability when target values are near zero, as the absolute difference is divided by the target value. Physiologically a glucose value of 0 is impossible, so that issue does not apply to this application.

2.6 Anticipated Success and Problems

Overall, we think we will be successful since the baseline ARIMA is limited to only learning from one patient, whereas our deep-learning models can learn from many patients at a time. The new aspect of the approach is to take advantage of this between-patient learning in addition to the within-patient learning to perform forecasting.

A major issue we need to address is the potential for data leakage. This is partially addressed by creating a test-set of completely separate patients. However, since this is a time-series forecasting problem, we also need to ensure there is no data leakage across the time-axis. Certain models can "look-ahead" in the input, so preventing this data-leakage is handled differently for each model. For ARIMA we only input history into the model when predicting the next time point, for the linear model we created a lookback window as inputs, LSTMs naturally can only look backwards by design, and for transformers we prevented this by only providing past values as input when predicting each output.

The other anticipated problems are model specific. For linear networks we can only use a fixed size lookback window which limits the available information, whereas ARIMA does not have this constraint. For the transformer network it may be difficult to get good results as sequences are long (up to 480 data points) and therefore the attention matrix will be large, leading to long training times.

2.7 Code Citations

All deep-learning models were custom made from PyTorch. We leveraged the torch.metrics library for the MAPE implementation, the statsmodule library for the ARIMA implementation, and PyTorch tutorials for the positional encoder implementation. [6,7,8]

3. Experiments and Results

After deciding upon the architectures and hyperparameters for reasons explained in the approach section, we conducted a series of experiments.

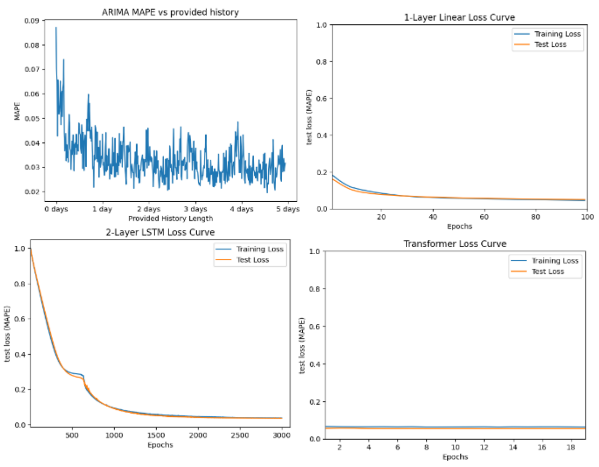

Figure 8 Characteristic Loss Curves and Arima Residuals

3.1 Experiment: Characteristic Loss Curves

Table 1: Final Test Losses (15 min prediction horizon)

	Arima	Linear	LSTM	Transformer
Final Test MAPE	0.033	0.043	0.034	0.055

Each of the deep learning models were trained on 90 patients for a prediction horizon of 15 minutes and are depicted in Figure 8. A prediction horizon denotes how far into the future the model is predicting. Since ARIMA is not a deep-learning model, instead of a loss curve we present how the MAPE changes with the length of provided history.

In general, none of the loss curves exhibit overfitting, as the training and test losses are all very close throughout each of the curves. If there was significant over fitting, we would see regions where test loss increases while the training loss decreases.

3.1.1 ARIMA Residuals

The ARIMA MAPE plot in Figure 8 was generated by creating individual models for each test patient and then calculating the MAPE for each provided history length. The MAPE steadily decreases until at least 1 day of history is provided. Then the error oscillates due to random effects. This curve validates our assumption that ARIMA performs worse when the time-series data is short.

3.1.2 Linear Model Loss Curve

The linear model loss curve shows a quick convergence compared to the LSTM model. One reason for this is that the linear model is simple with less parameters and an easy update rule. Additionally, one epoch for the linear model contains many training iterations since we split each patients' data into overlapping lookback windows.

3.1.3 LSTM Loss Curve

The LSTM model took many epochs to converge. This is likely because the input sequence is long (480 points), and LSTMs can have trouble holding memory along long sequences. Also, each epoch is less training input compared to the linear network since we didn't split the data into lookback windows here. The network finds a local minimum around epoch 500, as demonstrated by the intermediate plateau in Figure 8. In this local minimum, the LSTM simply guesses the average of the time-series for all time points. It learns a more complex relationship as it continues to decrease the loss past epoch 700.

3.1.4 Transformer Loss Curve

Due to the lengthy and computationally heavy training times for the Transformer, the model's learning process was limited to 20 epochs. The training wall-clock times were exacerbated due to the method chosen for preventing data-leakage across the time-axis. For each target output we provided all past points as input. This means the epochs for the Transformer models contain 480 times more unique inputs compared to the epochs for the LSTM models. These longer epochs allowed the loss function to converge in fewer epochs, despite the long wall-clock time. This is why one epoch is sufficient to reach a plateau for this prediction horizon. Although theoretically the Transformer model can learn complex self-attention relationships, most of the relevant information in glucose forecasting is contained in the immediately previous points, as was depicted in Figure 2.

3.1.5 Final Losses Comparison

Table 1 lists the final test MAPE values for each model when trained with 90 patients at a 15-minute prediction horizon. The ARIMA and LSTM models were almost tied for first, with the Linear model at third, and the Transformer model last. These results make intuitive sense, as Figure 2 depicted that the most salient time-points are the 3 most recent prior to the forecasted point. ARIMA and LSTMs excel with retrieving information from recent points, which explains their superior performance. The linear model was also provided these recent points; however, it cannot fully take advantage of within-patient learning as it does not have a hidden state. The transformer model performed the worst at this application, as the power of the attention matrix cannot be fully leveraged. The attention matrix adds many parameters while not providing too much value in this application since most information is contained in the most recent points. Additionally, the embedding layer used truncates the precision of our inputs, which will also contribute to the higher error.

3.2 Experiment: Extending Prediction Horizon

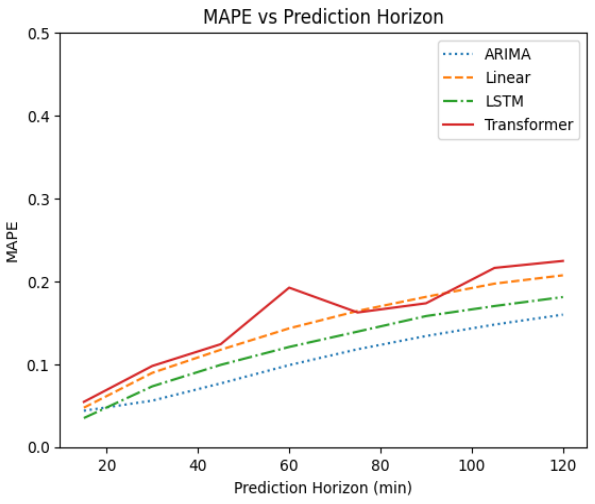

Figure 9: MAPE vs Prediction Horizon

To explore the impact of prediction horizon on the final test loss, each model was trained to predict up to different points in the future and the final test loss was compared. Figure 9 shows the results.

Comparatively, the transformer model shows additional variation, as the 20 epochs were no longer high enough for the model to converge when the prediction horizon was high. Each model seems to asymptotically approach a MAPE near 0.25. This is because a value of 0.25, in this context, is the error achieved by predicting the average of the time-series. The deep-learning models should not get worse than that value as it can always predict the mean regardless of prediction horizon.

The growth in error as prediction horizon increases is similar between all models. This indicates that, as the prediction horizon increases, information is lost from the input that none of the models can account for via their own mechanisms. As each time point is autocorrelated with 24 hours in the past (shown in figure 4). Additional days of data could allow the transformer to pay attention to 24-hour periods further back in time to have a slower growth in error, but with only 5 days of data this effect is not seen.

3.3 Experiment: Number of Training Patients

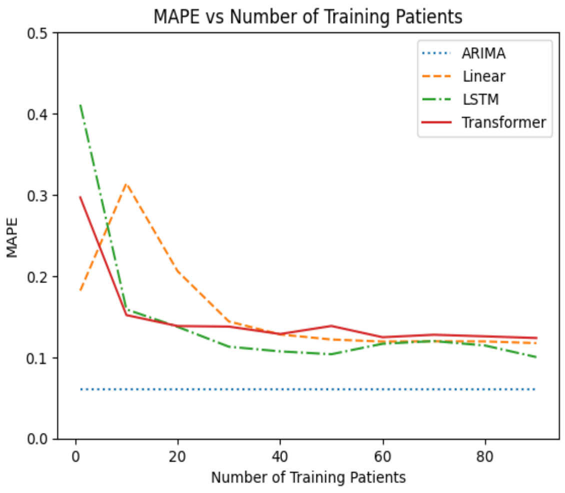

Figure 10: MAPE vs Number of Training Patients

To investigate the impact of between-patient learning, we trained each model with an increasing number of patients using a 45-minute prediction horizon. Figure 10 shows the final test loss for each model when trained with a given number of patients. ARIMA can only be trained on one patient, so the depicted line is for one patient and is for reference. For the linear model the number of training patients counter-intuitively increases the final test loss up to 10 patients. This is because the network has only a few examples of the patient ID input and is creating spurious relationships between that input and the forecasted point. As the number of training patients is increased, the linear model can take advantage of the between patient learning and the loss decreases as expected. The LSTM and Transformer perform badly with only one training patient, which is expected as both models are data hungry with many parameters. The Transformer and LSTM models have significant diminishing returns on adding additional patients past 30 patients. The amount of between patient learning has saturated by that point. Physiologically, glucose levels depend on many personalized factors and therefore the impact of between-patient learning may be saturated early.

Overall, the relative ordering of the errors between models matches what was discovered by the previous experiment at the 45-minute prediction horizon. These results exemplify the power of ARIMA, particularly on a dataset of this size.

3.4 Experiment: Qualitative Results

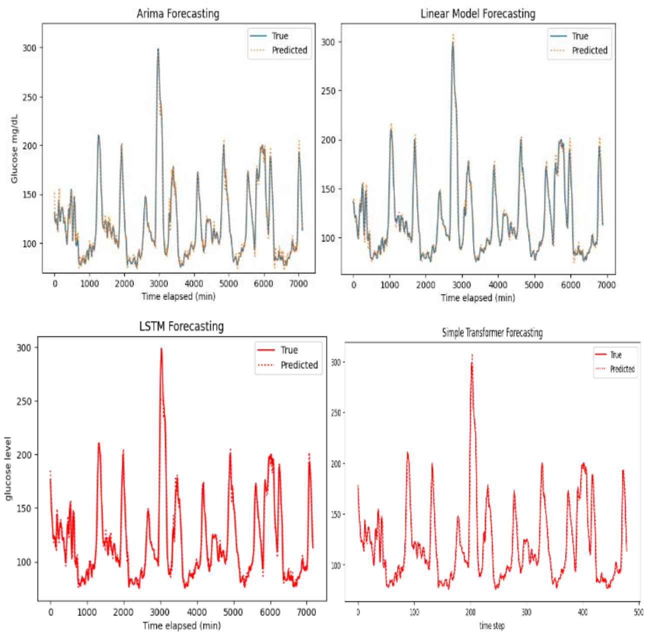

Figure 11: Forecasting Examples

To perform a qualitative inspection on the performance of each model. In Figure 11, We plotted the first out-of-sample test patient's time-series data overlapped with the predicted time-series data with a prediction horizon of 15 minutes.

ARIMA struggles early when the provided history is short, which agrees with our previous findings in Figure 8. This issue is not seen in other models. All models had trouble when the rate of change was high, especially around peaks. Comparatively, LSTM and Transformers seemed to perform the best around peaks. However, the LSTM significantly under-forecasts the peak around 3,000 minutes, which is an issue as peaks are the most critical times for patient-action. The other models tend to overestimate glucose levels around the peaks, which is a preferable behavior as it will still lead to patient action. MAPE penalizes over-estimates more than under-estimates as the metric is scaled by the target value. If desired, this bias can be removed by using mean absolute scaled error instead. This is not necessary for this application as qualitative inspection indicates a tendency for over-estimates despite using MAPE as the loss metric.

4. Conclusion

We succeeded in demonstrating that deep-learning methods can perform better than the baseline ARIMA when the provided history for an individual time-series is very small. This success was measured by the associated MAPE value. However, this effect is only relevant in that specific situation. In general, our chosen deep-learning architectures are not as robust as ARIMA particularly when the number of patients is small and when the prediction horizon is large.

Our results for each model were logically consistent and explainable via the underlying model architectures, which is another big success.

For future works it would be informative to see the impact on accuracy as the length of individual time-series increase. The performance of Transformer models in particular have an opportunity to improve as the number of daily cycles increases as they could use their self-attention mechanism to learn from correlations from the 24 hr lagged timepoints. However, for this dataset we were limited to 5 days of data per patient so that experiment could not be done.

To explore additional options available in deep-learning architectures, additional future work could incorporate non-timeseries information into the architecture, such as patient BMI or other physiological information, this approach would also be novel compared to the baseline.

Table 2: Work Division Table

Zafir Lari	• Ran prediction horizon and training size experiments on Transformer model variants. • Implemented Transformer Model • Experimented with various seq2seq and decoder-less Transformer implementations.	• See simple_transformer_final.ipynb • See simple_transformer_final.ipynb
Hamza Shaikh	• Preprocessed Dataset • Implemented ARIMA, Linear, and LSTM models. • Created prediction horizon and training size experiments. • Ran prediction horizon and training size experiments on ARIMA, Linear, and LSTM models.	• See preprocessing.ipynb • See ARIMA.ipynb • See Linear.ipynb • See LSTM .ipynb • See Produce Combined Graphs.ipynb

6.References

[1] Perez-Guzman, M. C., Shang, T., Zhang, J. Y., Jornsay, D., & Klonoff, D. C. (2021). Continuous Glucose Monitoring in the Hospital. Endocrinology and metabolism (Seoul, Korea), 36(2), 240–255. https://doi.org/10.3803/EnM.2021.201

[2] Prendin F, Del Favero S, Vettoretti M, Sparacino G, Facchinetti A. Forecasting of Glucose Levels and Hypoglycemic Events: Head-to-Head Comparison of Linear and Nonlinear Data-Driven Algorithms Based on Continuous Glucose Monitoring Data Only. Sensors. 2021; 21(5):1647. https://doi.org/10.3390/s21051647

[3] Zhao, Q., Zhu, J., Shen, X. et al. Chinese diabetes datasets for data-driven machine learning. Sci Data 10, 35 (2023). https://doi.org/10.1038/s41597-023-01940-7

[4] Otoom, Mwaffaq et al. "Real-Time Statistical Modeling of Blood Sugar." Journal of medical systems vol. 39,10 (2015): 123. doi:10.1007/s10916-015-0301-8

[5] Zeng, Ailing, et al. "Are transformers effective for time series forecasting?." Proceedings of the AAAI conference on artificial intelligence. Vol. 37. No. 9. 2023.

[6] Detlefsen, Nicki, et al. "TorchMetrics - Measuring Reproducibility in PyTorch". https://github.com/Lightning-AI/torchmetrics 2022

[7] Seabold, Skipper, and Josef Perktold. "statsmodels: Econometric and statistical modeling with python." Proceedings of the 9th Python in Science Conference. 2010.

[8] "Language Modeling with nn.Transformer and Torchtext." Language Modeling with nn.Transformer and Torchtext - PyTorch Tutorials 2.0.1+cu117 Documentation, pytorch.org/tutorials/beginner/transformer_tutorial.html. 2023.

7.Code Repository

https://github.com/rewazzu/cgm-deeplearning