Benchmark Machine Learning Approaches with Classical Time Series Approaches on the Blood Glucose Level Prediction Challenge

Jinyu Xie[1], **Qian Wang**[2],
[1] Mathworks, Inc.
[2] The Pennsylvania State University
xjygr08@gmail.com, quw6@engr.psu.edu

Abstract

There is a growing trend of applying machine learning techniques in time series prediction tasks. In the meanwhile, the classic autoregression models has been widely used in time series prediction for decades. In this paper, experiments are conducted to compare the performances of multiple popular machine learning algorithms including two major types of deep learning approaches, with the classic autoregression with exogenous inputs (ARX) model on this particular Blood Glucose Level Prediction (BGLP) Challenge. We tried two types of methods to perform multi-step prediction: recursive method and direct method. The recursive method needs future input feature information. The results show there is no significant difference between the machine learning models and the classic ARX model. In fact, the ARX model achieved the lowest average Root Mean Square Error (RMSE) across subjects in the test data when recursive method was used for multi-step prediction.

1 Introduction

The Blood Glucose Level Prediction Challenge with OhioT1DM Dataset [Marling and Bunescu, 2018] is aiming at developing powerful predictive models to predict the blood glucose level in 30 minutes. The dataset includes data collected from 6 anonymous patients with ID 559, 563, 570, 575, 588 and 591. For each patient, the following data were collected: a blood glucose level from continuous glucose monitor (CGM) every 5 minutes; periodic finger sticks blood glucose levels from; insulin doses, both bolus and basal; self-reported meal times with carbohydrate estimates; self-reported times of sleep, work, and exercise; and 5-minute aggregations of heart rate, galvanic skin response (GSR), skin temperature, air temperature, and step count.

This paper not only describes the detailed procedures of how the challenge results are obtained, but also conducts experiments to compare the performances of a wide spectrum of machine learning algorithms. In the light of No Free Lunch Theorem [Wolpert, 1996] [1], it is worth trying out various learning algorithms on a given problem. In the meanwhile, plenty of literature have reported the blood glucose level prediction results separately with different datasets using either classical time series approaches [Gani *et al.*, 2009; Eren-Oruklu *et al.*, 2009; Sparacino *et al.*, 2007; Turksoy *et al.*, 2013; Wang *et al.*, 2014; Xie and Wang, 2017] and machine learning approaches [Zecchin *et al.*, 2012; Plis *et al.*, 2014; Mirshekarian *et al.*, 2017; Mhaskar *et al.*, 2017; Fox *et al.*, 2018]. However, it is hard to compare the performance of different approaches since different datasets were used by different papers. Even with the same dataset, different features or regressors might be used to train the model, which profounds the comparison of different algorithms. We hope the comparison results listed in this paper is able to provide some empirical insights of algorithm selection on this particular blood glucose level prediction problem. The algorithms that are implemented for comparison include the classic autoregression with exogenous inputs (ARX) model, Huber Regression, Ridge Regression, Elastic Net Regression, Lasso Regression, Support Vector Regression with Linear Kernel, Support Vector with Radial Basis Kernel, Random Forest, Gradient Boosting Trees. In addition, the results of two deep learning models, a vanilla Long-Short-Term-Memory (LSTM) Network and a Temporal Convolution Network (TCN) [Bai *et al.*, 2018], are also reported. It is worth pointing out the LSTM Network model has its special mechanism to handle the memory of the sequence, which diverges from the typical autoregression manner used by the other approaches. Keep in mind that the results of LSTM Network might not be a fair comparison with the other models.

2 Model Structures

Here we introduce the general autoregression model structure which is suitable for most of the machine learning algorithms (except Recurrent Neural Network which maintains hidden states to keep memories) on time series prediction problems. In the end of the section, we introduce two methods to make multi-step predictions. To make the results of this paper comparable with other results, we will report the results using both multi-step prediction methods.

[1] There is no such a learning algorithm that outperforms any other learning algorithms in every problem set.

2.1 General Autoregression Model

The general autoregression model relies on the histories of both the target values and input feature values to make k-step-ahead predictions. The model structure is represented as follows.

$$\begin{aligned} y(t+k) =& f(y(t), ..., y(t-p+1), \\ & \mathbf{X}(t), ..., \mathbf{X}(t-q+1)) + e(t) \end{aligned} \tag{1}$$

where $y(t)$ is the target time series to predict, $\mathbf{X}(t)$ is a multivariate time series containing the input features used for prediction. p and q are the regression orders for the target sequence and the input sequence. The noise term $e(t)$ is assumed to be identically independently distributed. Now the time series prediction problem is formulated as a general regression problem. Function $f(\cdot)$ is the actual model that varies among different machine learning algorithms. All the non-deep-learning models implemented in this paper fall into this category. By fixing the regression orders p, q and varying the model type of $f(\cdot)$, fair comparison of different machine learning algorithms can be made. Note that for one-step-ahead prediction problem, when the ordinary least squares (OLS) algorithm is used to fit $f(\cdot)$, the predictive model becomes an ARX model with input order q and output order p.

2.2 Deep Learning Models

Recurrent Neural Network (RNN) and Convolution Neural Network (CNN) are the two major types of deep neural networks that achieved notable improvements recently in multiple machine learning tasks. We selected a vanilla Long-Short-Term-Memory (LSTM) RNN structure [Hochreiter and Schmidhuber, 1997] and a Temporal Convolution Network (TCN) [Bai *et al.*, 2018] as representatives to benchmark their performances in this Blood Glucose Level Prediction problem.

Vanilla Long-Short-Term-Memory Network

The vanilla LSTM network implemented in the paper is simply cascading multiple layers of LSTM cells with a fully-connected linear layer in the end to produce the predicted blood glucose value k step ahead. Unlike the general autoregression model, the prediction of the general RNN depends on both the current measurements and the recurrent hidden state:

$$\begin{aligned} y(t+k) &= f(\mathbf{h}(t), y(t), \mathbf{X}(t)) \\ \mathbf{h}(t+1) &= g(\mathbf{h}(t), y(t), \mathbf{X}(t)) \end{aligned} \tag{2}$$

RNN relies on the hidden state to keep its memory of the sequence. This fundamental structural difference between the general autoregression model and RNN model makes it difficult to make fair comparisons of their performances.

Temporal Convolution Network

The Temporal Convolution Network (TCN) we implemented is identical to the network introduced in [Bai *et al.*, 2018]. TCN adopted the key structures from state of the art networks WaveNet [Van Den Oord *et al.*, 2016] and ResNet [He *et al.*, 2016]. It utilizes the dilated causal convolution of WaveNet to obtain long term memories of the input sequence, and uses the residual connection inspired by ResNet to make training deep network easier. Similar to the vanilla LSTM Network, a fully-connected linear layer is appended in the end to make the final prediction.

Note that TCN is stateless when making predictions. The target prediction at time $t+k$ is still a function of its receptive field, i.e.

$$\begin{aligned} y(t+k) =& f_{TCN}(y(t), ..., y(t-r+1), \\ & \mathbf{X}(t), ..., \mathbf{X}(t-r+1)) + e(t) \end{aligned} \tag{3}$$

where r is the receptive field length of the TCN model. For a n layer TCN with convolution kernel size k and dilation factor d, the receptive field r is equal to $(k-1)d^n$ [Bai *et al.*, 2018]. Hence, TCN is still under the framework of the general autoregression model, which makes fair comparison possible with other machine learning algorithms.

2.3 Multi-step Prediction

There are two major strategies to obtain multi-step prediction: recursive method and direct method. To apply the recursive method, first a model is trained to predict the target value one step ahead. Then the predicted one-step ahead value is reused to produce the make the prediction of the next step. By iterating this procedure, one is able to obtain multi-step prediction. It should be noted that predicted input feature values $\hat{\mathbf{X}}(t+k)$ is also needed to make recursive predictions. To simplify the prediction procedure, the ground truth values of input features values $\mathbf{X}(\mathbf{t}+\mathbf{k})$ was used instead of the predicted values in the paper, so the prediction error only reflects the uncertainties that the model fails to capture (if predicted values $\hat{\mathbf{X}}$ are used, the final prediction error is the uncertainty of $\mathbf{X}$ plus the model uncertainty).

Direct method treats the k-step-ahead value $y(t+k)$ as target values directly and only uses the input feature values till current time t. There is no future information leakage using this method. The results using direct method is not comparable with the results using recursive method in this paper.

3 Data Cleaning

The data in the OhioT1DM Dataset is highly unsynchronized, since the data are collected from multiple devices, and some of the data is logged by patient manually. In addition, missing data exists in the data collected from the CGM sensors and the Basis Peak fitness band. Therefore, the data cleaning procedures can be split into two stages: resample stage to align all the data onto the same time grid, and imputation stage to fill the missing data.

At the resample stage, a time grid with 5 minute sample period was derived based on the start and end timestamps of the CGM blood glucose signal. Then all the signal were resampled to the time grid in a "forward filling" manner, which means if the value is missing at some time point on the designated time grid, then the closest value before the time point will be used. In addition to the "foward filling" rule, the same value can only be used once at most in the resampling process. As long as the closest previous value has been used once for resampling, the following missing values will just be tagged as missing. By resampling the data in this way, we

are able to reindex the signals onto the same timestamps and preserve the locations of the missing data at the same time. To perform this cleaning process, `reindex` function in the `pandas` package of `python` was used with `method` set to `ffill`, and `limit` set to 1. To be noted that, this resampling strategy was only applied to CGM sensor data and data collected from the Basis Peak fitness band. For the other event based data like basal bolus insulin, meal and exericse, the resample strategy is based on the context provided by the data description [Marling and Bunescu, 2018]. For example, the <**basal**> and <**temp_basal**> data only contains basal rate change time and the changed value. We combined the information provided by both to obtain the continously sampled insulin basal rate. Also notice that insulin bolus is in unit of dose in the original data. It was converted to rate (dose/min) by dividing "normal" or "square normal" type bolus by 5 minutes (sample period) and dividing "square dual" type bolus by its duration during the resampling process. Figure 1 shows the resampled data of subject 559.

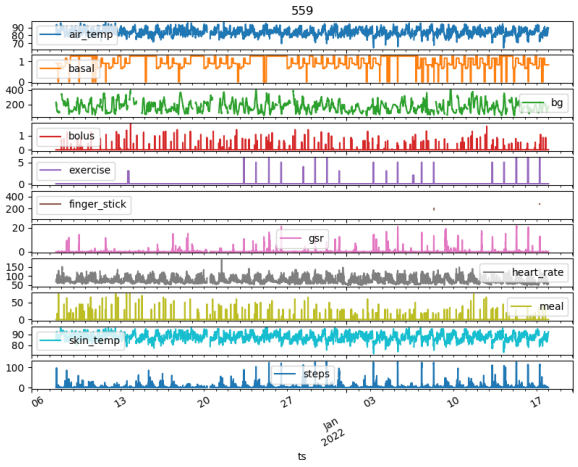

Figure 1: Resampled OhioT1DM training data, subject 559

At the imputation stage, each signal with missing data is imputed by a Kalman Smoothing technique described in [Moritz and Bartz-Beielstein, 2017]. The kernel model used by the Kalman Smoothing technique is a structural time series model introduced in [Commandeur *et al.*, 2011]. It is important to keep in mind that this Kalman Smoothing imputation technique requires the future information to perform the imputation. It cannot be used to impute data in real time. In practice, `na.kalman` function in the `ImputeTS` package of `R` was used with `method` set to "StructTS". Figure 2 shows the CGM blood glucose level before and after imputation. Same cleaning procedures were performed in the final testing dataset as well.

4 Feature Selection

Feature engineering is not a focus of this paper. Hence little effort was spent to search for useful features. The only feature

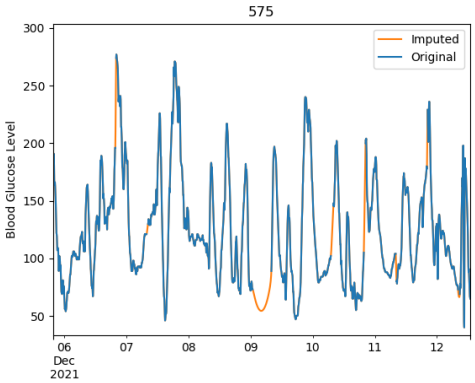

Figure 2: Blood glucose level before and after imputation

engineering we made was adding the insulin basal rate and bolus rate (the raw bolus dosage data was converted to dosage rate during the data cleaning stage) to get the total insulin delivery rate[2]. To benchmark different learning algorithms, same input features $\mathbf{X}(t)$ were used for training, which are the derived total insulin delivery rate, meal sizes, and the heart rate measured by the Basis Peak fitness band.

5 Training and Testing Data Preparation

While taking the data challenge, only the training data is available for model development. To evaluate the performances of each learning algorithm, the cleaned training data was split into sub-training data and sub-testing data. We only used the sub-training data to train the models. The sub-testing data was leveraged to pick the model for final submission of the challenge. The model that achieved the best performance on the sub-testing dataset were used to make the final prediction of the testing dataset of the challenge. In our case, the Linear Regression (ARX) model was selected for final challenge result submission.

For subject 559, 563, 570 and 588, the sub-training data takes the first 80% of the original data, while the rest is treated as the sub-testing data. For subject 575 and 591, noticeable amount of continuously missing data is observed between date Dec 26 to Dec 28 and Dec 26 to Jan 5 respectively. Therefore, the dates of missing data are used as split point for subject 575 and 591.

6 Model Training

For each learning algorithm, we trained two models for each subject, one model for 1-step-ahead (5-minute-ahead) prediction and the other for 6-step-ahead (30-minute-ahead) prediction. The 1-step-ahead predictive model was used in the recursive method to produce 6-step-ahead predictions.

Training non-deep-learning models

For training non-deep-learning models, the sub-training data is further split into 4 folds. Then three cross-validation datasets were generated: 1st fold for training and rest for

[2]The units of basal and bolus insulin are assumed to be the same, since they are not provided by the dataset.

Model	Hyperparameters
Huber	alpha=0.0001, epsilon=1.3
Ridge	alpha=0.0001
ElasticNet	alpha=0.0001, l1_ratio=0.3
Lasso	alpha=0.0001
SVR-LinearKernel	C=1, epsilon=0.01
SVR-RadialBasis	C=100, epsilon=0.01, gamma=0.003
RandomForest	n_estimators=100, min_samples_split=20, max_depth=30
GradientBoostingTrees	n_estimators=3000, max_depth=30, gamma=0.001

Table 1: Hyperparameters chosen by grid search when using the recursive method. Hyperparameter names follow `scikit-learn` and `XGB` (for GradientBoostingTrees) packages of `python`. Hyperparameters not listed in the table used default settings of the package.

Model	Hyperparameters
Huber	alpha=0.0001, epsilon=1.35
Ridge	alpha=0.0001
ElasticNet	alpha=0.0001, l1_ratio=1
Lasso	alpha=0.0001
SVR-LinearKernel	C=30, epsilon=0.015
SVR-RadialBasis	C=100, epsilon=0.03, gamma=0.008
RandomForest	n_estimators=300, min_samples_split=80
GradientBoostingTrees	n_estimators=3000, max_depth=10, gamma=0.1

Table 2: Hyperparameters chosen by grid search when using the direct method. Hyperparameter names follow `scikit-learn` and `XGB` (for GradientBoostingTrees) packages of `python`. Hyperparameters not listed in the table used default settings of the package.

validation; 1st and 2nd folds for training and rest for validation; first 3 folds for training and the last 1 fold for validation. Grid search was performed for each learning algorithm to fine-tune the hyperparameters. The model with the lowest average mean square error across all three cross-validation datasets was saved, and was used later on to generate multi-step predictions (6 steps to get 30-minute-ahead predictions in this case) on the sub-testing dataset and the final challenge testing dataset. The hyperparameters used in each model is listed in Table 1 and Table 2. Before fitting the model, the blood glucose, insulin, meal and heart rate data was normalized to the scale of 0 to 1. Note that the normalization only utilized the information of data to fit, which prevented information leakage when producing the validation scores. Exactly the same normalization operation was performed on the sub-testing data and the final challenge testing data. The regression orders p and q were set to 12 (1 hour) for all the learning algorithms.

Training deep learning models

While training the two deep learning models, the sub-training data is further split into 5 folds with the first 4 folds for training and the last fold for validation. The validation data was used for early stopping. In practice, the training algorithm checks the validation score at the end of each epoch, and only saves the model when it achieves the best validation score so far. The training and validation data was also normalized to the scale of 0 to 1 in the same way of training non-deep-learning models.

At the training stage, the original long sequence data of each subject was truncated in a sliding window fashion so that each subject will have multiple training examples. Here we set the sliding window size to 576 samples (48 hours), and step size was 12 samples (1 hour). Adam optimization with learning rate 0.001 was used in training both models. The coefficients used for computing running averages of gradient and its square were set to 0.9 and 0.999 respectively. Learning rate is shrunk to its $1/10$ if the model has been trained for more than 10 epochs and the validation loss is larger than the loss in the last 3 epochs. The gradient norm was clipped at 1. Batch size was set to 1. All the deep learning networks were trained for 20 epochs.

The vanilla LSTM Network has 3 hidden LSTM layers with hidden size of 50. The TCN model has 2 layers of TCN blocks (referred to TCN2 in Table 4 and Table 5) with kernel size of 4 and dilation factor of 2, so that it has the same regression order of 12. To investigate the performance of the TCN model with a deeper structure, another TCN model with 10 layers of TCN blocks (other parameters were the same as the shallower one) was implemented (referred to TCN10 in Table 4 and Table 5). Dropout of 0.5 was applied after each TCN layer. It should be noted that both the vanilla LSTM Network and the 10-layer TCN model might not be a fair comparison with other models.

7 Prediction

The final challenge testing dataset was first cleaned under the exactly the same procedures as the training data. With the cleaned challenge data, two sets of 30-minute-ahead prediction were produced by the 1-step-ahead models and the 6-step models using recursive method and direct method respectively. Table 3 shows the test points change after each data cleaning step, and the final test points used to produce the challenge results. Though the target blood glucose signal was imputed during the cleaning stage, the imputed values were ignored when computing the final RMSE scores. Since the regression order $p = q = 12$, and the model is making 6-step-ahead prediction, the first $12 + 6 - 1 = 17$ test points were ignored as well when generating the final results. That is why the final test points used are 17 points less than the test points after resampling. It should be noted that 1 test point was removed from the data of subject 570, and 8 new test points were introduced to the data of subject 575 during the resampling stage. The results of 575 might not be comparable in the challenge.

Subject	570	575*	588	559	591	563
Original	2745	2590	2791	2514	2760	2570
Resampled	2744	2598	2791	2514	2760	2570
Imputed	2879	2719	2880	2876	2847	2691
Test point used	2727	2581	2774	2497	2743	2553

Table 3: Test points used to compute RMSE. *8 new points were introduced in the data of subject 575 during the resampling stage, which makes the challenge results of subject 575 less comparable.

8 Results

The final test scores for each model using recursive method and direct method are reported in Table 4 and Table 5. The

Model	559	563	570	575	588	591	Mean	SD
ZOH (Baseline)	23.20	20.71	19.05	25.63	21.99	24.64	22.54	2.25
Linear Regression (ARX)	**18.36**	19.02	16.03	23.90	18.25	**21.99**	**19.59**	2.85
ElasticNet	19.28	**18.51**	16.83	24.08	19.13	22.04	19.98	2.62
GradientBoostingTrees	22.60	19.74	17.61	23.83	18.82	22.30	20.82	2.45
Huber	19.09	20.39	16.16	25.95	18.65	24.48	20.79	3.72
Lasso	19.89	18.85	17.59	24.38	20.17	22.25	20.52	2.44
RandomForest	21.35	20.04	16.90	24.60	19.72	22.31	20.82	2.61
Ridge	**18.36**	19.02	16.03	23.90	18.25	**21.99**	**19.59**	2.85
SVR-LinearKernel	18.69	19.71	**15.73**	25.00	17.90	22.80	19.97	3.38
SVR-RadalBasisKernel	19.13	20.08	15.84	25.59	**17.66**	22.23	20.09	3.45
VanillaLSTM*	69.57	56.30	46.28	90.47	71.72	73.09	67.91	15.20
TCN2	19.86	18.91	19.60	**23.46**	20.72	25.35	21.32	2.53
TCN10*	20.74	19.50	17.67	27.55	20.74	23.33	21.59	3.46

Table 4: Root Mean Square Error (RMSE) on the final challenge testing data using recursive method. Models tagged by star might not be a fair comparison. ARX results will be the final result for the challenge if future feature values are legitimate to use in the prediction.

Model	559	563	570	575	588	591	Mean	SD
ZOH (Baseline)	23.20	20.71	19.05	25.63	21.99	24.64	22.54	2.25
Linear Regression	18.30	19.11	16.02	23.85	18.20	22.25	19.62	2.89
ElasticNet	18.71	**18.51**	18.95	23.92	18.66	**22.12**	20.15	2.30
GradientBoostingTrees	20.53	19.79	18.38	23.80	19.32	22.53	20.73	2.06
Huber	18.47	19.08	16.10	25.24	17.98	22.69	19.93	3.38
Lasso	18.86	18.51	18.95	23.97	18.91	22.12	20.22	2.27
RandomForest	21.07	20.22	18.05	24.47	19.71	22.76	21.05	2.29
Ridge	18.30	19.11	16.02	23.84	18.20	22.25	19.62	2.89
SVR-Linear Kernel	18.25	19.29	15.69	24.61	17.82	22.65	19.72	3.31
SVR-RadalBasisKernel	**18.19**	19.12	**15.67**	24.61	**17.49**	22.12	**19.53**	3.27
VanillaLSTM*	21.23	18.62	17.12	25.09	19.81	22.47	20.72	2.85
TCN2	19.86	18.91	19.60	**23.46**	20.72	25.35	21.32	2.53
TCN10*	19.15	21.24	17.60	24.39	20.21	24.65	21.20	2.84

Table 5: Root Mean Square Error (RMSE) on the final challenge testing data using direct method. Models tagged by star might not be a fair comparison. Linear Regression results will be the final result for the challenge if future feature values are not legitimate to use in the prediction.

zero order hold approach (use current measurement as the prediction in 30 minutes) is used as a baseline performance.

The results using recursive method indicate that the classical ARX model performs the best in average. Ridge Regression achieved almost the same results as ARX because the coefficient of regularization penalty was set to 0.0001 as a result of grid search, which makes the Ridge Regression nearly identical to the Linear Regression used by ARX. Nonlinear models like SVR with Radial Basis Kernel, Random Forest, Gradient Boosting Trees and the two types of Deep Neural Network did not outperform other linear models. Performance degraded for some subjects when using a deeper 10-layer-TCN structure compared to the 2-layer-TCN.

It is noticed that the RMSE of the vanilla LSTM Network using recursive method is significantly higher than other models. Further investigation found that the one-step-ahead prediction of the vanilla LSTM is still accurate, while the recursive multi-step prediction accumulated undesirable errors resulting in large oscillation of the prediction. The vanilla LSTM Network potentially overfits the training data. The one-step-ahead performance of the vanilla LSTM Network is listed in Table 6.

Subject	559	563	570	575	588	591
RMSE	4.10	5.39	3.42	5.40	4.20	5.16

Table 6: One-step-ahead-prediction performance of the vanilla LSTM Network

When using direct method to make multi-step predictions, SVR with Radial Basis Kernel had the best average performance, while the Linear Regression is still very competitive with the second best average performance. Again Ridge Regression selected 0.0001 penalty on the regularization terms during the grid search, so its performance was nearly identical to Linear Regression. Unlike the recursive method, the direct method does not prone to error accumulating issues, and the vanilla LSTM Network is doing a decent job in the multi-step prediction compared to its poor performance in re-

cursive method. It is worth pointing out that TCN models failed to outperform the baseline model for some subjects.

9 Conclusion

This paper benchmarked multiple popular machine learning algorithms including the classical ARX method and 2 modern deep neural networks on the OhioT1DM Dataset. The results indicate that the classical ARX model achieved the lowest RMSE in the final testing dataset when making multi-step prediction in a recursive manner. The Linear Regression is also a good choice when making predictions with the direct method, since there is no need of hyperparameter tuning in Linear Regression and short training time can be achieved. Though the predictive models achieved decent performance in glucose level prediction, whether the predictive model captured the correct relationships between blood glucose and the input features like insulin, meal and heart rate is the still an important aspect to investigate in the future.

References

[Bai *et al.*, 2018] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[Commandeur *et al.*, 2011] Jacques JF Commandeur, Siem Jan Koopman, Marius Ooms, et al. Statistical software for state space methods. 2011.

[Eren-Oruklu *et al.*, 2009] Meriyan Eren-Oruklu, Ali Cinar, Lauretta Quinn, and Donald Smith. Estimation of future glucose concentrations with subject-specific recursive linear models. *Diabetes technology & therapeutics*, 11(4):243–253, 2009.

[Fox *et al.*, 2018] Ian Fox, Lynn Ang, Mamta Jaiswal, Rodica Pop-Busui, and Jenna Wiens. Deep multi-output forecasting: Learning to accurately predict blood glucose trajectories. *arXiv preprint arXiv:1806.05357*, 2018.

[Gani *et al.*, 2009] Adiwinata Gani, Andrei V Gribok, Srinivasan Rajaraman, W Kenneth Ward, and Jaques Reifman. Predicting subcutaneous glucose concentration in humans: data-driven glucose modeling. *Biomedical Engineering, IEEE Transactions on*, 56(2):246–254, 2009.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[Marling and Bunescu, 2018] Cindy Marling and Razvan Bunescu. The ohiot1dm dataset for blood glucose level prediction. 2018.

[Mhaskar *et al.*, 2017] Hrushikesh N Mhaskar, Sergei V Pereverzyev, and Maria D van der Walt. A deep learning approach to diabetic blood glucose prediction. *Frontiers in Applied Mathematics and Statistics*, 3:14, 2017.

[Mirshekarian *et al.*, 2017] Sadegh Mirshekarian, Razvan Bunescu, Cindy Marling, and Frank Schwartz. Using lstms to learn physiological models of blood glucose behavior. In *Engineering in Medicine and Biology Society (EMBC), 2017 39th Annual International Conference of the IEEE*, pages 2887–2891. IEEE, 2017.

[Moritz and Bartz-Beielstein, 2017] Steffen Moritz and Thomas Bartz-Beielstein. imputeTS: Time Series Missing Value Imputation in R. *The R Journal*, 9(1):207–218, 2017.

[Plis *et al.*, 2014] Kevin Plis, Razvan C Bunescu, Cindy Marling, Jay Shubrook, and Frank Schwartz. A machine learning approach to predicting blood glucose levels for diabetes management. In *AAAI Workshop: Modern Artificial Intelligence for Health Analytics*, number 31, pages 35–39, 2014.

[Sparacino *et al.*, 2007] Giovanni Sparacino, Francesca Zanderigo, Stefano Corazza, Alberto Maran, Andrea Facchinetti, and Claudio Cobelli. Glucose concentration can be predicted ahead in time from continuous glucose monitoring sensor time-series. *Biomedical Engineering, IEEE Transactions on*, 54(5):931–937, 2007.

[Turksoy *et al.*, 2013] Kamuran Turksoy, Elif S Bayrak, Laurie Quinn, Elizabeth Littlejohn, and Ali Cinar. Adaptive multivariable closed-loop control of blood glucose concentration in patients with type 1 diabetes. In *American Control Conference (ACC), 2013*, pages 2905–2910. IEEE, 2013.

[Van Den Oord *et al.*, 2016] Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[Wang *et al.*, 2014] Qian Wang, Peter Molenaar, Saurabh Harsh, Kenneth Freeman, Jinyu Xie, Carol Gold, Mike Rovine, and Jan Ulbrecht. Personalized state-space modeling of glucose dynamics for type 1 diabetes using continuously monitored glucose, insulin dose, and meal intake an extended kalman filter approach. *Journal of diabetes science and technology*, 8(2):331–345, 2014.

[Wolpert, 1996] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

[Xie and Wang, 2017] Jinyu Xie and Qian Wang. A personalized diet and exercise recommender system in minimizing clinical risk for type 1 diabetes: An in silico study. In *ASME 2017 Dynamic Systems and Control Conference*, pages V001T08A003–V001T08A003. American Society of Mechanical Engineers, 2017.

[Zecchin *et al.*, 2012] Chiara Zecchin, Andrea Facchinetti, Giovanni Sparacino, Giuseppe De Nicolao, and Claudio Cobelli. Neural network incorporating meal information improves accuracy of short-time prediction of glucose concentration. *Biomedical Engineering, IEEE Transactions on*, 59(6):1550–1560, 2012.