Predicting Blood Glucose with Deep Neural Network and Deep Reinforcement Learning

---

Maria Panagiotou

Contents

1 Introduction

Diabetes mellitus is a group of metabolic diseases that affect the body's ability to regulate blood glucose levels. Diabetes can be classified into Type 1 diabetes (T1DM) and Type 2 diabetes (T2DM). In the former, the immune system attacks the insulin producing pancreatic cells resulting in absolute deficiency of insulin secretion, while T2DM is characterized by increased resistance of the body cells to insulin, which frequently coexists with limited insulin secretion [1]. In this work, we focus on patient with T1DM, who need to compensate for the absence of insulin secretion by administering exogenous artificial insulin. In this report, we provide an implementation of time series forecasting for blood glucose measurements in 30 and 60 minutes prediction horizons, using the OhioT1DM dataset [2]. For the deep learning approach an Long-short-term memory (LSTM) neural network and a bi-directional LSTM neural network (bi-LSTM) were implemented, while for the reinforcement learning (RL) approach a custom environment was implemented with the purpose of addressing the problem with deep reinforcement learning (Deep RL).

2 Dataset

The OhioT1DM dataset [2] contains eight weeks' data of continuous glucose monitoring, insulin, physiological sensor, and self-reported life-event data for each of 12 people with T1DM. All people used Medtronic 530G or 630G insulin pumps and Medtronic Enlite CGM sensors throughout the 8-week data collection period. Table 1 shows the gender, age range, insulin pump model, and sensor band type for each data contributor, by cohort.

Specifically, the dataset includes:

- A CGM blood glucose (cbg) level every 5 minutes.
- Blood glucose levels from periodic self-monitoring of blood glucose (finger sticks).
- An indicator if the cbg measure is missing for a certain timestamp (missing cbg).
- Insulin doses (both bolus and basal).
- Self-reported meal times with carbohydrate estimates.
- Heart rate, aggregated in 5 minutes intervals.
- Galvanic skin response (gsr), aggregated in 5 minutes intervals.

ID	Gender	Age	Pump Model	Sensor Band	Cohort
540	male	20-40	630G	Empatica	2020
544	male	40-60	530G	Empatica	2020
552	male	20-40	630G	Empatica	2020
567	female	20-40	630G	Empatica	2020
584	male	40-60	530G	Empatica	2020
596	female	40-60	530G	Empatica	2020
559	female	40-60	530G	Basis	2018
563	male	40-60	530G	Basis	2018
570	male	40-60	530G	Basis	2018
575	female	40-60	530G	Basis	2018
588	female	40-60	530G	Basis	2018
591	female	40-60	530G	Basis	2018

Table 1: Gender, age range, insulin pump model, and sensor band type for each data contributor, by cohort [2]

2.1 Data Preprocessing

The given data are aggregated into 5 minutes intervals. For the first preprocessing step the missing values from the dataset were checked, to observe missing values in all the features. In the case of the heart rate and gsr features several missing values were observed, which could not be filled. For the basal insulin the missing values were filled using the last valid observation and in case of not prior observations the next valid observation was used to fill the gaps. For the continuous blood glucose measurement the missing values initially were replaced with the finger stick measurement if it was available, otherwise filled with the median filling method. For the bolus insulin and the carbon intakes because such features are manually recorder from the person, the missing values were considered as zero values. Furthermore, from all the features only four were selected: cbg, basal insulin, insulin boluses and the carbohydrate intakes (ch), because they have the highest impact on blood glucose dynamic. Finally, the data were normalized with Minmax scaler. Table 2 shows the number of the training and test set after the preprocessing, which was the same as before the preprocessing.

ID	BGLP Challenge	Training Examples	Test Examples
540	2020	13109	3065
544	2020	12671	3136
552	2020	11097	3950
567	2020	13535	2870
584	2020	13247	2995
596	2020	13629	3002
559	2018	12080	2876
563	2018	13097	2691
570	2018	11611	2891
575	2018	13103	2718
588	2018	13105	2880
591	2018	12755	2847

Table 2: Number of training and test examples per data contributor after preprocessing

As an input for the following architectures a sliding window was used with a span of 120 minutes (or 24 samples of the 5 minutes intervals), which was used to estimate the future value of blood glucose in the different prediction horizons. The training and test data are split as provided by the OhioT1DM database, for each person separately.

3 Architectures

3.1 Long short-term memory (LSTM)

LSTM is an advanced Recurrent Neural Network (RNN) architecture, which overcomes the vanishing gradient problem and effectively captures long term dependencies [3]. The most common variant of LSTM is described by:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(W_{xf}x_t = W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{2}$$

$$c_t = f_t c_{t-1} + i_t tanh(W_{hc}x_t + W_{hc}h_{t-1} + b_c)) \tag{3}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{4}$$

$$h_t = o_t tanh(c_t) \tag{5}$$

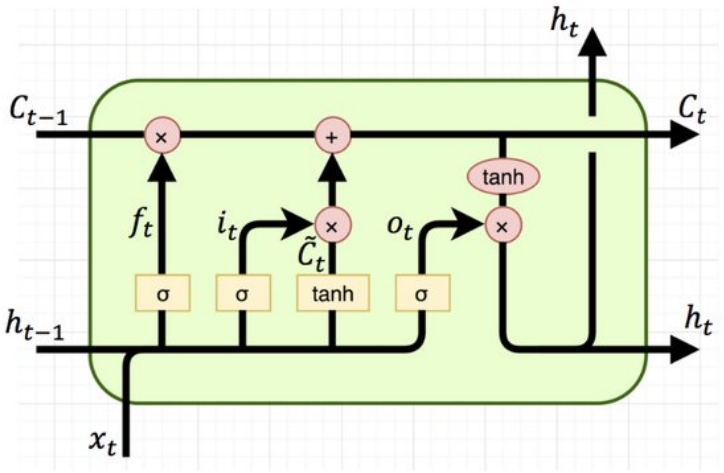


Figure 1: Block diagram of LSTM cell.

Figure 1 shows the memory cell (neuron) of an LSTM, which contains one self connected memory cell c_t and three multiplicative gates, e.g., the input gate i_t, the forget gate f_t and the output gate o_t. The forget gate is responsible to decide whether we should keep the information from the previous timestamp or forget it (Equation 2). The input gate is used to quantify the importance of the new information carried by the input (Equation 1). The output gate controls the amount of information passed from the cell to the output (Equation 4). The memory cell has a self connected edge of weight 1, so that the gradient does not vanish or explode (Equation 3).

The Bidirectional long short term memory (Bi-LSTM) is a type of LSTM model which processes the data in both forward and backward direction. This feature of flow of data in both directions makes the Bi-LSTM different from other LSTMs.

The first implemented architecture is an LSTM timeseries predictor. We considered one LSTM layer with 50 units and 2 fully connected layers with 20 and 10 units, respectively. The output layer, the one unit dense layer, was used to output the final predicted blood glucose value.The activation function of relu is used for the LSTM blocks. The network is trained for 100 epochs, batch size of 1, with the mean squared error (mse) as loss function and the Adam optimizer with learning rate at 0.01. Additionally, early stopping is used with patience of 3 to prevent overfitting. Figure 2 shows the input and output dimension of each layer.

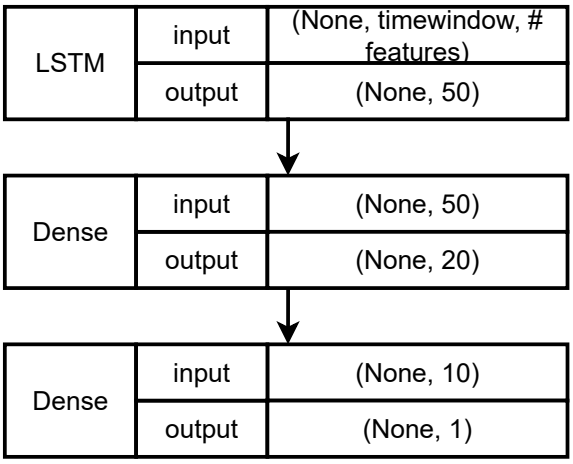


Figure 2: LSTM Architecture.

To expand the aforementioned architecture we implemented a bidirectional LSTM as depicted in Figure 3. The architectures comprised one LSTM layer with 4 units and 1 bi-LSTM layer with 8 units, followed by 5 fully connected layers with 8, 4, 64 and 4 units, accordingly. The activation function of relu is used for the LSTM blocks. The network is trained for 100 epochs, batch size of 1, with the mean squared error (mse) as loss function and the Adam optimizer with learning rate at 0.01. Moreover, early stopping is used with patience of 3 to prevent overfitting, similar to LSTM model.

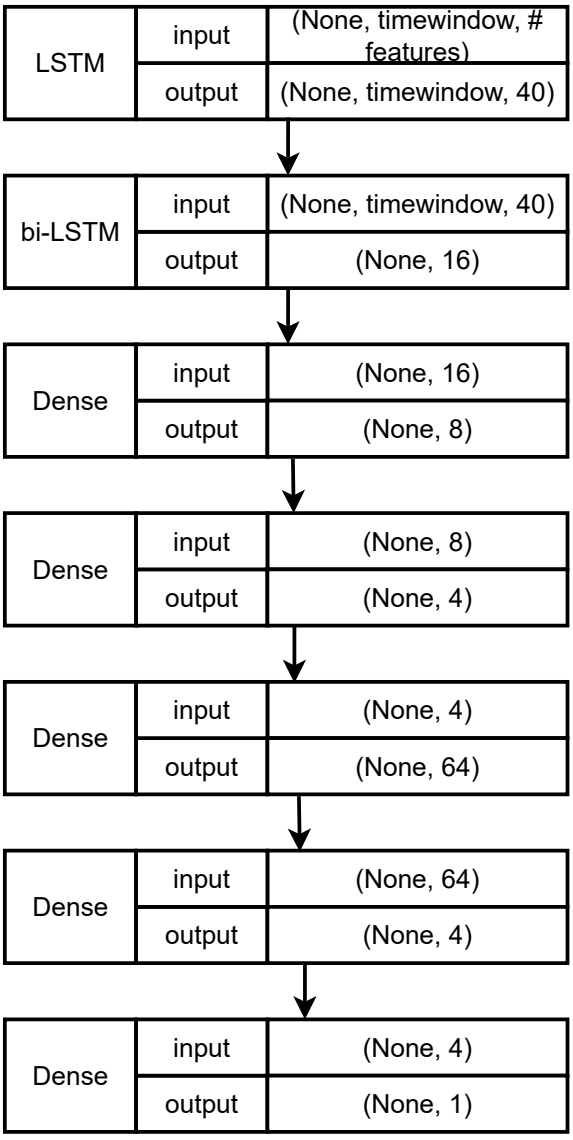


Figure 3: bi-LSTM Architecture.

This validation flow is performed for each patient separately, aiming in a personalized scheme, since this is a logical starting point for the specific task. Finally, we used Keras framework for the implementation of the mentioned architectures.

3.2 Deep Reinforcement Learning

3.2.1 Reinforcement Learning theoretical background

Reinforcement Learning is one of the three fields of machine learning, with the difference that the problem is formulated to an interactive communication between two entities: the Environment and the Agent. The formulation is usually made with Markov Decision Processes (MDPs). An MDP is a tuple of 4 elements (S, A, R, P), where S is the state space, A the actions space, R the reward function and P the transition probability matrix. The Agent receives the State (or Observation) from the Environment and selects an Action to perform which is communicated back to the Environment. The Environment executes the Agents Action and responds with the new State based on the transition probability matrix and a Reward that are both forwarded to the Agent to repeat the interaction. Based on the described interaction a sequence is constructed $(S_t, A_t, R_t, S_{t+1}, \dots)$ and a policy that the Agent follows, learns which Actions to performed based on the State in order to maximize the Reward. To learn from the sequence the Agent uses functions such as the Value function $V_\pi(S)$, which is the expected reward based on the current state and the Action-Value function $Q_\pi(s, a)$, which represents the expected rewarded based on the current state and selected action. In addition, there are many variations of all the above that construct different strategies in order to increase the Reward of the Agent in different problems [4, 5].

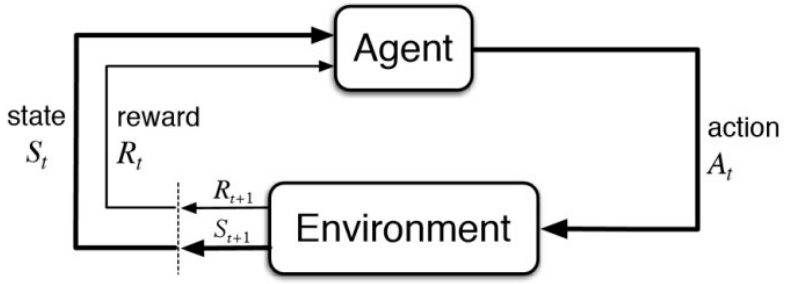


Figure 4: Reinforcement Learning illustration

3.2.2 Environment Implementation

For the implementation of the environment the OpenAI gym library was used to implement a custom environment. Based on the OpenAI gym, an environment must include at least 3 basic functions: 1)__init__ , 2) step, 3) reset. The first one is responsible of constructing the environment initializing all of its parameters, the second one executes the actions selected by the agent and returns the new state and reward, while the third one resets the environment to it's initial state [6].

To create the custom environment the input values ($3D$ array: Number of windows x Samples in each window x Number of features) and the target/prediction values ($2D$ array: Number of values x Number of features) must be provided for its initialization. Based on the above values the observation space and action space of the environment are created as continued float values, which define the observations and actions the Agent can receive and execute, respectively. Regarding the observation, for each window the $2D$ array of (Samples in each window x Number of features) is transformed to an $1D$ array by reshaping. Therefore, the observation space is the window constructed during the preprocessing of the data and the action space is the prediction the agent needs to make. For the step function, a pointer initialized during the construction of the environment is used to point the current state, in this case current window. When the agent selects an action to execute the pointer moves to the next window based on the input data, and the step function returns the new state to the agent with the generated reward. The reward is defined as the negative absolute value of the difference between the actions (predictions) and the expected (true) value ($-|a_t - t_t|$). Finally, the reset function resets the pointer to point at the first window of data and the state to the first window as well.

3.2.3 Deep RL for Timeseries Forecasting

With the above custom environment, a Deep Neural Network is used to represent the agent in order to solve the time series prediction problem. Specifically, the already implemented Deep Deterministic Policy Gradient (DDPG)[5, 7] algorithm is used which is characterized as a model-free and off-policy algorithm. The reason this algorithm was selected is mainly because the observation and action space of the custom environment are continued values. The DDPG algorithm uses the Actor-Critic method in order to implement the agent [8], by using two networks one for the Actor and one for the Critic. With a short description the Critic uses the Value function or Action-Value function mentioned before, and the Actor represents the policy. The Critic based on the rewards from the environment calculates the temporal difference (TD) error, which is then used to update the Critics function and the policy of the Actor.

Regarding the experiments performed with Deep RL, the training for each experiment consisted of a total of 200000 steps. Moreover as mentioned in [5], the same experiments were repeated by introducing noise (Ornstein-Uhlenbeck Process Action Noise) during training in order to help with the exploration of the custom environment from the agent.

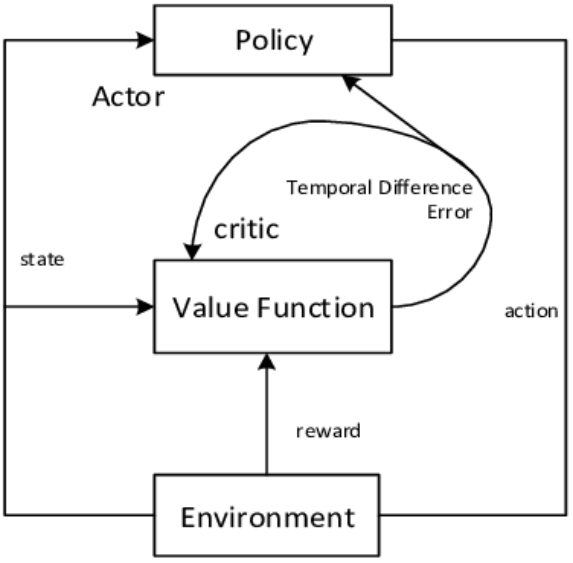


Figure 5: The framework of actor-critic method [8]

4 Results

4.1 Evaluation Criteria

For the evaluation of the prediction performance we used the root mean square error (RMSE, $[mg/dl]$), correlation coefficient (CC) and mean absolute relative difference (MARD).

RMSE indicates the difference between the target and the predicted data. This is calculated by taking the square root of the mean of the square of all the errors as shows in the equation bellow:

$$RMSE = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(y_k - \hat{y})^2} \tag{6}$$

MARD is the mean of the absolute error between the predicted values and the actual values (Equation 7). A lower MARD means better prediction performance.

$$MARD = \frac{1}{n}\Sigma_{k=1}^{n}(\frac{|y_k - \dot{y}_k|}{y_k} \tag{7}$$

CC represents the linear dependence between two datasets. For our work, CC could

be described with equation 8:

$$CC = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{8}$$

where y_k: the actual value, $\hat{y}$: the predicted value and n: the number of the predictions.

In our case to calculate the aforementioned metrics, we denormalized the test data before the calculation.

4.2 Experiments

4.2.1 LSTM

Tables 3 and 5 depict the results of our experiments for the personalized LSTM model for $ph = 30$ and $ph = 60$ minutes, respectively. With bold are shown the best results of each metric. We observe that the best performance for both $ph = 30$ and $ph = 60$ minutes is obtained by personID:540. The last row of the aforementioned tables presents the mean and the standard deviation of the total performance.

Table 4 demonstrates the results for the personalized bi-LSTM model for $ph = 30$ minutes. As the table shows the bi-LSTM model for the personID:552 was the best performing one with significant difference from a lot of the others. Moreover due to lack of time we could not perform the experiments for the bi-LSTM for the $ph = 60$.

ID	RMSE($mgdL^{-1}$)	MARD	CC
540	26.64	0.13	0.92
544	**16.81**	**0.08**	**0.94**
552	16.86	0.08	0.91
567	35.44	0.17	0.72
584	44.60	0.19	0.72
596	19.69	0.10	0.92
559	24.40	0.11	0.92
563	32.09	0.19	0.86
570	24.12	0.08	0.93
575	30.20	0.13	0.87
588	45.03	0.16	0.60
591	23.27	0.13	0.89
mean $\pm$ *std*	28.26 ± 9.59	0.13 ± 0.04	0.85 ± 0.12

Table 3: Results of the LSTM model for $ph = 30$ min.

ID	RMSE($mgdL^{-1}$)	MARD	CC
540	29.65	0.13	0.90
544	**18.26**	**0.08**	**0.93**
552	18.55	0.10	0.93
567	26.50	0.14	0.87
584	42.31	0.18	0.84
596	22.32	0.11	0.89
559	22.92	0.09	0.93
563	18.92	0.09	0.91
570	27.42	0.1	0.93
575	28.83	0.13	0.89
588	23.42	0.10	0.87
591	24.20	0.13	0.89
mean $\pm$ *std*	25.25 ± 6.55	0.12 ± 0.03	0.90 ± 0.03

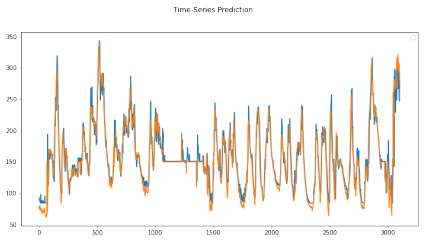
Table 4: Results of the bi-LSTM model for $ph = 30$ min.

ID	RMSE($mgdL^{-1}$)	MARD	CC
540	46.51	0.22	0.71
544	26.93	0.13	0.85
552	**25.31**	**0.13**	**0.79**
567	37.46	0.19	0.69
584	84.98	0.31	0.55
596	30.99	0.16	0.77
559	54.79	0.25	0.52
563	43.71	0.25	0.61
570	41.30	0.16	0.79
575	46.36	0.25	0.69
588	62.63	0.21	0.41
591	40.73	0.27	0.68
$mean \pm std$	45.14 ± 16.54	0.21 ± 0.06	0.67 ± 0.13

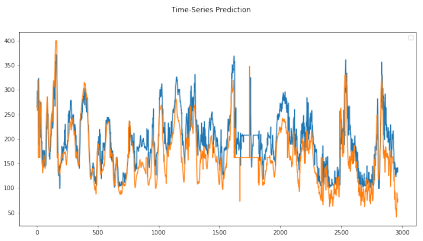
Table 5: Results of the LSTM model for $ph = 60$ min.

From the tables above we observe that between the two LSTM models implemented the bi-LSTM model performed better than the LSTM model, at least for the $ph = 30$.

The bellow Figure 6 shows the true values (blue) and the predictions (orange) as a timeseries graph. Both graphs represent the experimenters contacted for $ph = 30$ by the bi-LSTM model, for the best performing personID:544 (left graph) and the worst performing personID:584 (right graph). The worst performing person annotated to the tables with red color. Even though in both cases the predictions seems to follow the trend of the true values, for the personID:544 the predictions are very close to the true values, while for the personID:584 the predictions are slightly lower than the true values throughout the figure.



(a) PersonID:544, with best performance for $ph = 30$ min.



(b) PersonID:584, with worst performance for $ph = 30$ min.

Figure 6: Best and worst performance of bi-LSTM model for $ph = 30$ min. With the orange is the actual values and blue the prediction of bg.

4.2.2 Deep RL

As mentioned in Section 3.2 four experiments conducted for each person by changing the predictions horizon and introducing noise during the training. Specifically, as with the LSTM, two predictions horizon were tested $ph = 30$ and $ph = 60$ minutes and the Ornstein Uhlenbeck Action Noise was introduced. The aforementioned noise derives from the Ornstein–Uhlenbeck process:

$$d_{x_t} = \theta(\mu - x_t)d_t + \sigma dW_t \tag{9}$$

From the above equation the θ and μ represent constants, σ the variance (scale) of the noise, dW_t the Weiner process and for the RL setting the x_t denotes to the state at time t.

The Tables 6 and 8 display the results for the experiments without noise for $ph = 30$ min and $ph = 60$ min, respectively. With bold are shown the best results for each patient, while with red the worst performing. The best performance in both cases is observed for personID:544.

The Tables 7 and 9 illustrates the results for the experiments with noise for $ph = 30$ min and $ph = 60$ min, respectively. Similarly, the best performance for the $ph = 30$ min is observed to the personID:544, while the personID:552 has the best performance for the $ph = 60$ min.

ID	RMSE($mgdL^{-1}$)	MARD	CC
540	41.13	0.24	0.80
544	**22.47**	**0.11**	**0.90**
552	25.03	0.13	0.83
567	31.80	0.18	0.80
584	50.51	0.24	0.70
596	23.95	0.14	0.87
559	33.32	0.17	0.86
563	43.38	0.28	0.72
570	39.34	0.16	0.83
575	38.23	0.21	0.79
588	38.90	0.16	0.73
591	33.80	0.24	0.78
mean $\pm$ *std*	34.25 ± 10.07	0.19 ± 0.05	0.80 ± 0.06

Table 6: Results of the deep RL model for $ph = 30$ min.

ID	RMSE($mgdL^{-1}$)	MARD	CC
540	27.72	0.14	0.91
544	**17.97**	**0.08**	**0.94**
552	18.27	0.09	0.90
567	25.44	0.13	0.86
584	35.04	0.16	0.81
596	20.66	0.11	0.91
559	24.14	0.11	0.93
563	24.80	0.13	0.87
570	33.15	0.13	0.89
575	29.24	0.14	0.88
588	26.67	0.11	0.84
591	24.25	0.14	0.89
mean $\pm$ *std*	25.61 ± 5.27	0.12 ± 0.02	0.89 ± 0.04

Table 7: Results of the deep RL model with noise for $ph = 30$ min.

ID	RMSE($mgdL^{-1}$)	MARD	CC
540	70.14	0.87	0.54
544	**29.81**	**0.16**	**82**
552	36.64	0.20	0.65
567	49.76	0.36	0.60
584	90.59	0.35	0.57
596	42.39	0.27	0.68
559	62.19	0.43	0.59
563	60.30	0.31	0.59
570	59.61	0.22	0.70
575	55.56	0.37	0.61
588	44.46	0.21	0.57
591	48.70	0.37	0.60
mean $\pm$ *std*	54.18 ± 16.23	0.34 ± 0.19	0.63 ± 0.08

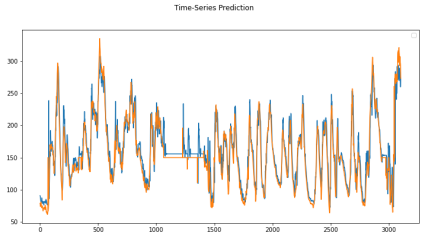
Table 8: Results of the deep RL model for $ph = 60$ min.

ID	RMSE($mgdL^{-1}$)	MARD	CC
540	43.34	0.22	0.76
544	27.44	0.14	0.86
552	**26.33**	**0.13**	**0.77**
567	40.27	0.23	0.66
584	76.35	0.29	0.63
596	33.59	0.18	0.78
559	35.59	0.16	0.82
563	47.14	0.20	0.65
570	37.75	0.15	0.83
575	44.23	0.24	0.71
588	49.00	0.17	0.61
591	35.99	0.22	0.75
mean $\pm$ *std*	41.25 ± 13.18	0.19 ± 0.05	0.74 ± 0.08

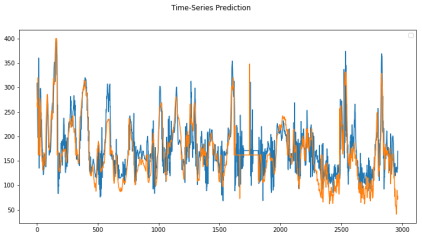
Table 9: Results of the deep RL model with noise for $ph = 60$ min.

Generally, we observe an increased performance in the deep RL models trained with noise. Comparing the average results of the experiments with and without noise, the former shows a significant decrease of the $RMSE$ by 8.16, 0.07 for the $MARD$ and an increase for the CC by 0.09. Finally the best performance from all the experiments for the deep RL architecture was observed for the personID:544 in the experiment with $ph = 30$ min and noise, resulting an $RMSE = 17.97$, $MARD = 0.08$ and $CC = 0.94$.

Figure 7 demonstrates the timeseries prediction in the test set in deep RL model, with $ph = 30$ min, for PersonID:544, who had the best performance and for personID:584 who had the worst performance. The actual values of bg are represented with the orange color while with blue the prediction for the bg. As shown in the graphs for the personID:544 the predicted values seem to be quite accurately close to the true values and following the trends, while for the other person, personID 544, the predictions seem to diverge from true values even though in most cases they follow their trends.



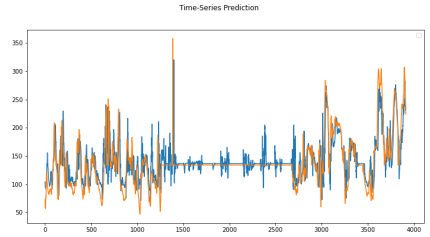
(a) PersonID:544, with best performance for $ph = 30$ min.



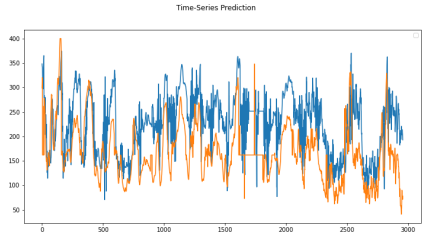
(b) PersonID:584, with worst performance for $ph = 30$ min.

Figure 7: Best and worst performance of RL with noise model for $ph = 30$ min. With the orange is the actual values of bg and blue the prediction

Figure 8 display the timeseries prediction in the test set in deep RL model, with $ph = 60$ min, for personID:552, who had the best performance and for personID:584 who had the worst performance. As before the actual values of bg are represented with the orange color while with blue the prediction for the bg. From the graphs we can clearly observe the difference between the two persons, as the predictions for personID:552 are more close to the true values than the predictions for the personID:584. In the case of the personID:552 the prediction values are very close to the true values, following the trends of the timeseries, while for the personID:584 the predictions seem to be lower than the true values in almost all the cases.



(a) PersonID:552, with best performance for $ph = 60$ min.



(b) PersonID:584, with worst performance for $ph = 60$ min.

Figure 8: Best and worst performance of deep RL with noise model for $ph = 60$ min. With the orange is the actual values of bg and blue the prediction

Finally as expected the performance of all the models for the $ph = 30$ is better than the $ph = 60$ for all the persons.

4.3 Comparison with other NN models

Table 10 and 11 show the results of different architectures [9] over the prediction horizons of 30 and 60 minutes, respectively for the same set of data. The results are the average of the metrics over the patients with the standard error of the mean deviation.

In the first case, with $ph = 30$ min, our models performed worse than the other models, with significant difference from the state of the art model in Sun et.al paper [10]. The worst performing from our models is the LSTM, with the other two models, bi-LSTM and deep RL, having almost the same performance.

ID	RMSE($mgdL^{-1}$)	CC	MARD
Mirshekarian [11]	21.34 ± 1.71	0.94 ± 0.01	0.10 ± 0.01
Meijner [12]	19.92 ± 1.35	0.95 ± 0.02	0.10 ± 0.01
Gülesir [13]	22.18 ± 1.32	0.93 ± 0.01	0.11 ± 0.01
Sun [10]	$\mathbf{19.73 \pm 1.31}$	$\mathbf{0.95 \pm 0.01}$	$\mathbf{0.10 \pm 0.01}$
Idriss [14]	23.57 ± 1.98	0.93 ± 0.01	0.12 ± 0.01
Aiello [15]	22.64 ± 1.76	0.93 ± 0.01	0.11 ± 0.01
Zhu [16]	21.74 ± 1.45	0.94 ± 0.01	0.11 ± 0.01
Mayo [17]	22.35 ± 2.48	0.93 ± 0.01	0.10 ± 0.01
Muñoz [18]	21.22 ± 1.39	0.94 ± 0.01	0.12 ± 0.01
Khadem [19]	21.80 ± 1.56	0.94 ± 0.01	0.11 ± 0.01
Our LSTM	28.26 ± 9.59	0.85 ± 0.12	0.13 ± 0.04
Our bi-LSTM	25.25 ± 6.55	0.90 ± 0.03	0.12 ± 0.03
Our RL	25.61 ± 5.27	0.89 ± 0.04	0.12 ± 0.02

Table 10: Results of other NN models for each performance metric for $ph = 30$ min.

In the second case, with $ph = 60$, we observe a significant improvement of our models compared to the other models, but with significant difference from the state of the art model in Sun et.al paper [10]. Again our LSTM is the worst performing model, while the RL model has a better performance and out performing two of the models shown in the table 11.

ID	RMSE($mgdL^{-1}$)	CC	MARD
Mirshekarian [11]	38.58 ± 2.80	0.79 ± 0.03	0.20 ± 0.01
Meijner [12]	36.55 ± 2.54	0.81 ± 0.02	0.19 ± 0.01
Gülesir [13]	37.25 ± 2.42	0.80 ± 0.02	0.20 ± 0.01
Sun [10]	$\mathbf{34.48 \pm 2.12}$	$\mathbf{0.83 \pm 0.02}$	$\mathbf{0.19 \pm 0.01}$
Idriss [14]	43.88 ± 3.59	0.72 ± 0.03	0.22 ± 0.01
Aiello [15]	44.47 ± 3.18	0.72 ± 0.02	0.23 ± 0.01
Zhu [16]	35.33 ± 2.45	0.83 ± 0.02	0.18 ± 0.01
Mayo [17]	40.71 ± 4.06	0.76 ± 0.03	0.20 ± 0.02
Muñoz [18]	40.69 ± 1.86	0.82 ± 0.02	0.26 ± 0.01
Khadem [19]	37.08 ± 2.25	0.80 ± 0.02	0.21 ± 0.01
Our LSTM	45.14 ± 16.54	0.67 ± 0.13	0.21 ± 0.06
Our RL	41.25 ± 13.18	0.74 ± 0.08	0.19 ± 0.05

Table 11: Results of other NN models for each performance metric for $ph = 60$ min.

Overall, the bi-LSTM model has the best performance, followed by the deep RL model, which has slightly lower performance.

5 Conclusion and Future Thoughts

In this work, three architectures are implemented for blood glucose prediction, using the Ohio Dataset, where we decided to keep only the features of cbg, basal and bolus insulin and ch intakes. We provide promising results especially in the deep RL model which is something new. On the contrary with the state-of-art architecture (Sun et.al [10]), our models have slightly lower performance but more experiments and parameterization could results better performance. Concluding, in future work using fine tuning and more complicated architectures such as transformer and other deep RL model, we could find a model which could overcome the present state of the art architecture in this application.

References

[1] K. Zarkogianni, "Intelligent personalized medical decision support systems for the management of diabetes mellitus," 2011.

[2] C. Marling and R. Bunescu, "The ohiot1dm dataset for blood glucose level prediction: Update 2020," in *CEUR workshop proceedings*, vol. 2675, p. 71, NIH Public Access, 2020.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[7] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on Machine Learning* (E. P. Xing and T. Jebara, eds.), vol. 32 of *Proceedings of Machine Learning Research*, (Bejing, China), pp. 387–395, PMLR, 22–24 Jun 2014.

[8] H. Wang, P. Zhang, and Q. Liu, "An actor-critic algorithm using cross evaluation of value functions," *IAES International Journal of Robotics and Automation (IJRA)*, vol. 7, p. 39, 03 2018.

[9] F. Tena, O. Garnica, J. Lanchares, and J. I. Hidalgo, "A critical review of the state-of-the-art on deep neural networks for blood glucose prediction in patients with diabetes," *arXiv preprint arXiv:2109.02178*, 2021.

[10] Q. Sun, M. V. Jankovic, L. Bally, and S. G. Mougiakakou, "Predicting blood glucose with an lstm and bi-lstm based deep neural network," in *2018 14th symposium on neural networks and applications (NEUREL)*, pp. 1–5, IEEE, 2018.

[11] S. Mirshekarian, R. Bunescu, C. Marling, and F. Schwartz, "Using lstms to learn physiological models of blood glucose behavior," in *2017 39th Annual*

International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 2887–2891, IEEE, 2017.

[12] C. Meijner and S. Persson, "Blood glucose prediction for type 1 diabetes using machine learning long short-term memory based models for blood glucose prediction," Master's thesis, 2017.

[13] M. von Gizem Gülesir, T. der Einreichung, S. Kauschke, and G. Gülesir, "Predicting blood glucose levels of diabetes patients," 2018.

[14] T. El Idriss, A. Idri, I. Abnane, and Z. Bakkoury, "Predicting blood glucose using an lstm neural network," in *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 35–41, IEEE, 2019.

[15] E. M. Aiello, G. Lisanti, L. Magni, M. Musci, and C. Toffanin, "Therapy-driven deep glucose forecasting," *Engineering Applications of Artificial Intelligence*, vol. 87, p. 103255, 2020.

[16] T. Zhu, K. Li, J. Chen, P. Herrero, and P. Georgiou, "Dilated recurrent neural networks for glucose forecasting in type 1 diabetes," *Journal of Healthcare Informatics Research*, vol. 4, no. 3, pp. 308–324, 2020.

[17] M. Mayo and T. Koutny, "Neural multi-class classification approach to blood glucose level forecasting with prediction uncertainty visualisation," in *KDH 2020*, vol. 2675, pp. 80–84, CEUR Workshop Proceedings, 2020.

[18] M. Munoz-Organero, "Deep physiological model for blood glucose prediction in t1dm patients," *Sensors*, vol. 20, no. 14, p. 3896, 2020.

[19] H. Khadem, H. Nemat, J. Elliott, and M. Benaissa, "Multi-lag stacking for blood glucose level prediction," in *Knowledge Discovery in Healthcare Data 2020*, vol. 2675, pp. 146–150, CEUR-Workshop Proceedings, 2020.