Abstract—Control of blood glucose is essential for diabetes management. Current digital therapeutic approaches for subjects with Type 1 diabetes mellitus (T1DM) such as the artificial pancreas and insulin bolus calculators leverage machine learning techniques for predicting subcutaneous glucose for improved control. Deep learning has recently been applied in healthcare and medical research to achieve state-of-the-art results in a range of tasks including disease diagnosis, and patient state prediction among others. In this work, we present a deep learning model of tasks including disease diagnosis, and patient state prediction and medical research to achieve state-of-the-art results in a range of control. Deep learning has recently been applied in healthcare techniques for predicting subcutaneous glucose for improved pancreas and insulin bolus calculators leverage machine learning with Type 1 diabetes mellitus (T1DM) such as the artificial management. Current digital therapeutic approaches for subjects in forecasting adverse glycaemic events with minimal time lag.

In addition, the model also provides competitive performance in forecasting adverse glycaemic events with minimal time lag both in a simulated patient dataset (MCC$_{hypo}$ = 0.83±0.05 and MCC$_{hyper}$ = 0.80±0.10) and in a real patient dataset (MCC$_{hypo}$ = 0.79±0.04 and MCC$_{hyper}$ = 0.38±0.10). This approach is evaluated on a dataset of 10 simulated cases generated from the UVa/Padova simulator and a clinical dataset of 5 real cases each containing glucose readings, insulin bolus, and meal (carbohydrate) data. Performance of the recurrent convolutional neural network is benchmarked against four algorithms. The prediction algorithm is implemented on an Android mobile phone, with an execution time of 6ms on a phone compared to an execution time of 780ms on a laptop in Python.

Index Terms—Type 1 diabetes, continuous glucose monitor (CGM), glucose prediction, deep learning, long short term memory (LSTM).

I. INTRODUCTION

Diabetes is a chronic illness characterised by a lack of glucose homeostasis. A healthy pancreas dynamically controls the release of insulin and glucagon hormones through the α-cells and β-cells respectively, in order to maintain euglycaemia [1]–[3]. In Type 1 diabetes, an autoimmune disease, the β-cells are compromised and therefore suffer from impaired production of insulin. This leads to increased periods of hyperglycaemia (blood glucose concentration > 180mg/dL) and hypoglycaemia (blood glucose concentration < 70mg/dL) [4], [5]. Insulin therapy is needed to manage blood glucose (BG) levels to maximise the time in (advised) target [6], [7].

The standard approach to diabetes management requires the subject to actively undertake blood glucose measurements a handful of times throughout the day with a finger prick test - self monitoring of blood glucose (SMBG). The recent development and uptake of continuous glucose monitoring (CGM) devices allow for improved sampling (5 minutes) of glucose measurements. This approach has proven to be effective in enabling glucose control and thus improving the outcome of subjects in clinical trials relative to standard approaches [6]–[8]. Further improvement of glucose control can be realised through prediction, allowing users to take action ahead of time to minimise the occurrence of adverse glycaemic events. The challenge lies in the multiple factors such as insulin variability, ingested meals, stress and other physical activities that influence glucose variability [9]. In addition, the glycaemic responses of different people to the same conditions can be very different, which is conditioned by high subject variability.

Machine learning covers a range of techniques that allow intelligent systems to tackle problems by learning and extracting patterns in data to build appropriate models. Consequently, the techniques involve discovering a suitable mapping from the representation of input data to the output. The performance of traditional machine learning algorithms such as logistic regression, k-nearest neighbours [10], or support vector regression [11] heavily depend on the representation of the data they are given. Typically, the features - information the representation comprises - are engineered with prior knowledge and statistical features (mean, variance) [12], principal component analysis (PCA) [13] or linear discriminant analysis [14]. Artificial neural networks are also investigated widely in diabetes management [15]–[19]. However, convolutional neural networks are mostly used with less than 3 layers, which limits the power of neural networks with current training techniques.

Deep learning offers an approach to automatically learn both the features and consequent nonlinear mapping to the output. This approach, which incorporates multi-layer neural networks has been effective in solutions that have improved on previous upper bounds of performance in computer vision [20], board games [21], diseases diagnosis [22], [23], and healthcare [24], [25]. Deep learning typically shows superior performance to traditional techniques due to this ability to automatically learn features with physical meanings [27], [28], hence potentially encoding features that may not be previously known to researchers.

In this paper, we propose a deep learning algorithm for glucose prediction using a multi-layer convolutional recurrent neural network (CRNN) architecture. The model is primarily trained on data comprising CGM, carbohydrate and insulin data. After preprocessing, the time-aligned multi-dimensional time series data of BG, carbohydrate and insulin (other factors are optional) are fed to the deep neural network (DNN) for training. The architecture of the DNN is composed of three parts: a multi-layer convolutional neural network that extracts...
the data features using convolution and pooling, followed by a recurrent neural network (RNN) layer with long short term memory (LSTM) cells and fully-connected layers. In the LSTM cell, an input gate, an output gate and a forget gate are incorporated to remember the previous model state and update the model state simultaneously when current data are fed seamlessly. The convolutional layer comprises a 1D Gaussian kernel filter to perform the temporal convolution, and pooling layers are used for reducing the feature set. LSTM are the variant of gated RNN used since it shows good performance in predicting time series given long time dependencies [29]. In addition, it deals well with the vanishing gradient problem for training traditional RNNs. The final output is a regression output by the fully connected layer. The DNN is realized using the open-source software library Tensorflow [30], and it can be easily implemented to portable or wearable devices with its simplified version Tensorflow Lite. The performance of the proposed method is evaluated on datasets of simulated cases as well as clinical cases of T1DM subjects, and compared against the latest algorithms including support vector regression (SVR) [11], the latent variable model (LVX) [31], the autoregressive model (ARX) [32], and neural network for predicting glucose algorithm (NNPG) [15], which show the competitive performance of the method.

The paper is organized as follows. Section II briefly discusses the setup and specification of the system, the pipeline for preprocessing the data. Section III details the architecture of the proposed multi-layer convolutional recurrent neural network. The performance of the proposed method is evaluated and discussed in Section IV. Finally, Section V concludes the paper with a discussion of challenges in this approach and avenues for future research.

II. METHODS

After introducing the data acquisition and hardware platform, the method of the glucose prediction is explained. The approach consists of several components: preprocessing, feature extraction using CNN, time series prediction using LSTM and signal converter. The architecture of the proposed CRNN is shown in Fig. 2. In the diagram, the input of the algorithm is time series of glycaemic data from CGM, carbohydrate and insulin information (time and amount); other related information are optional (exercises, alcohol, fat, stress, etc.). The output of preprocessing is cleaned as time-aligned glycaemic, carbohydrate and insulin data, which are then fed to the CNN. The output of CNN serves as the input of RNN, which is a multi-dimensional time series data, representing the concatenation of features of the original signals. The output of the RNN is the predictive blood glucose level 30-min later, while hidden states are inherited and updated continuously internally inside of the RNN component. We evaluate the models with 30-min prediction horizon (PH) because it is widely used in glucose prediction software, and is easier to compare results with other works [11], [15], [17], [31], [33], [34]. We proceed with an explanation data pipeline and components of the model architecture.

A. Data Acquisition and Implementation

The data used in this paper include two datasets, simulated data and clinical data. Simulated data of T1DM subjects was generated using the UVA/Padova T1D, which is the only simulator for glucose level simulation approved by the Food and Drug Administration (FDA). Clinical data was obtained from a 6-month clinical dataset corresponding to 10 adult subjects with T1D undergoing a clinical trial evaluating the benefits of an advanced insulin bolus calculator was employed [35]. The information included in the dataset needs glucose, meal, insulin and associated time stamps. Other information is optional for the method.

After the model has successfully undergone training and validation, we also implement our algorithm on mobile phones through Tensorflow Lite due to its efficiency running on portable devices. The model is converted to a Lite model file and installed on an Android or iOS system. It needs the associated application programming interface (API) and interpreter to carry out the inference. Fig. 1 shows how the Tensorflow Lite model file [30] is wrapped and loaded in a mobile-friendly format.

B. Outlier Detection and Filtering

In the preprocessing component, the main purpose is to clean the data, filter the unusual points and make it suitable as the input to the CNN. Besides the normal steps including time stamp alignment, normalization and missing data imputation for time series data with trend, the most important operation to improve the data quality is the outlier detection, interpolation and filtering, in particular for clinic data. Because in the clinical data of glucose time series, there are many missing or outlier data points due to errors in calibration, measurements, and/or mistakes in the process of data collection and transmission. Here, several methods are introduced to handle these scenarios [35].

- Dimension Reduction Model: the time series can be projected into lower dimensions using linear correlations such as principle component analysis (PCA) [37], and
Multi-temporal convolution definition in which a set of feature vectors. The convolution operation follows the layer convolutions, which transforms the input data into a representation, thus the amount of parameters as well as the computation complexity. For instance, if the accepted size is $L1 \times D1$, and the down-sampled parameters are spatial extent $F$ and stride $S$, and it results in a max-pooling vector $Y$ of size $L2 \times D2$ where

$$L2 = (L1 - F)/S + 1$$
$$D2 = D1$$

where $z[m]$ is the result of the convolution. The input signal can be fed block by block, using a sliding window setting. The windows can be overlapped or non-overlapped, determined by the allowed CNN size and computations. The CNN automatically learns the associated weights and recognises particular patterns and features in the input signal that can best represent the data for future time steps, using the back-propagation technique.

The proposed CNN has 2 hidden layers, consisting 2 convolutional layers, with max pooling applied to down-sample the feature map obtained from the previous convolutional layer. It is common to periodically insert a pooling layer in-between successive convolutional layers to progressively reduce the size of the representation, thus the amount of parameters as well as the computation in the CNN. It also guards against the problem of overfitting. For instance, if the accepted size is $L1 \times D1$, and the down-sampled parameters are spatial extent $F$ and stride $S$, and it results in a max-pooling vector $Y$ of size $L2 \times D2$ where

$$L2 = (L1 - F)/S + 1$$
$$D2 = D1$$

where $z[m]$ is the result of the convolution. The input signal can be fed block by block, using a sliding window setting. The windows can be overlapped or non-overlapped, determined by the allowed CNN size and computations. The CNN automatically learns the associated weights and recognises particular patterns and features in the input signal that can best represent the data for future time steps, using the back-propagation technique.

The proposed CNN has 2 hidden layers, consisting 2 convolutional layers, with max pooling applied to down-sample the feature map obtained from the previous convolutional layer. It is common to periodically insert a pooling layer in-between successive convolutional layers to progressively reduce the size of the representation, thus the amount of parameters as well as the computation in the CNN. It also guards against the problem of overfitting. For instance, if the accepted size is $L1 \times D1$, and the down-sampled parameters are spatial extent $F$ and stride $S$, and it results in a max-pooling vector $Y$ of size $L2 \times D2$ where

$$L2 = (L1 - F)/S + 1$$

where $z[m]$ is the result of the convolution. The input signal can be fed block by block, using a sliding window setting. The windows can be overlapped or non-overlapped, determined by the allowed CNN size and computations. The CNN automatically learns the associated weights and recognises particular patterns and features in the input signal that can best represent the data for future time steps, using the back-propagation technique.

The proposed CNN has 2 hidden layers, consisting 2 convolutional layers, with max pooling applied to down-sample the feature map obtained from the previous convolutional layer. It is common to periodically insert a pooling layer in-between successive convolutional layers to progressively reduce the size of the representation, thus the amount of parameters as well as the computation in the CNN. It also guards against the problem of overfitting. For instance, if the accepted size is $L1 \times D1$, and the down-sampled parameters are spatial extent $F$ and stride $S$, and it results in a max-pooling vector $Y$ of size $L2 \times D2$ where

$$L2 = (L1 - F)/S + 1$$
$$D2 = D1$$

where $z[m]$ is the result of the convolution. The input signal can be fed block by block, using a sliding window setting. The windows can be overlapped or non-overlapped, determined by the allowed CNN size and computations. The CNN automatically learns the associated weights and recognises particular patterns and features in the input signal that can best represent the data for future time steps, using the back-propagation technique.

For some cases when the data fluctuates with high frequency, 1D Gaussian kernel filter is implemented on the glucose time series to get rid of the noise. The result of this process is a smoothed, continuous time series of glycaemic data from the CGM, as well as time-aligned carbohydrate and insulin information. We feed the model sample points that from this 3-dimensional data that cover the last 24 hours before the current time in the time series. A sliding window of size 24 is therefore determined to train the model, because we found it is an appropriate setting to balance the tradeoff between the prediction accuracy and the computation complexity.

C. A Multi-layer Convolutional Network

The filtered time series signal goes through the multi-layer convolutions, which transforms the input data into a set of feature vectors. The convolution operation follows the temporal convolution definition in which

$$z[m] = \sum_{i=-L}^{L} x[i] \cdot \delta[m-i],$$

where $x$ represents the input signal, $\delta$ denotes the kernel, and $z$ is the result of the convolution. The input signal can be fed block by block, using a sliding window setting. The windows can be overlapped or non-overlapped, determined by the allowed CNN size and computations. The CNN automatically learns the associated weights and recognises particular patterns and features in the input signal that can best represent the data for future time steps, using the back-propagation technique.

For some cases when the data fluctuates with high frequency, 1D Gaussian kernel filter is implemented on the glucose time series to get rid of the noise. The result of this process is a smoothed, continuous time series of glycaemic data from the CGM, as well as time-aligned carbohydrate and insulin information. We feed the model sample points that from this 3-dimensional data that cover the last 24 hours before the current time in the time series. A sliding window of size 24 is therefore determined to train the model, because we found it is an appropriate setting to balance the tradeoff between the prediction accuracy and the computation complexity.

where $x$ represents the input signal, $\delta$ denotes the kernel, and $z$ is the result of the convolution. The input signal can be fed block by block, using a sliding window setting. The windows can be overlapped or non-overlapped, determined by the allowed CNN size and computations. The CNN automatically learns the associated weights and recognises particular patterns and features in the input signal that can best represent the data for future time steps, using the back-propagation technique.

The proposed CNN has 2 hidden layers, consisting 2 convolutional layers, with max pooling applied to down-sample the feature map obtained from the previous convolutional layer. It is common to periodically insert a pooling layer in-between successive convolutional layers to progressively reduce the size of the representation, thus the amount of parameters as well as the computation in the CNN. It also guards against the problem of overfitting. For instance, if the accepted size is $L1 \times D1$, and the down-sampled parameters are spatial extent $F$ and stride $S$, and it results in a max-pooling vector $Y$ of size $L2 \times D2$ where

$$L2 = (L1 - F)/S + 1$$
$$D2 = D1$$

where $z[m]$ is the result of the convolution. The input signal can be fed block by block, using a sliding window setting. The windows can be overlapped or non-overlapped, determined by the allowed CNN size and computations. The CNN automatically learns the associated weights and recognises particular patterns and features in the input signal that can best represent the data for future time steps, using the back-propagation technique.

For some cases when the data fluctuates with high frequency, 1D Gaussian kernel filter is implemented on the glucose time series to get rid of the noise. The result of this process is a smoothed, continuous time series of glycaemic data from the CGM, as well as time-aligned carbohydrate and insulin information. We feed the model sample points that from this 3-dimensional data that cover the last 24 hours before the current time in the time series. A sliding window of size 24 is therefore determined to train the model, because we found it is an appropriate setting to balance the tradeoff between the prediction accuracy and the computation complexity.

where $x$ represents the input signal, $\delta$ denotes the kernel, and $z$ is the result of the convolution. The input signal can be fed block by block, using a sliding window setting. The windows can be overlapped or non-overlapped, determined by the allowed CNN size and computations. The CNN automatically learns the associated weights and recognises particular patterns and features in the input signal that can best represent the data for future time steps, using the back-propagation technique.

The proposed CNN has 2 hidden layers, consisting 2 convolutional layers, with max pooling applied to down-sample the feature map obtained from the previous convolutional layer. It is common to periodically insert a pooling layer in-between successive convolutional layers to progressively reduce the size of the representation, thus the amount of parameters as well as the computation in the CNN. It also guards against the problem of overfitting. For instance, if the accepted size is $L1 \times D1$, and the down-sampled parameters are spatial extent $F$ and stride $S$, and it results in a max-pooling vector $Y$ of size $L2 \times D2$ where

$$L2 = (L1 - F)/S + 1$$
$$D2 = D1$$

where $z[m]$ is the result of the convolution. The input signal can be fed block by block, using a sliding window setting. The windows can be overlapped or non-overlapped, determined by the allowed CNN size and computations. The CNN automatically learns the associated weights and recognises particular patterns and features in the input signal that can best represent the data for future time steps, using the back-propagation technique.
The main difference between a normal LSTM and the proposed model is that the proposed model has the transform and recover steps to modify the BG values before going through and after coming out the conventional LSTM model.

Because in training, instead of using the BG values directly, we use the change of BG between the current BG $x(t)$ and the future BG $x(t+6)$ as the target label. It is called the transform step. In the training, the input sliding window matrices are multi-dimensional time series (including BG values, meal, insulin). After the model has been trained, the inference output is the change of BG $\Delta x(t)$ between $x(t)$ and $x(t+6)$. So we need the BG value $x(t)$ to obtain the prediction of BG at time $t+6$ with $x(t+6) = x(t) + \Delta x(t)$. It is called the recover step.

Specifically, a modified LSTM RNN estimates the conditional probability $p(y_1, \ldots, y_T|x_1, \ldots, x_T)$ given a sequence of data, where $x_1, \ldots, x_T$ denotes the input sequence and $y_1, \ldots, y_T$ is the corresponding output sequence, with size $T$ and $T'$ respectively. If $x_t, h_t, c_t$ are used to denote the input vector, output vector and memory cell vector respectively; $W, U,$ and $b$ are the parameter matrices/vectors that can be learned in the network. $f_t, i_t,$ and $o_t$ denote the forget gate, input gate, and output gate vectors. Then mathematical form of the update process can be explicitly written as:

$$
\begin{align*}
 f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \quad i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
 o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \quad g_t = \sigma_t(W_g x_t + U_g h_{t-1} + b_g) \\
 c_t &= f_t \circ c_{t-1} + i_t \circ g_t \quad h_t = o_t \circ \sigma_t(c_t), \\
 y_t &= o_t \circ \sigma_t(c_t) + x_t,
\end{align*}
$$

where $\sigma_g, \sigma_t, \circ$ is the sigmoid function, hyperbolic tangent and entrywise product, respectively. In [3], the 1st to 5th equations are the same to the equations of normal LSTM. However the last 2 equations of $h_t$ and $y_t$ are modified according to the proposed glucose model. It has been shown in Figure [3]. In the figure, the difference between the proposed modified LSTM and the normal LSTM is indicated using the red colour. The output of the LSTM cell is not $h_t$, but $y_t$. The $h_t$ is used as a remembered state that goes to the next time stage. However the output $y_t$ is calculated from $h_t$ plus the original input $x_t$. 

\[ Y_t = Y_t - 1 \]

The output of the CNN, a 11-dimensional time series, is used to train the LSTM network. We generated a RNN as 1 hidden layer, consisting of a wide LSTM layer. The LSTM layer has 64 cells, with a dropout applied in this layer. Dropout refers to ignoring neurons randomly during the training phase of certain set of neurons. In many cases it has been verified that dropout can effectively avoid the overfitting problem and improve generalisation [21], [22].

The main difference between a normal LSTM and the proposed LSTM is that the proposed model has the transform and recover steps to modify the BG values before going through and after coming out the conventional LSTM model.
That is because the output (and targets) of the neural network is the change of the blood glucose level. The real value of the predictive glucose level needs to be recovered from the glucose change by adding the baseline glucose value.

Finally, the last layer of RNN feeds a multi-layer fully connected network, which consists of 2 hidden layers (256 neurons and 32 neurons) and an output layer (a single neuron) with the glucose change as output. The fully connected layer produces the output with an activation function

\[
Z_i = act(\sum_{i=1}^{N} Y_i w_i + b_i),
\]

where \(Z_i\) is the multi-dimensional output, \(act()\) is an activation function, \(w_i\) and \(b_i\) are weights of the fully connected network that can be trained. Particularly, \(act()\) can be chosen from a set of activation functions such as sigmoid function \(act(a) = 1/(1 + e^{-a})\), rectifier \(act(a) = \log(1 + \exp(a))\) or simple linearly \(act(a) = a\). In this paper we choose the simple linear function \(act(a) = a\) as the activation function for its simplicity.

In the training, mean-absolute-error between the target and the predictive value are being minimised. The optimiser we use is RMSprop, because it is usually a good choice for recurrent neural networks. It maintains a moving (discounted) average of the square of gradients, and divides gradient by the root of this average.

III. RESULTS

In this section we test the proposed CRNN algorithm for glucose level prediction using both in silico dataset and a clinical dataset, and compare with state-of-the-art algorithms developed in recent years.

A. Verification Set-up

For all methods, the datasets include CGM data recorded every 5 minutes, meal data indicating meal time and amount of carbohydrates, as well as insulin data with each bolus quantity and the associated time. Exercise, stress and alcohol consumption were included in the datasets but as optional variables in the model. In the data preprocessing step, incidences of meal and insulin intake are aligned to CGM data with the same timestamps.

The in silico data considered in this section is generated via UVA/Padova T1D [15]. This simulator is approved by the FDA and therefore, serves as robust and validated framework for generating simulated cases. The cohort of T1D cases generated can be configured with varying meal and insulin information such that each case sufficiently differs. We generate a dataset of 10 unique adult cases and each has 360 days of data for each case. There are 3 meals per day. Insulin entries vary in each day, from 1 to 5. The insulin entry can be with a meal (meal and insulin at almost the same time), or without a meal (correction bolus). Exercise is considered at certain points in the data as well, which occur at any time except for nights. The intra-subject variabilities are also introduced for the simulator. The training data accounts for 50% of the dataset, and the testing set is the rest of the data.

The clinical data was collected from T1DM subjects in a 5 month clinical trial. The CGM data was measured using Dexcom G4 Platinum CGM sensors, with measurements received every 5 minutes. The CGM sensors were inserted from the first day of the study, and calibrated according to the manufacturer instructions. Other information available in the dataset such as meal, insulin, exercises was logged by the diabetic subjects themselves. Though the data has good quality, there still exist many periods of missing data, bad points or unexpected fluctuations. The diabetic subjects behaved normally during the study as they would in their daily lives. Similar to the in silico experiment, each subject’s clinical data is halved for training and testing data.

B. Criteria for Assessment

Several criteria are used to quantitatively assess the performance of the proposed prediction algorithm. The root-mean-square error (RMSE) and mean absolute relative difference (MARD) between the predicted and reference glucose readings - for simulated or clinical CGM data - serve as the primary indicator to evaluate the general predictive performance of the algorithm.

\[
RMSE = \sqrt{\frac{1}{N} \sum_{k=1}^{N} (y(k) - \hat{y}(k|k - PH))^2},
\]

where \(\hat{y}(k|k - PH)\) denotes the prediction results provided the historical data and \(y\) denotes the reference glucose measurement, and \(N\) refers to the data size.

\[
MARD = \frac{1}{N} \sum_{k=1}^{N} \left| \frac{\hat{y}_k(k|k - PH) - y(k)}{y(k)} \right|
\]

The RMSE and MARD provide an overall indication of the predictive performance of the proposed algorithm. As mentioned earlier, the benefit of glucose prediction is in avoiding adverse glycaemic events. In this clinical context, these metrics are limited in the insight they provide. Additional metrics are needed to assess the proposed algorithm in the following perspective:

- Capability of the forecasting algorithm in differentiating between adverse glycaemic events and non-adverse glycaemic events.
- Time delay in the predicted glucose readings and the reference values to evaluate the response time provided to deal with the potential adverse glycaemic event.

The Matthews Correlation Coefficient (MCC) is used to evaluate the performance of the algorithms for detecting either adverse glycaemic event (hypoglycaemia or hyperglycaemia).

\[
MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}},
\]

where \(TP, FP, FN, TN\) stand for true positive, false positive, false negative, and true negative respectively. Here, positive indicates a hypoglycaemia (< 70 mg/dL)/hyperyglycaemia.
The LVX method are applied based on the MATLAB code involved in these algorithms are tuned carefully for optimal data generated in a way described previously. The parameters in silico horizon. Different algorithms were tested on the compared based on the accuracy over a 30-minute prediction data pre-processing. The performance of the algorithms are with that of four baseline methods: NNPG, SVR, LVX and ARX. The effective prediction horizon is defined as the prediction algorithm for a predicted value relative to its reference events far outweigh adverse events.

A singular quantitative metric is not sufficient in evaluating performance of the proposed algorithm. Consequently, the set of metrics collectively give a comprehensive description of the quality of the prediction algorithm performance.

C. In Silico Data

The performance of the proposed algorithm is contrasted with that of four baseline methods: NNPG, SVR, LVX and ARX (3rd order). The results are compared after the same data pre-processing. The performance of the algorithms are compared based on the accuracy over a 30-minute prediction horizon. Different algorithms were tested on the in silico data generated in a way described previously. The parameters involved in these algorithms are tuned carefully for optimal result. In SVR, the SVR function in Python is applied with the optimal parameters \( C = 1e2, \gamma = 0.01, cache\_size = 1000 \). The LVX method are applied based on the MATLAB code provided in \([31]\). The optimal predictor length and the number of LVs are \( J_e = 4 \) and \( N_{LV} = 4 \) respectively - this represents 20-minute historical data of glucose measurement, insulin and meal information being used for prediction. The 3rd order ARX model is optimized by MATLAB function \( arx() \) for every specific subject. The results of RMSE and MARD are summarized in the Table I and the results of forecasting of adverse glycaemic events are summarized in the Table II.

In Table I we compare the predictive error of the algorithms to measure the accuracy of the algorithms. The CRNN algorithm exhibits the best overall RMSE and MARD for the 10 simulated cases. The results in Table I are statistically significant relative to each algorithm. This observation is also evident in both the hyperglycaemia and hypoglycaemia region. In the hyperglycaemia region the CRNN shows a statistically significant improvement in glucose prediction in adverse regions (hyperglycaemia and hypoglycaemia) over other algorithms, with the exception of LVX where the improvements in RMSE (+3.13mg/dL and +0.28mg/dL) and MARD(+0.62% and +0.27%) are not statistically significant. However, Table II shows the CRNN report a statistically significant improvement in effective prediction time (+1.5min) over LVX. The CRNN model also reports relatively low standard deviations in RMSE and MARD compared to other algorithms.

\[
PH_{eff} = PH - \tau_{delay}
\]

\[
= PH - \arg \max_{\tau} (\hat{y}_k(k) - PH) * y(k) \tag{8}
\]

The effective prediction horizon definition is used to define the prediction horizon, taking into account delays due to the responsiveness of the algorithm for a predicted value relative to its reference value. Cross correlation of the predicted and actual readings is employed in performing a time delay analysis of the proposed algorithm to determine the effective prediction horizon.

\[
MCC = \frac{TP*TN - FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}
\]

**TABLE I**

| Metric       | CRNN   | NNPG   | SVR    | LVX    | ARX    |
|--------------|--------|--------|--------|--------|--------|
| RMSE (mg/dL) | 9.38 ± 0.71 | 12.91 ± 1.19\(^a\) | 12.48 ± 1.94\(^a\) | 11.32 ± 1.34\(^a\) | 13.27 ± 1.19\(^a\) |
| MARD (%)     | 5.50 ± 0.62 | 7.05 ± 0.94\(^b\) | 6.40 ± 1.36 | 6.59 ± 0.80\(^b\) | 7.46 ± 1.02\(^b\) |
| Hyperglycaemia| 14.82 ± 2.43 | 21.68 ± 2.36\(^b\) | 21.74 ± 4.41\(^b\) | 17.95 ± 4.69 | 19.91 ± 1.80\(^b\) |
| RMSE (mg/dL) | 8.44 ± 0.87 | 12.72 ± 2.65\(^b\) | 13.13 ± 8.39 | 8.72 ± 0.55 | 9.93 ± 0.74\(^b\) |
| MARD (%)     | 12.28 ± 1.48 | 20.27 ± 4.01\(^b\) | 18.81 ± 13.75 | 12.55 ± 1.70 | 14.13 ± 1.94 \(^b\) |

\(^a\) p-value ≤ 0.05 \(^b\) p-value ≤ 0.05 \(^c\) p-value ≤ 0.005

**TABLE II**

| Metric       | CRNN | NNPG | SVR    | LVX    | ARX    |
|--------------|------|------|--------|--------|--------|
| MCC          | 0.80 ± 0.10 | 0.67 ± 0.20 | 0.77 ± 0.07 | 0.77 ± 0.08 | 0.74 ± 0.10 |
| MCC          | 0.83 ± 0.05 | 0.81 ± 0.04\(^b\) | 0.82 ± 0.05 | 0.82 ± 0.05 | 0.79 ± 0.05 |
| Time (min)   | 29.0 ± 0.7 | 20.8 ± 1.8\(^b\) | 23.3 ± 1.6\(^c\) | 27.5 ± 1.3\(^c\) | 20.5 ± 1.7\(^c\) |

\(^a\) p-value ≤ 0.05 \(^b\) p-value ≤ 0.05 \(^c\) p-value ≤ 0.005

(> 180 mg/dL) event in the next 30 minutes, and true means that the classification is correct. A standard confusion matrix typically includes the Accuracy as opposed to Matthews Correlation Coefficient (MCC). This modification addresses the imbalance in classes inherent in this situation - non-adverse events far outweigh adverse events.

In Table II we compare the predictive error of the algorithms to measure the accuracy of the algorithms. The CRNN algorithm exhibits the best overall RMSE and MARD for the 10 simulated cases. The results in Table I are statistically significant relative to each algorithm. This observation is also evident in both the hyperglycaemia and hypoglycaemia region. In the hyperglycaemia region the CRNN shows a statistically significant improvement in glucose prediction in adverse regions (hyperglycaemia and hypoglycaemia) over other algorithms, with the exception of LVX where the improvements in RMSE (+3.13mg/dL and +0.28mg/dL) and MARD(+0.62% and +0.27%) are not statistically significant. However, Table II shows the CRNN report a statistically significant improvement in effective prediction time (+1.5min) over LVX. The CRNN model also reports relatively low standard deviations from which we infer a benefit in building individualized models.

An illustration of a comparison of various algorithms are shown in Fig. 5 for a virtual adult 4. As seen in the Figure 5, the CRNN exhibits the best responsiveness as the predictive curve responds rapidly towards the sharp glycaemic uptrend. The algorithm learns representations that appropriately account for both sharp slopes and gradual increments in the glycaemic curve. Consequently, at a glycaemic peak, CRNN yields a predictive curve with even higher slope to compensate
D. Clinical Data

As mentioned in the previous section, the data obtained in the clinical trial exhibits missing data, and erroneous data. This results in non-physiological discontinuities that would affect the training process. To mitigate these occurrences, the data is processed with interpolations for gaps in data, yet the interpolation points are not included in all calculations.

Table III shows the RMSE and MARD of the performance of the algorithms for the 5 cases of real data. The CRNN still maintains the best results for both the RMSE and MARD relative to the baseline methods. Relative to the performance achieved in the simulated datasets, the performance with real clinical shows worse average RMSE and MARD. This can be attributed to added external factors leading to discontinuity in data - more variability and unaccounted perturbations.

Table IV shows that the CRNN maintains the best performance in regard to detection of hyperglycaemic events. Given results in the previous section, it was observed that the RMSE and MARD of the various algorithms pointed to reduced performance between simulated cases and real cases. This observation is also made in regard to the performance in accuracy in detecting adverse glycaemic events. The reduction in performance in detecting hypoglycaemia is not significant statistically relative to the performance exhibited in other algorithms. Regarding the effective prediction horizon, CRNN maintains a better effective prediction horizon (+7.2 min) than LVX which would give a potential subject more time to avert an adverse glycaemic event. It would be more useful due to the effective prediction horizon, and CRNN has a better hypoglycaemic and hyperglycaemic accuracy in terms of RMSE. This is a significant improvement over the other baseline methods.

As seen in Figure 5, the CRNN and LVX both achieve good predictive curves compared to the ground truth measurements. Specifically, at the inflection periods during peaks and troughs, the LVX tends to have higher and lower predictions, respectively. The CRNN follows the trend at both local and global peak points closely, which increases its overall accuracy.

IV. DISCUSSION

A. Performance in Simulated and Clinical Data

In the in silico dataset, more subjects with T1DM and more complex scenarios (such as food, insulin, exercises) can be simulated. While for the clinical dataset, as real data collected from clinical trials, they are more practical and significant if people want to compare the performances of different methods. As a result, both in silico and clinical dataset are used in the results section.

In the previous section we noted a discrepancy in the performance of the proposed algorithm and baseline algorithms in simulated cases and the real patient cases. In our opinion, the drop in performance can be primarily attributed to the increased complexity of real data generated from a patient relative to the simulated data generated from a physiological model. In addition the gaps in data and method of interpolation may contribute to the further reduction in performance of the model. Relative to the baseline algorithms, the CRNN is better at capturing the features since deep learning affords a better capacity at learning optimal representations of features. This would explain the relatively lower variance.
TABLE III
RMSE AND MARD COMPARISON OF DIFFERENT PREDICTION METHODS FOR 5 REAL ADULT DIABETIC SUBJECTS

| Metric            | CRNN     | NNPG     | SVR      | LVX      | ARX      |
|-------------------|----------|----------|----------|----------|----------|
| Overall RMSE (mg/dL) | 21.13 ± 1.23 | 23.05 ± 1.24 | 27.23 ± 1.93 | 25.21 ± 0.83 | 25.36 ± 0.96 |
| RMSE (%)          | 10.08 ± 0.83 | 11.79 ± 0.86 | 12.44 ± 1.02 | 11.46 ± 1.24 | 11.75 ± 1.19 |
| Hyperglycaemia RMSE (mg/dL) | 25.28 ± 1.43 | 28.39 ± 1.32 | 35.21 ± 2.24 | 29.07 ± 0.70 | 29.62 ± 0.80 |
| MARD (%)          | 8.17 ± 0.75  | 9.35 ± 0.81  | 10.52 ± 1.05 | 8.73 ± 0.45  | 9.07 ± 0.44  |
| Hypoglycaemia RMSE (mg/dL) | 22.11 ± 4.73 | 31.10 ± 6.01 | 34.51 ± 8.56 | 24.73 ± 3.14 | 25.55 ± 2.27 |
| MARD (%)          | 27.18 ± 8.29 | 46.81 ± 10.18 | 43.29 ± 13.31 | 31.03 ± 9.48 | 29.20 ± 3.95 |

p-value ≤ 0.05 * p-value ≤ 0.01 † p-value ≤ 0.005

TABLE IV
COMPARISON OF ACCURACY MEASURES IN DIFFERENT PREDICTION METHODS FOR 5 REAL ADULT DIABETIC SUBJECTS.

| Metric            | CRNN     | NNPG     | SVR      | LVX      | ARX      |
|-------------------|----------|----------|----------|----------|----------|
| Hypoglycaemia MCC | 0.38 ± 0.10 | 0.09 ± 0.36 | 0.20 ± 0.16 | **0.39 ± 0.07** | 0.36 ± 0.07 |
| Hyperglycaemia MCC | **0.79 ± 0.04** | 0.78 ± 0.04 | 0.74 ± 0.04 | 0.76 ± 0.03 | 0.76 ± 0.03 |
| Effective Prediction Horizon Time (min) | **19.4 ± 2.0** | 13.9 ± 1.6 | 18.0 ± 1.7 | 12.4 ± 1.8 | 10.4 ± 1.6 |

p-value ≤ 0.05 * p-value ≤ 0.01 † p-value ≤ 0.005

Fig. 6. One-day period prediction results for clinical adult 17. The solid black line, dotted green line, solid magenta line, dashed blue line, dash-dotted red line indicate the simulated glucose measurements, the prediction results of the ARX method, the prediction results of the SVR method, the prediction results of the LVX algorithm, the prediction results of the CRNN method, respectively.

in metrics for the performance of the CRNN in different cases relative to baseline models. Finally, it is also noted in the hypoglycaemia event forecast where the LVX exhibits marginally better - though statistically insignificant - result over CRNN (+0.01). This could be attributed to the reduced number of hypoglycaemia cases present in clinical data to train on relative to simulated cases where CRNN exhibits best results in hypoglycaemia forecasting.

B. Results Comparisons

We achieved a mean RMSE = 9.38mg/dL in silico using the proposed method, and it is the best amongst other advanced algorithms, including SVR, LVX and 3rd order ARX. In addition, we want to compare our algorithm with other approaches in the literature. Also working on glucose dataset generated from simulators, our algorithm is also better than the results of RMSE = 18.78mg/dL [33] and RMSE = 13.65mg/dL in [34].
For several other works, it is difficult to evaluate the RMSE through direct comparison due to the unavailability of code and parameters for replicating these models. In addition, the lack of a benchmark dataset means the performances are of computed on different datasets. However, we may compare the results with widely used methods as benchmarks, such as SVR or NNPG. For instance, for PH = 30 min as shown in Table 3 [17], the algorithm is 0.1 mg/dL better than the result of SVR in terms of RMSE on the real dataset; our algorithm is 6.0 mg/dL better than the SVR in terms of RMSE on the real dataset. In [16], for PH = 30 min their RMSEs are 1.3 better than NNPG for the simulated data and 0.2 mg/dL better than NNPG for the real datasets. Our RMSEs are 3.1 mg/dL better than NNPG for the simulated data and 6.1 mg/dL better for the real datasets. As far as we are aware, the proposed algorithm achieves a performance state-of-the-art accuracy with regard to RMSE.

C. Parameters in ML and Hyperparameters in DL

For conventional machine learning tasks, feature selection is crucial to achieving a good performance. The quality of the engineered features or patterns the system learns from the data determines the accuracy of prediction or classification. Moreover, it involves tedious parameter tunings, which tends to an entirely heuristic approach. In deep learning, the features can be learnt optimally from the data automatically, and these features can be interpretable by visualization [46]. Although, several hyperparameters need to be set, including the number of layers, number neurons, activation functions, etc. A tedious grid search approach makes has to be implemented to determine optimal hyperparameters.

D. Applied to App on Mobiles

The proposed CRNN is an adaptive algorithm for different diabetic subjects. The property of adapte lies in two aspects. Firstly, it is data driven and individualised, thus different subjects have different models. Secondly, the model can be continuously evolving while people are using it. In details, the model is saved as a trained neural network. We use Keras sequential model with Tensorflow backend to train the neural network, and the result model can be easily saved as a single file. This file can be compiled as a “.tflite” or a “.pb” file to the app on mobiles, by using a Tensorflow Lite converter. The model file can be upload, upload and updated continuously at the cloud. Then the app may use it to provide the predictive glycaemic curve and show it the screen. A demonstration on the Android system is shown in Figure 7. In addition, we also found that the execution time of the model is 6ms on a Android phone (LG Nexus5 with Processor:2.26GHz quad-core, RAM:2GB) and 780ms on a laptop (MacPro with Processor: 3.1GHz Intel Core i5, RAM:8GB), though laptops usually have better processing units. The reasons might be the quantisation of weights and biases (e.g. 8 bit integer vs. 32 bit floating point) leading to simpler and faster computation at each layer.

E. Limitations

Though the CRNN has achieved very good accuracy in prediction, some challenges exist that can form the basis of further research. The underlying assumption of the deep learning approach, and indeed in the supervised learning paradigm, is that data is independent and identically distributed. Given that learning is solely based on historical data, unexpected predictions may occur given that correlations learned in the data may not imply causation. Thus hybrid approaches whereby the deep learning model is used to make an accurate prediction, and rules of meal/bolus supported by physiological model avoid apparent errors that might result. Based on the CRNN approach proposed in this paper, it is possible to develop the hybrid method, which may have the advantages of both conventional and DL algorithms.

V. Conclusion

In this paper a convolutional recurrent neural network was proposed as an effective method for blood glucose prediction. The architecture of the neural network includes a multi-layer CNN followed by a modified RNN, where the CNN can capture the features or patterns of the multi-dimensional time series data properly, and the modified RNN is capable of analyzing the previous sequential data and provide the predictive glucose level further. The method can train models for each diabetic subject using their own data, thus the model for the prediction is adaptive and specialized for each person. After obtaining the trained neural network, it can be applied locally or on portable devices, which can then be implemented widely in the daily life. It has been found that the proposed method reduces the RMSE, MARD in the in silico experiments. While in the test using the clinical data, the proposed method performs quite well in most cases comparing to other latest approaches, and shows competitive performance in forecasting hypoglycaemia and hyperglycaemia.

VI. Appendix

The authors thank EPSRC, ARISES project for support and providing the clinical data. We would also like to thank Prof.
TABLE V
COMPARISON OF DIFFERENT PREDICTION METHODS FOR THE
10-ADULT VIRTUAL DIABETIC SUBJECTS CONSIDERED IN
TERMS OF RMSE (mg/dL)

| Case | CRNN | NNPG | SVR | LVX | ARX |
|------|------|------|-----|-----|-----|
| 1    | 9.26 | 12.93 | 11.80 | 11.15 | 13.33 |
| 2    | 9.23 | 12.12 | 10.90 | 14.61 | 12.40 |
| 3    | 10.28 | 14.75 | 13.49 | 12.07 | 15.48 |
| 4    | 8.61 | 12.45 | 11.12 | 10.21 | 12.71 |
| 5    | 9.18 | 11.60 | 10.84 | 10.03 | 11.80 |
| 6    | 8.93 | 13.72 | 12.20 | 11.95 | 14.29 |
| 7    | 10.89 | 15.02 | 14.86 | 12.00 | 14.79 |
| 8    | 8.54 | 11.54 | 12.50 | 10.29 | 12.00 |
| 9    | 9.84 | 11.78 | 10.92 | 10.27 | 12.28 |
| 10   | 8.99 | 13.16 | 12.46 | 10.08 | 13.57 |
| Mean | 9.38 | 12.91 | 12.48 | 11.32 | 13.27 |
| SD   | 0.71 | 1.19 | 1.94 | 1.34 | 1.19 |

TABLE VI
COMPARISON OF DIFFERENT PREDICTION METHODS FOR THE
10-ADULT VIRTUAL DIABETIC SUBJECTS CONSIDERED IN
TERMS OF MARD (%)

| Case | CRNN | NNPG | SVR | LVX | ARX |
|------|------|------|-----|-----|-----|
| 1    | 5.42 | 6.77 | 6.84 | 6.45 | 8.45 |
| 2    | 4.87 | 6.22 | 6.06 | 7.09 | 7.30 |
| 3    | 5.41 | 7.47 | 6.52 | 6.46 | 8.92 |
| 4    | 5.70 | 7.60 | 6.43 | 6.78 | 9.69 |
| 5    | 5.62 | 6.64 | 5.90 | 6.25 | 7.95 |
| 6    | 5.43 | 7.44 | 6.76 | 8.02 | 9.18 |
| 7    | 5.02 | 6.24 | 6.23 | 6.65 | 7.60 |
| 8    | 4.89 | 6.23 | 7.44 | 6.57 | 7.51 |
| 9    | 5.50 | 6.42 | 6.50 | 6.83 | 7.73 |
| 10   | 7.18 | 9.38 | 7.98 | 10.04 | 11.56 |
| Mean | 5.50 | 7.05 | 6.40 | 6.59 | 7.46 |
| SD   | 0.62 | 0.94 | 1.36 | 0.80 | 1.02 |

TABLE VII
DIFFERENT PREDICTION METHODS COMPARISON FOR THE
CLINICAL 5 DIABETIC SUBJECTS CONSIDERED IN TERMS OF
RMSE (mg/dL)

| Case | CRNN | NNPG | SVR | LVX | ARX |
|------|------|------|-----|-----|-----|
| 1    | 20.22 | 22.13 | 27.35 | 25.71 | 25.82 |
| 2    | 19.28 | 21.27 | 23.08 | 23.77 | 23.79 |
| 3    | 21.81 | 23.98 | 27.06 | 25.80 | 26.12 |
| 4    | 21.59 | 23.15 | 27.96 | 24.77 | 24.73 |
| 5    | 22.77 | 23.72 | 25.57 | 22.99 | 25.99 |
| Mean | 21.13 | 23.05 | 27.23 | 25.21 | 25.36 |
| SD   | 1.23 | 1.24 | 1.93 | 0.83 | 0.96 |

REFERENCES

[1] S. Oviedo, J. Veh, R. Calm, and J. Armengol, “A review of personalized blood glucose prediction strategies for 16m patients,” International Journal for Numerical Methods in Biomedical Engineering, vol. 33, no. 6, p. 2833, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.2833.

[2] M. Vettoretti, A. Facchinetti, G. Sparacino, and C. Cobelli, “Type 1 diabetes patient decision simulator for in silico testing safety and effectiveness of insulin treatments,” IEEE Transactions on Biomedical Engineering, pp. 1–1, 2018.

[3] P. Posh, P. Herrero, M. Reddy, M. Xenou, N. Oliver, D. Johnston, C. Tournouz and P. Georgiou, “An advanced bolus calculator for type 1 diabetes: System architecture and usability results,” IEEE Journal of Biomedical and Health Informatics, vol. 20, no. 1, pp. 11–17, Jan 2016.

[4] A. Facchinetti, S. Favero, G. Sparacino, and C. Cobelli, “An online failure detection method of the glucose sensor-insulin pump system: Improved overnight safety of type 1 diabetic subjects,” IEEE Transactions on Biomedical Engineering, vol. 60, no. 2, pp. 406–416, Feb. 2013.

[5] S. Zavitsanou, A. Mantalaris, M. C. Georgiadis, and E. N. Pistikopoulos, “In silico closed-loop control validation studies for optimal insulin delivery in type 1 diabetes,” IEEE Transactions on Biomedical Engineering, vol. 62, no. 10, pp. 2369–2378, Oct. 2015.

[6] K. van Heusden, E. Dassau, H. C. Zisser, D. E. Seborg, and F. J. D. III, “Control-relevant models for glucose control using a priori patient characteristics,” IEEE Transactions on Biomedical Engineering, vol. 59, no. 7, pp. 1839–1849, Jul. 2012.

[7] M. M. Ahmadi and G. A. Julienn, “A wireless-implantable microsystem for continuous blood glucose monitoring,” IEEE Transactions on Biomedical Circuits and Systems, vol. 3, no. 3, pp. 169–180, Jun. 2009.

[8] A. Facchinetti, “Continuous glucose monitoring sensors: Past, present and future algorithmic challenges,” Sensors, vol. 16, no. 12, 2016.

[9] A. Revert, F. Garelli, J. Pic, H. D. Battista, P. Rossetti, J. Vehi, and J. Bondia, “Safety auxiliary feedback element for the artificial pancreas in type 1 diabetes,” IEEE Transactions on Biomedical Engineering, vol. 60, no. 8, pp. 2113–2122, Aug. 2013.

[10] Guitierrez-Usana and N. HE, “A method for evaluating data-preprocessing techniques for odour classification with an array of gas sensors,” IEEE Trans Syst Man Cybern. B Cybern., vol. 29, no. 5, pp. 626 – 632, 1999.

[11] E. I. Georga, V. C. Protopappas, D. Ardig, M. Marina, I. Zavaroni, D. Polyzos, and D. I. Fotiadis, “Multivariate prediction of subcutaneous glucose concentration in type 1 diabetes patients based on support vector regression,” IEEE Journal of Biomedical and Health Informatics, vol. 17, no. 1, pp. 71–81, Jan 2013.

[12] K. Yan and D. Zhang, “Blood glucose prediction by breath analysis system with feature selection and model fusion,” in 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Aug, 2014, pp. 6046–6049.

[13] S. Bedoui, R. Faleh, H. Samet, and A. Kachouri, “Electronic nose system and principal component analysis technique for gases identification,” in 10th International Multi-Conferences on Systems, Signals Devices (S3D’13), Mar. 2013, pp. 1–6.

[14] A. Loutfi, S. Coradeschi, G. K. Mani, P. Shankar, and J. B. B. Rayappan, “Electronic noses for food quality: A review,” Journal of Food Engineering, vol. 144, pp. 103 –111, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0260877414003276.

[15] C. Pérez-Gándia, A. Facchinetti, G. Sparacino, C. Cobelli, E. Gómez, M. Rigla, A. de Leiva, and M. Hernando, “Artificial neural network algorithm for online glucose prediction from continuous glucose monitoring,” Diabetes technology & therapeutics, vol. 12, no. 1, pp. 81–88, 2010.

[16] C. Zecchin, A. Facchinetti, G. Sparacino, G. D. Nicolao, and C. Cobelli, “Neural network incorporating meal information improves accuracy of short-time prediction of glucose concentration,” IEEE Transactions on Biomedical Engineering, vol. 59, no. 6, pp. 1550–1560, Jun. 2012.

[17] K. Plis, R. Bunescu, C. Marling, J. Shubrook, and F. Schwartz, “A machine learning approach to predicting blood glucose levels for diabetes management,” in Modern Artificial Intelligence for Health Analytics Papers from the AAAI-14.
