Learning models for glucose prediction in diabetes therapy

Mark Skelton

Submitted for the Degree of Master of Science in

Machine Learning

Department of Computer Science
Royal Holloway University of London
Egham, Surrey TW20 0EX, UK

August 31, 2020

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 7728

Student Name: Mark Skelton

Date of Submission: 31/08/2020

Signature: M. Skelton

Abstract

Type-1 diabetic patients can recieve treatment via an artificial pancreas, which measures the glucose levels of the patient, calculates the amount of insulin which needs to be administered, and then injects the appropriate amount of insulin into the patient. In order to calculate how much insulin to administer, the future glucose levels must be predicted which is the aim of this project. An attempt of using artificial neural networks is made for predicting glucose levels 30 minutes into the future, sufficient for predicting hyperglycemic and hypoglycemic events. Two models are attmpted, with one being a Long-Short Term Memory (LSTM) model, and another being a Convolutional Recurrent Neural Network (CRNN) model. Results show that both of these models are adequate for predicting glucose levels and potentially for use in an artificial pancreas. It is shown that data about insulin can be discarded, with the predictive powers of the LSTM reducing marginally. Lastly, an experiment is conducted regarding how the number of timesteps for the input data changes the root mean squared error (RMSE) of the LSTM model.

Contents

1 Introduction

The pancreas is responsible for blood glucose control by releasing insulin (when there is too much glucose) and glucagon (when there is too little glucose). More specifically, insulin is required to reduce the blood glucose level and facilitates transport of glucose from the blood into the cells and type 1 diabetic patients do not produce enough insulin in their pancreas. Hence, insulin needs to be administered externally, and this is usually done manually by the patient via an injection. We want the blood glucose concentration to be within the range 70-120 mg/dl. It is essential that the patient keeps their blood glucose levels within this range in order to avoid long-term complications [5].

Currently, most patients treat themselves with manual measurement and injection of insulin. Therefore they appear twice in the system, once as the glucose metabolic system to be controlled and again as the controller itself. This is tricky as external disturbances have to be taken into account, thus causing events where the blood glucose concentration is outside of this range. Therefore we want to avoid the patient needing to determine each insulin dose manually and limit the large variations in blood glucose concentration. This can be accomplished using an artificial pancreas.

The artificial pancreas is a device that continually monitors glucose concentrations and adjusts insulin delivery rates accordingly. More specifically, a continuous glucose monitor (CGM) transmits information about glucose concentrations to a control algorithm device (CAD), which could be a smartphone. The CAD calculates the amount of insulin to deliver using a control algorithm based on the continuous glucose measurements , which is the problem partly under consideration in this project. The calculated amount of insulin is sent to an insulin pump for administering the appropriate amount of insulin. This in turn affects the blood glucose level of the patient which affects the CGM readings. This is a closed-loop system and requires no outside input by the patient, thus increasing quality of treatment and quality of life [1], [9].

As previously mentioned, this project will focus on glucose (or CGM value) prediction which is required for calculating the amount of insulin that needs to be administered. Data consists of the three variables: CGM values, insulin values, and ingested carbohydrates. For each record, there is a one day trajectory with a one minute resolution. Larger than one minute resolutions may be obtained by missing out the data points at the corresppong times. The training data consists of 756 records and the test data consists of 324 records. It was obtained from the paper [4]. So the overall objective is to predict future CGM values using past CGM, insulin, and carbohydrate values.

In section two we go over the literature, in section three we look at the model designs considrered, in section four we look at the reults of each model, and in section five we conclude our findings.

1.1 How will this project improve my future career?

This project will demonstrate and reinforce the knowledge that I have obtained from my machine learning degree, epsecially in terms of preprocessing data and in evaluating and improving the performance of a machine learning model. I will improve my ability to train neural network models, and machine learning models in general, which is something I need to learn since I aspire to become a machine learning engineer. It should give me good experience with real world data, which is something that I could demonstrate to future employers.

I can learn to show the results of the research effectively, and to point out the intersing parts of the results and explain what they mean. It should show that I can take leadership over the project and decide the interesting areas for research as well.

Lastly, it should also show that I can handle large projects, and that I can successfully plan and execute the steps required for completion. It should show that I have good time management and the discipline to work towards the deadlines that I have set for myself.

2 Background

2.1 The Artificial Pancreas

The articles by Boughton et al. [1] and Lunze et al. [9] both talk about the artificial pancreas system as a whole, summarising that diabetic patients have a pancreas that does not produce enough insulin and hence an artificial pancreas would be useful. There is the single-hormone system which focuses on insulin injection only, and the dual-hormone system which delivers both insulin and usually glucagon. Adding glucagon could potentially reduce the chance of hypoglycaemia (low glucose concentrations), but the system and device becomes more complicated.

We want minimal device burden, user interaction, and inconvenience while still maintaining optimal glucose control. This can be difficult; for example meal announcements can be tricky to implement fully into a closed loop system. Currently a hybrid approach of requiring insulin boluses for meals is used. There is a trade-off between the ability to control the blood glucose concentration and the reduction of patient input (and device intrusiveness), including the request of external patient data such as weight, actual glucose ingestion (meals), and muscular activity.

Closed-loop systems have been shown to reduce the amount of time in hypoglycaemia and increased time in the target glucose range compared with the control group (who used sensor-augmented pump therapy). This is true for both single and dual-hormone systems, with the dual-hormone system doing an even better job at this. However, the lack of commercially available room-temperature-stable glucagon, device complexity, and the short duration and small size of clinical studies all limit the progress of dual-hormone systems. A hybrid single-hormone closed-loop system is approved for use in people aged 7 years or older and has at least 100,000 users. There is no commercially available dual-hormone system yet.

Some ways to improve closed-loop systems include using CGM devices with increased accuracy and longer wear time. Also, algorithms implemented directly into the insulin pump and thus reducing device burden would be a good target. More specifically to this project, improved algorithms that integrate multiple signals (such as measures of activity) that can more accurately measure insulin requirements than using only sensor glucose input would be beneficial. This project does just this, with information going beyond just glucose concentration measurements by also using information such as meals.

To improve artificial pancreas systems, have control methods which automatically respond to disturbances without the need for external patient information. This is not addressed in my project, as these disturbances form a part of the data. Extending the model to include glucagon as a actuating variable is not likely to be a part of my project either (maybe in an extension). Obtaining optimal performance considering time delays is the main aim of my project, since it is all about predicting glucose levels.

Also, improving performance evaluation by doing animal trials with a similar metabolic behaviour and extending patient models to include the effect of physical activity would be instructive. Improving therapy devices, such as better sensors to reduce the time delay and without needing calibration, would be beneficial. Then, the entire device could also be completely implantable (right now it is subcutaneous) which reduces device burden. If this were possible then models that can anticipate disturbances would not be needed.

2.2 Standard Models

The article by Lunze et al. [9] goes further and talks about the different models used for prediction, as well as for evaluation and testing purposes. Indeed, these models can be used as artificial patients, which are useful for testing artificial pancreas systems due to the avoidance of any dangers in real human trials. The variables involved include glucose concentration as the controlled variable and the insulin infusion rate as the actuating variable.

A serious problem for control design is the time delay between glucose sensing and insulin injection. Therefore, advanced control algorithms based on patient models are needed which themselves require patient data in order to adapt control performance to system changes. In fact, in diabetic patients there is a time delay of up to 40 minutes due to delays in the effects of the blood glucose concentration after injection of insulin and the measurement delay between blood glucose and interstitial glucose concentration.

The models are described by a system of differential equations, which are white box models as they have interpretable model components. An example is the minimal model which is used to simulate the response of blood glucose concentration to an injection of glucose (an intravenous glucose tolerance test). This is a small nonlinear third order model. Another model by Sorensen is a patient model consisting of 19 differential equations including the insulin and glucagon systems. It is often used as an artificial patient for testing. Others have produced models for different uses. All models are represented by a set of differential equations involving variables such as (for the minimal model) glucose concentration (in plasma), insulin concentration (in plasma), insulin infusion rate, and intravenous glucose injection (a disturbance).

So patient models are essential for closed-loop computational trials, and for use in white-box models. What about black box? In black-box models, we control blood glucose concentration without detailed knowledge of the patient's internal metabolic behaviour. Methods include curve fitting, and using a lookup table to determine the necessary system input for the desired system output. Also used is rule-based control where, depending on the past and current system outputs, the system input is determined. A common method is the PID method, where the control strategy is dependent on output error and its proportional-integral-derivative (PID) behaviour. Another is the repetitive control strategy. These common methods are preferred as they are still based on mathematical calculations, even though no detailed knowledge of the patient's behaviour is required.

Grey-box models can be used for both prediction and evaluation purposes. Here, a partial theoretical structure is combined with data to produce the model [12].

There are a couple of tables in the paper by Lunze et al. [9] which compare different control algorithms.

In conclusion, black-box models have a simple structure, do not require detailed information of patient behaviour, and are easily designed. However, the system is not optimal due to the time delay. Model-based control algorithms may prevent critical events from occurring since they predict the plant model's behaviour. Their response depends on the accuracy of the internal model, meaning that control performance depends on the complexity of the model and adaptation to the patient's specific glucose metabolism.

2.3 Machine Learning

Now onto some machine learning methods. An article by Sparacino et al. [13] describes some of the earliest machine learning models applied to this problem. Two models are considered in this paper; one using a first-order polynomial model to describe past glucose data, and the other using a first-order autoregressive (AR) model to describe past glucose data. So for each model, the parameters are identified using weighted least squares techniques. Once the model is defined, it is used to forecast glucose levels for a given prediction horizon (PH).

The data involved is simply a time-series of glucose concentration, monitored every three minutes (from 28 diabetic patients) for nearly 48 hours using a CGM device.

Results show that glucose levels can be predicted in time. With a prediction horizon of 30 minutes the crossing of the hypoglycaemic threshold can be predicted 20-25 minutes in advance which is sufficient time to mitigate the event with sugar ingestion.

For both polynomial and AR models, the algorithm is to initialise at time step $n = 1$ and repeating:

1. Collect n^{th} sample u_k

2. Fit the model to $u_n, u_{n-1}, \ldots$; retrieving model parameters θ. Use weighted linear least squares

3. The model is then used to predict $\hat{u}_{n+T}$

4. $n = n + 1$

Both methods are based on time-varying models, or time-varying θ. All the past data participates with different relative weights to determine θ. μ is the forgetting factor, between 0 and 1, to determine how much weighting to give to past data. For example: 1, 0.2, 0.04, 0.008, ... for $\mu = 0.2$.

The results of both models are shown to be acceptable, with no one model being clearly better than the other in terms of mean square prediction error. Overall AR is preferable due to its higher accuracy even when large values of μ are considered.

In this project we will aim to build a neural network to model glucose behaviour. It has been attempted before, with Li et al. [8] using a convolutional recurrent neural network (CRNN) as shown in figure 1. Machine learning allows systems to build models by finding patterns in the data and extracting them for prediction. Deep learning generally performs better than traditional machine learning techniques, due to their ability to automatically learn features with higher complexity and representations. Hence it is useful for the problem of glucose prediction, and so a CRNN architecture is proposed. The CRNN architecture involves layers for Convolutional Neural Networks (CNN), and layers for Recurrent Neural Networks (RNN), which contain Long-Short Term Memory (LSTM) cells. These concepts are explained in the following subsection.

The model is trained on data comprising of CGM, carbohydrate, and insulin data. The data originates from a mixture of simulated cases and clinical cases. After pre-processing, this data is fed to a CRNN for training. The method trains models for each diabetic patient, using their own data. The architecture is split into three parts, and is trained end-to-end in Tensorflow:

1. A multi-layer CNN to extract the data features (convolutional layers using a 1D Gaussian kernel filter to perform the temporal convolution and pooling layers)
2. The output of the CNN is input for the RNN layer, which uses LSTM cells
3. Fully-connected layers, giving a regression output

In the paper, this model is compared with support vector regression (SVR), the latent variable model (LVX), the autoregressive model (ARX), and a neural network for predicting glucose algorithm (NNPG) which is an existing neural network design. It is shown to be competitive in terms of forecasting accuracy and has superior performance in terms of RMSE (Root Mean Squared Error).

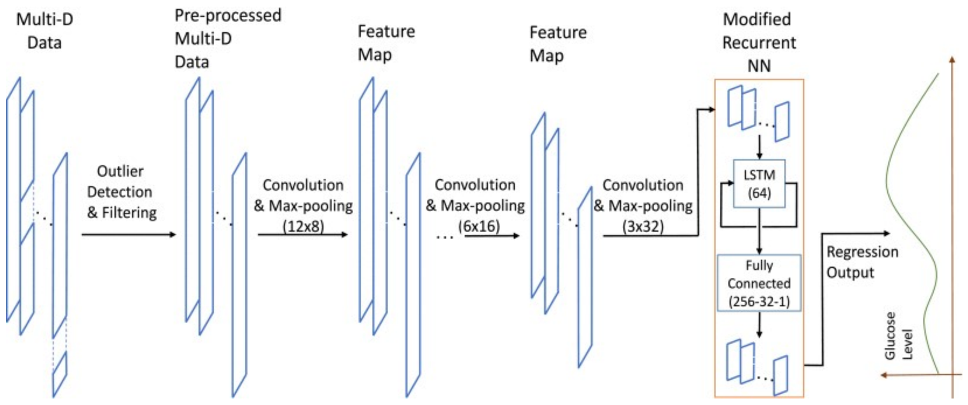

Multi-D Data
Outlier Detection & Filtering
Pre-processed Multi-D Data
Convolution & Max-pooling (12x8)
Feature Map
Convolution & Max-pooling (6x16)
Feature Map
Convolution & Max-pooling (3x32)
Modified Recurrent NN
LSTM (64)
Fully Connected (256-32-1)
Regression Output
Glucose Level

Figure 1: The CRNN architecture used by Li et al. for predicting glucose levels [8]

2.4 Artificial Neural Networks

Artificial neural networks (NN) are a series of layers of neurons that send signals from one layer to the next to output a prediction for a given problem, and it is trained as follows. A forward pass starts with input signals going from the input layer of neurons. These signals are sent to the neurons of the next layer, after being multiplied by a weight. The combination of signals add up, and then an activation function is applied to give the output signal. This is demonstrated in figure 2. This is repeated until the end of the NN is reached, giving the final output for the prediction. This prediction is compared with the true label to give the loss. The backward pass (back propagation) is then used to find the gradients of the loss with respect to the weights, which is then used to update the weights such that the loss is slightly reduced via gradient descent. This is shown in figure 3. This process is repeated until convergence, producing a fully trained NN [3].

2.4.1 CNN

CNN architectures involve two types of layers, which are convolutional layers and pooling layers (usually, max pooling). CNN architectures are usually applied to problems involving images, so it is described in relation to images as inputs. Image inputs are usually 2-dimensional pictures split into 3 different colour channels (red, green, blue).

Convolutional layers involve filters that map the input pixels of a certain region of an image (covering all channels) to a single number as output. This

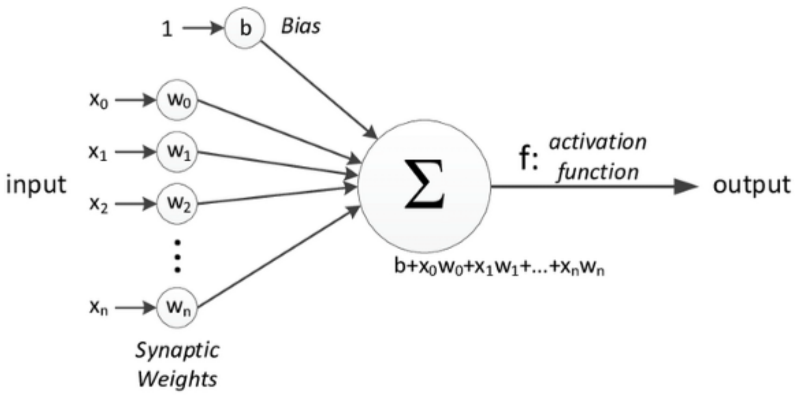


Figure 2: A perceptron. The input signals are multiplied by some weights, summed and then the avctivation function is applied [3]

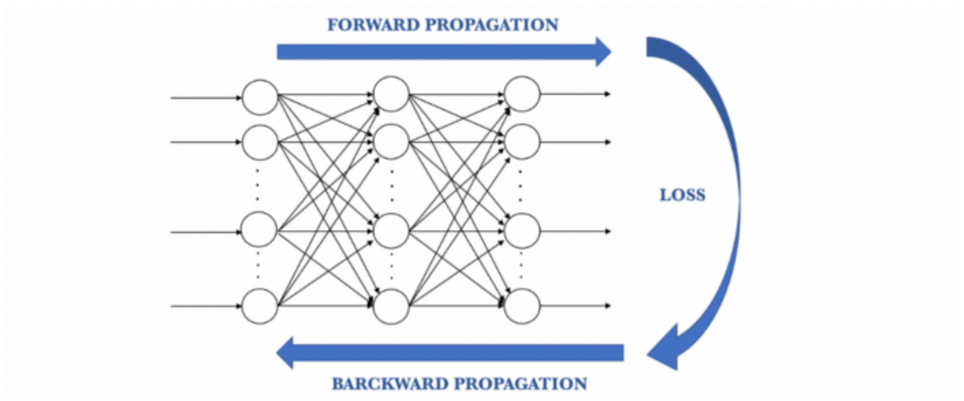


Figure 3: A typical NN where a forward pass is applied, the loss is found, and then back propagation is performed [3]

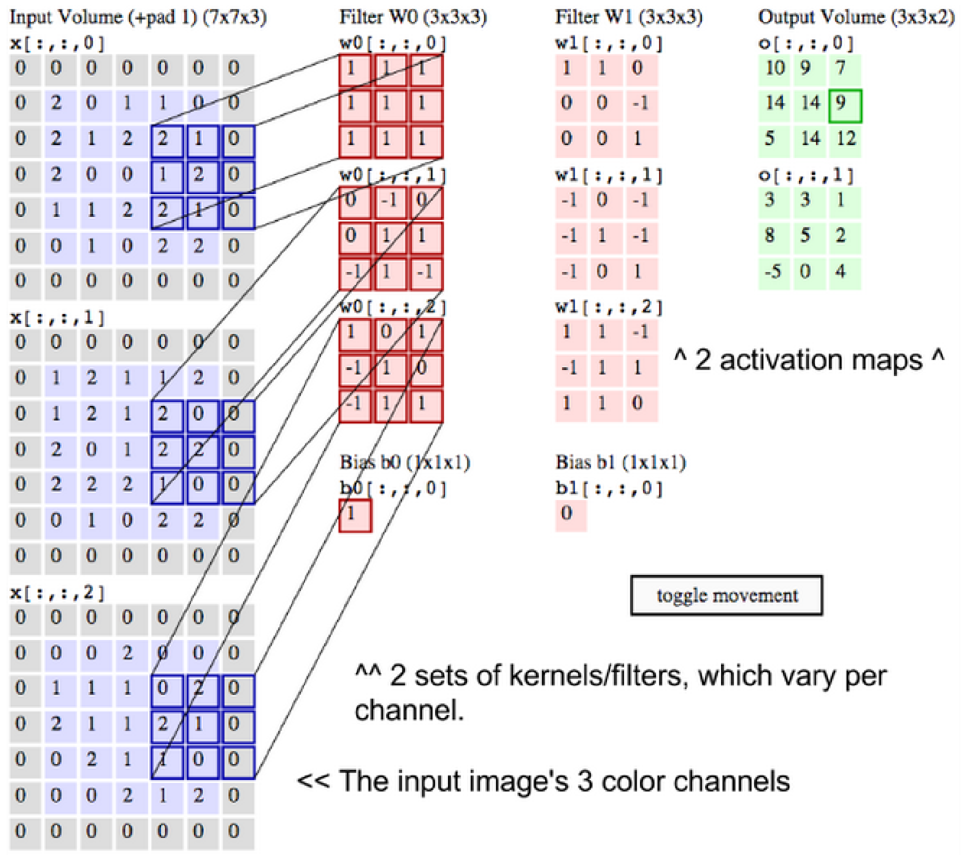


Figure 4: An example of the application of one filter to one region; stride length of 2 [10]

filter is repeated for each region of the image to produce a channel of the output image. Multiple filters can be applied to give multiple channels, with one filter producing one output channel. An example is shown in figure 4.

Max pooling layers take a specified window of the input image and simply outputs the maximum, as shown in figure 5. The window slides across the image with a sepcified stride length and outputs the maximum of each region to create the output image. It is possible to apply some other function to each region instead of just taking the maximum, such as taking the average (average pooling).

Usually a convolutional layer is applied followed by a max pooling layer. A sequence of these can be applied in a CNN architecture, which at some

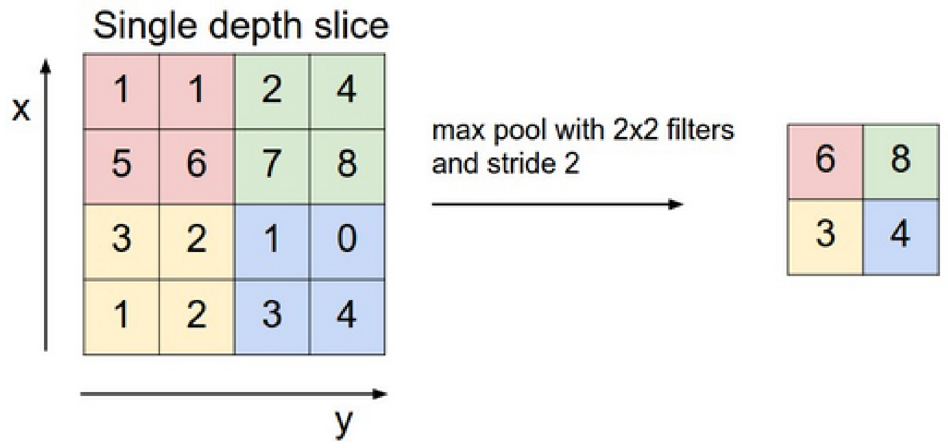


Figure 5: An example of applying max pooling to an image [10]

point usually flattens the 2-dimensional image into a vector and finshes with some dense layers. CNN architectures are useful for image problems due to their ability to find local patterns in the image (maybe like the ears of a cat). [10]

2.4.2 LSTM

The LSTM model was first described in the original paper by Hochreiter et al. [7], which showed that LSTM models learn a lot faster than RNN models, and more successfully, while also solving tasks involving a long time lag. RNN models are used for learning dependencies of present states based on past input information. This is done by keeping information from the previous input, by taking the output from the previous input and feeding it back into the NN alongside the next input. See figure 6 for an unrolled RNN.

Theoretically, this should keep all the information from previous input, but in prcatice only the most recent information is kept. This is where the LSTM model comes in. The architecture is shown in figure 7.

The LSTM architecture is the same as that of the RNN, except with a more external memory and with gates controlling the memory. Each module, which is where input at each time step is processed, now also contains functions that are described as the forget gate, input gate, and output gate, which controls how much of the input memory is forgotten, how much of the input information is added to the memory, and how much of the memory to

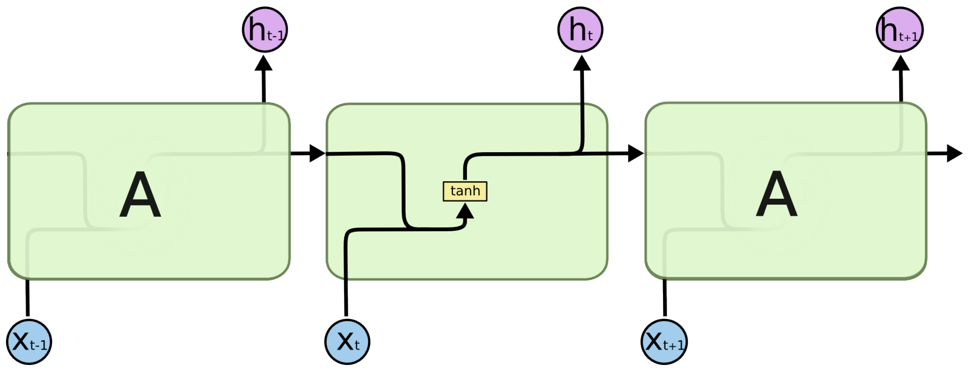


Figure 6: A simple RNN [11]

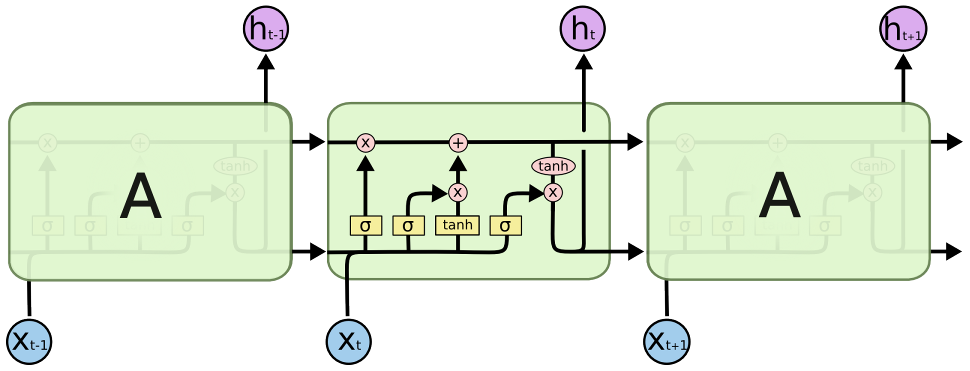


Figure 7: An LSTM [11]

output into the output state of the module respectively.

The LSTM model can thus learn the weights involved with each gate and therefore it can learn which information is imprtant to remember and when to use it. Useful information can be stored in memory for long periods in this architecture, meaning that long-range time dependencies can be learnt. Overall, LSTM models perform better than RNN models because of this [11].

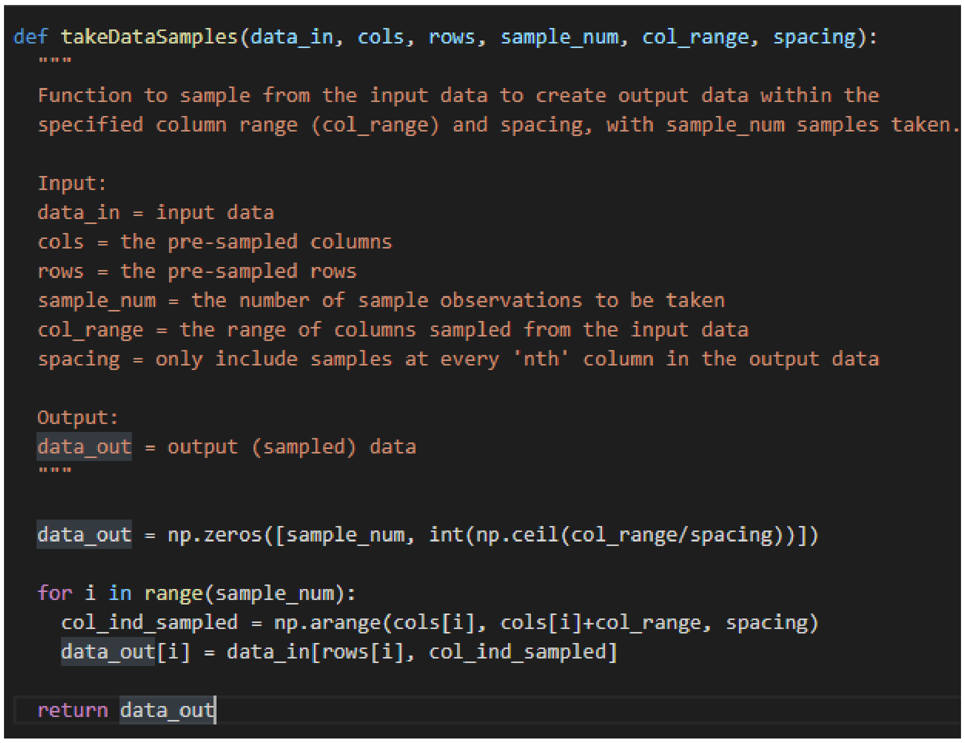
3 Methods

In this project I used data from [4], which is time series data for glucose, insulin and carbohydrate measurements. Each sample corresponds to one 24 hour period, with a resolution of one minute, for an artificial patient. The training data contains 756 samples and the test data contains 324 samples. This data is preprocessed in order to work well for an artificial neural network (NN) and to increase the sample size, as explained below. Afterwards, the data is fed into a basic NN, which is a single layer RNN which outputs the prediction for the next 30 minutes.

3.1 Data Preprocessing

So we want to be able to predict glucose levels with a prediction horizon of 30 minutes. However, if I use the data as it is and always take the final 30 minutes then the prediction horizon will always be sat the end of the day which is not the intention. We want to predict 30 minutes ahead at any given time of the day. Additionally not all of the original data will be needed because it is unlikely that glucose levels many hours prior will be a good indicator of glucose levels in only 30 minutes time. On top of this, the resolution of one minute may be more detail than is required for our purposes. A resolution of 5 or 10 minutes will be sufficient, since it is the typical update period of a CGM, and it gives enough detail to give warning for hypoglycemic and hyperglycemic events in 20-30 minutes. It would also be instructive to investigate the length of time needed to give accurate predictions for glucose levels, which is explored in the results section.

Therefore a function to take random time series samples is required, as shown in figure 8. In it, a random row (original sample) and a random column (time step) are chosen as the starting point for the new sample. This ensures that a variety of 30 minute prediction horizons are chosen, in terms of the time of day. The number of samples can also be specified,



```python
def takeDataSamples(data_in, cols, rows, sample_num, col_range, spacing):
  """
  Function to sample from the input data to create output data within the
  specified column range (col_range) and spacing, with sample_num samples taken.

  Input:
  data_in = input data
  cols = the pre-sampled columns
  rows = the pre-sampled rows
  sample_num = the number of sample observations to be taken
  col_range = the range of columns sampled from the input data
  spacing = only include samples at every 'nth' column in the output data

  Output:
  data_out = output (sampled) data
  """

  data_out = np.zeros([sample_num, int(np.ceil(col_range/spacing))])

  for i in range(sample_num):
    col_ind_sampled = np.arange(cols[i], cols[i]+col_range, spacing)
    data_out[i] = data_in[rows[i], col_ind_sampled]

  return data_out
```

Figure 8: The function used to take data samples

which means that many more data samples than in the original data can be obtained. In this project 10,000 data samples were chosen, which is two orders of magnitude larger than in the original data set and therefore the accurracy of the model should benefit. The column range and resolution also need to be specified; in this case the column range is a time period consisting of 190 minutes and a resolution of 10 minutes.

The validation set is created by splitting the training data. A ratio of 70% for the training set and 30% for the validation set was used. Both the training set and validation set were obtained from the original training data file, with the test data coming from the original test data file. The prediciton horizon, in other words the label, is split from the data at this point as well. The label consists of the array of glucose readings with enough time steps (depending on the resolution chosen) to cover the prediction horizon of 30 minutes.

The data is normalized by shifting the data to have a mean of 0 and a standard deviation of 1, for both the glucose readings and insulin values. The carbohydrate values are both sparse (most entries have a value of 0) and very discontinuous (values change suddenly), therfore it was decided that simply shifting the carbohydrate data to be between 0 and 1 was appropriate to avoid problems with division by 0 (to get a standard deviation of 1).

Finally the data for glucose, insulin and carbohydrates is concatenated for each time step, for each data sample. The data has the shape (batch, time step, feature) which is appropriate input for RNN.

3.2 LSTM Model

The model is a single LSTM layer, a type of RNN layer, which outputs a vector of length 10. Also the tanh activation function is applied to the output. This is fed into a dense layer, with no activation function applied to the output, which outputs glucose predictions for the number of time steps needed to cover the prediction horizon at the given resolution. The reason for attempting an LSTM model is that LSTM models, or RNN models in general, take sequential data as input. This is the form of the data in this problem, and while using the usual dense layers could work they would not be as good at picking up dependencies of present data on past data.

The model is compiled using the mean squared error as the loss function, and using the RMSprop optimizer, both required for gradient descent and for updating the model. The model is evaluated using the root mean squared error (RMSE).

The model is trained on 7000 samples out of the 10000 samples in the training set proper, with 3000 samples used as the validation set, for evaluating the model as it trains. The validation set is used instead of the test set so that the test set is never used in any way to improve the model and thus avoiding data leakage. The test set, which has 5000 samples, is used for evaluating the model with the results given in the results section. A batch size of 64 (which is the number of samples used before an update via gradient descent) is used, and 200 epochs (the number of runs through the training set) is used.

3.3 Ignoring Insulin Data

How significant is insulin data for predicting glucose levels? This is a valid question to ask, since glucose data is also used as input data and insulin levels are adjusted by the pancreas according to glucose levels. Hence, it is reasonable to assume that some sort of correlation exists between insulin and glucose values, meaning that insulin does not contribute much additional information in regards to predicting future gluose levels.

So we tried the same model as above, but without insulin data as input.

3.4 Varying input time series lengths

In the previous sections a fixed input time series length of 120 minutes was used to train the model, but how much data is needed to make accurate predictions for a prediction horizon of 30 minutes? This is a useful question to ask, as the less data is need to for prediction then the more efficient the algorithm can be in data usage.

This was explored by iterating through different input time series lengths, preprocessing the data, creating the model, training the model, and then evaluating the performance of the model by calculating the RMSE which is stored in a list. The list of RMSE for each number of input timesteps can then be plotted to show what the the minimum number of timesteps that is approximately required for close to optimal performance.

Note that the data must be preprocessed again only because the sampling of the data requires knowledge of the number of timesteps, since difference between the starting index of the sample and the actual endpoint of the original sample cannot be smaller than the number of timesteps to be sampled. Also, every other step for data preprocessing occurs after this sampling step meaning that the entire process must be repeated for every time series length trialled.

Here, the full training set of size 10000 is used per model, since the output is silenced as the model trains and so the validation set is not needed. The test set still has 5000 samples for the evaluation of every model.

3.5 CRNN

A more complicated model involving convolutional and maxpooling layers before the usual LSTM and dense layers was attempted, similar to the model by Li et al. [8]. The reason for adding convolutional layers is that they are good at finding and extracting important local features from within the data. They are usually used for image recognition techniques, where they do just this.

More specifically, the layers were structured as follows:

1. A 1-dimensional convolutional layer with 8 filters, of size 3, and padding such that the input number of timesteps equalled the output number of timesteps (i.e. padding = 'same'). A ReLU activation function was applied to the output.

2. A 1-dimensional max pooling layer with a pool size of 2, a stride length of 2, and padding = 'same'.
3. A 1-dimensional convolutional layer with 16 filters, of size 3, and padding = 'same'. A ReLU activation function was applied to the output.
4. A 1-dimensional max pooling layer with a pool size of 2, a stride length of 2, and padding = 'same'.
5. An LSTM layer, with an output size of 10, and also applying a tanh activation function to the output.
6. A dense layer with an output size of 3, and linear output (in other words, no activation function is applied).

Again, the CRNN model is compiled using the mean squared error as the loss function, and using the RMSprop optimizer. The model is trained on 7000 samples out of the 10000 samples in the training set proper, with 3000 samples used as the validation set. There are 5000 samples in the test set, as before. The batch size is set to 64, and 200 epochs are used.

4 Results

4.1 LSTM model compared with other models

The LSTM model had an RMSE of 0.3452 mmol/L for a prediction horizon of 30 minutes. Remember that an input of 120 minutes was used, from 756 records, and input data included glucose, insulin and carbohydrate readings. Figure 9 shows a comparison of predicted glucose level and the true glucose level as a function of time for a few samples. The preceding timesteps are shown as well as the timesteps to predict for the true label, whereas only the timesteps to predict are shown for the glucose level predictions obviously.

The CRNN model developed by Li et al. [8] also used an input time of 2 hours exactly, and also used glucose, insulin and meals as input data. The data used for this paper was generated by 10 adult artificial patients. While the data used for both models were from artificial patients, the models and therefore the data itself may vary. This CRNN model was found to have an RMSE of 9.38 mg/dL for a prediction horizon of 30 minutes when testing on 10 virtual adult patients, which equates to 0.5211 mmol/L.

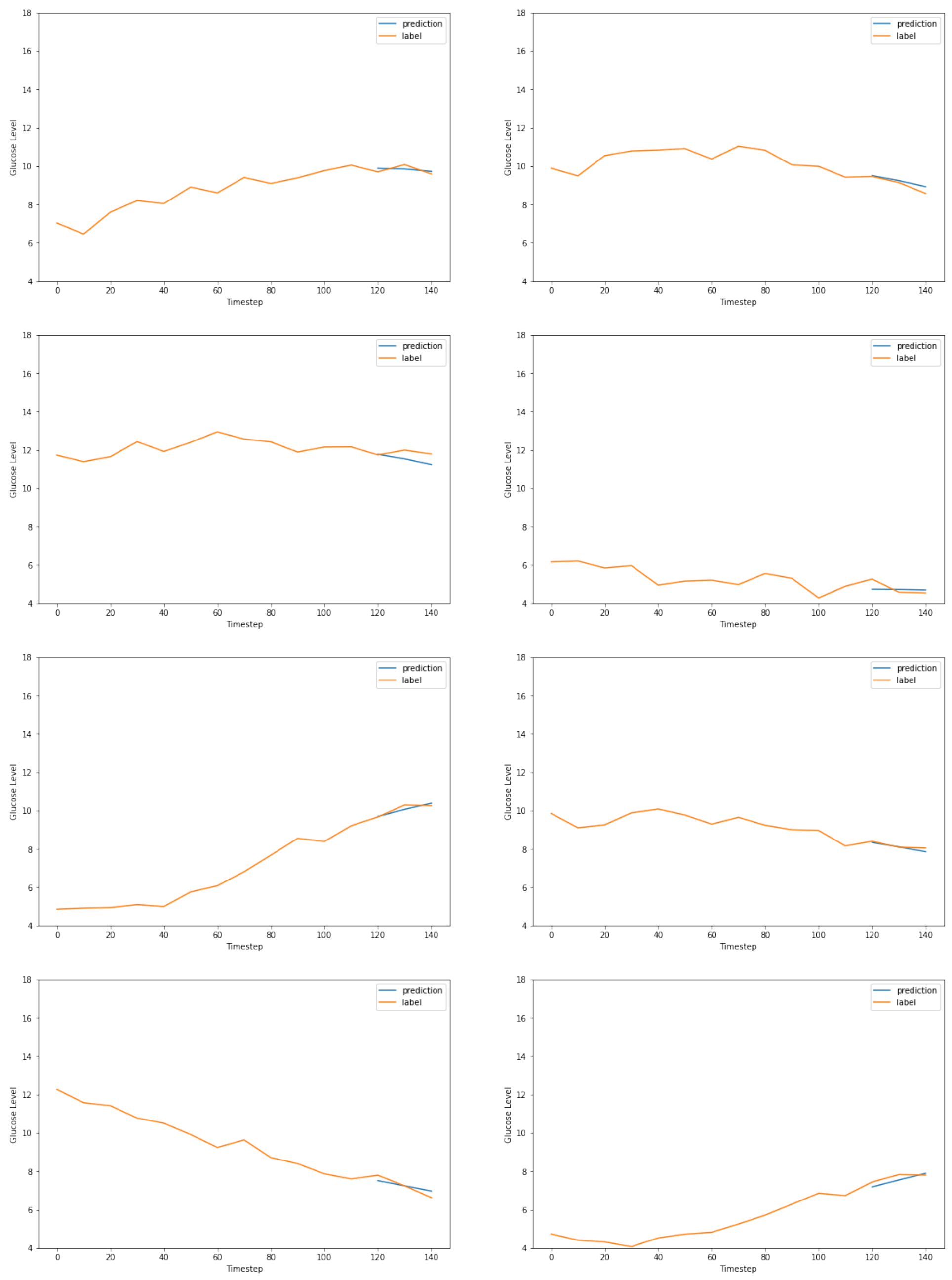


Figure 9: Glucose Level vs Timestep, for the LSTM Model with Insulin in the Input

4.2 LSTM with and without insulin

The RMSE for the input data which includes insulin was found to be 0.3452 mmol/L, and the RMSE for the input data not including insulin was found to be 0.3504 mmol/L. The prediction horizon for both cases is 30 minutes. The plots comparing predicted glucose levels and true glucose levels are also shown for the LSTM model without insulin in figure 10.

4.3 Varying input time series lengths

The results of trialling different input time series lengths are sumarised in figure 11, which shows a step decline in RMSE as the number of timesteps increases before levelling out at around 100 minutes.

4.4 CRNN

After evaluating the CRNN model on the test set, an RMSE of 0.3533 mmol/L was output. Figure 12 shows the plots comparing predicted and true glucose levels for a few samples.

4.5 Compare all models in one plot

The predictions from the LSTM model, both with and without insulin, and the CRNN model is shown alongside the true glucose labels in figures 13 and 14.

5 Conclusions

5.1 LSTM model compared with other models

Considering the simplicity of the LSTM model compared to the CRNN model by Li et al. [8], it is surprsing to see that the RMSE of the LSTM is so much lower than the RMSE of the CRNN, with the difference between the two models being approximately 0.18 mmol/L. A direct comparison is difficult since the datasets that produced the two models are different, and therefore the differnce in quality between the two models could be due to the quality of the data. However, the results are very promising and shows that the LSTM model is a capable model for predicting glucose values for type-1 diabetic patients.

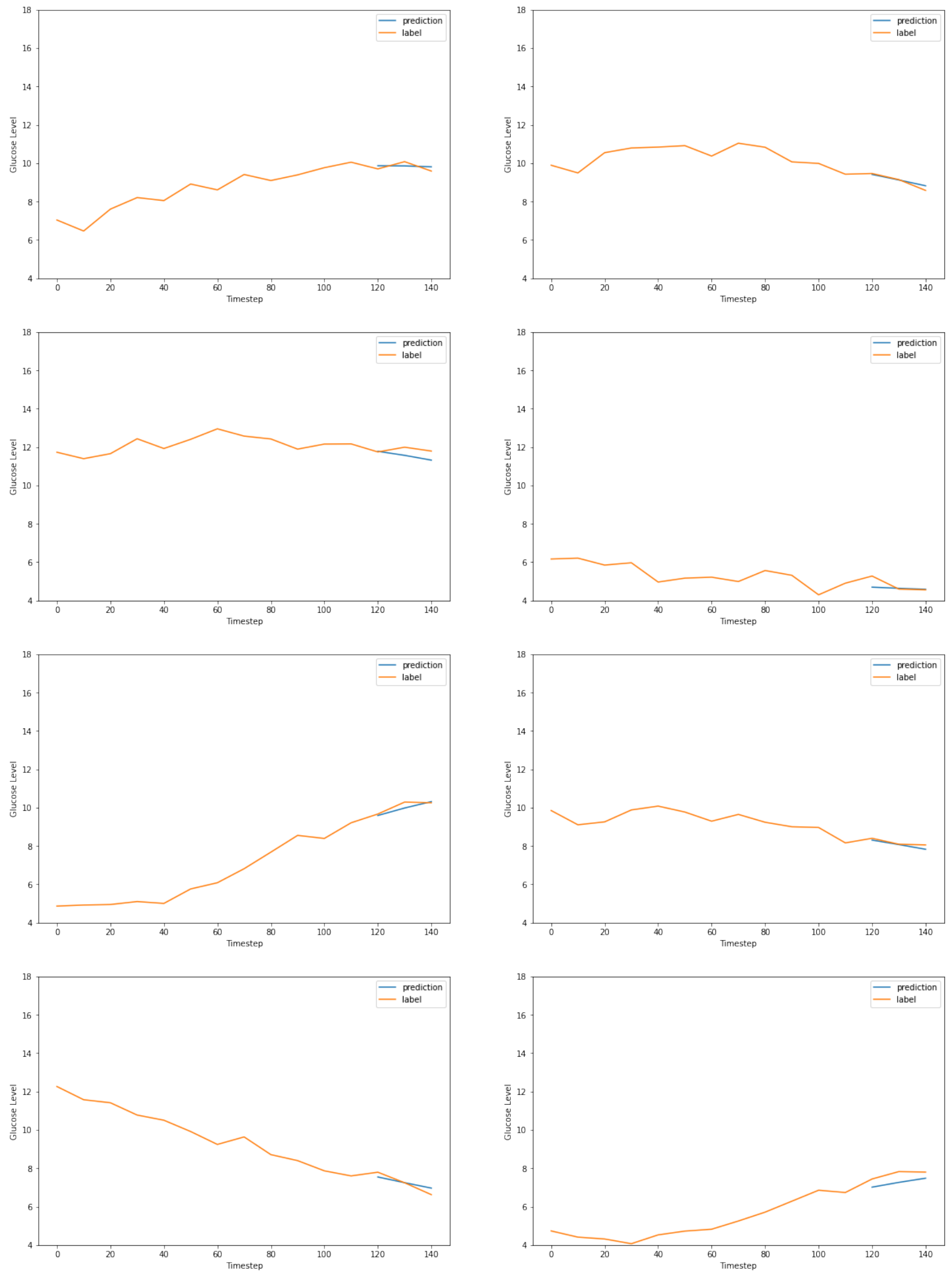


Figure 10: Glucose Level vs Timestep, for the LSTM Model without Insulin in the Input

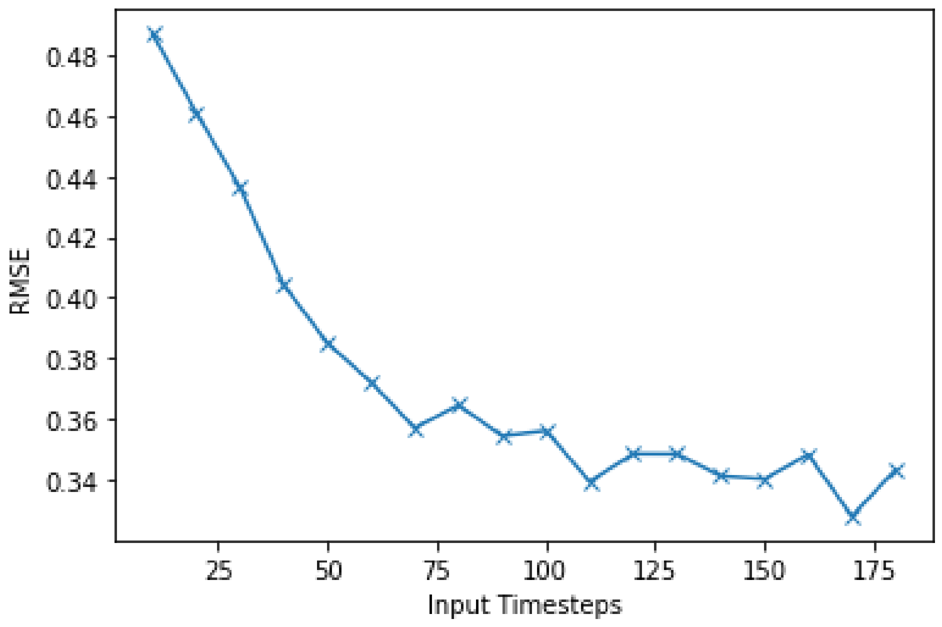


Figure 11: RMSE vs Input Timesteps

Perhaps the difference in RMSE could be attributed to the resolution of the data being different for this study compared to the study conducted by Li et al.

The LSTM model appears to be good enough to predict hypoglycemic and hyperglycemic events and to be used as a model for the artificial pancreas.

5.2 LSTM with and without insulin

The RMSE for the input containing insulin values is better than when insulin is not included in the input. However, the difference is only about 0.01 mmol/L which is only a relatively small difference in RMSE. This result shows that insulin values in the input data is not essential and can be removed to obtain similar values of accuracy, thus saving on costs for obtaining such data to be spent on gaining data on more essential features.

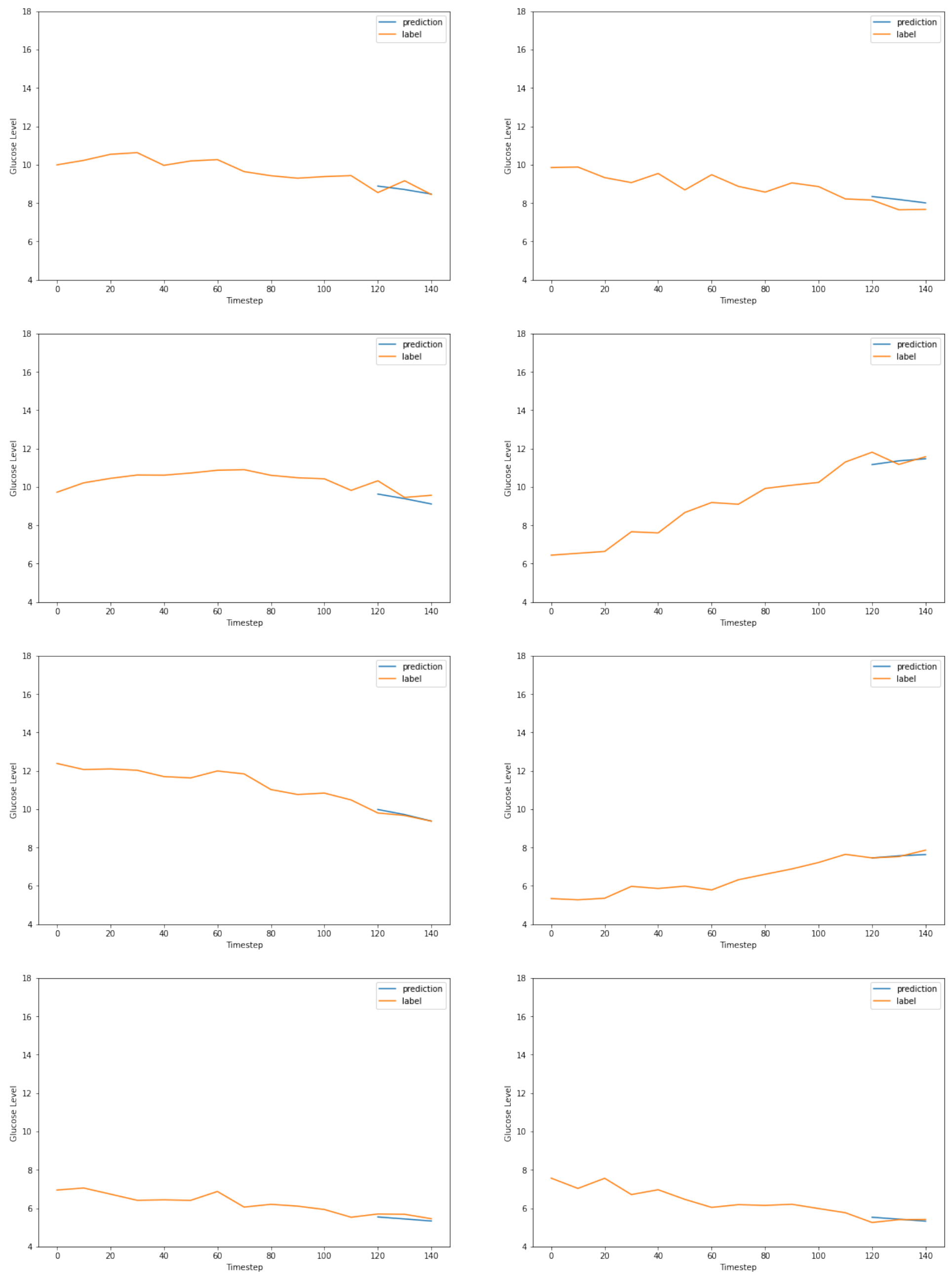


Figure 12: Glucose Level vs Timestep, for the CRNN Model

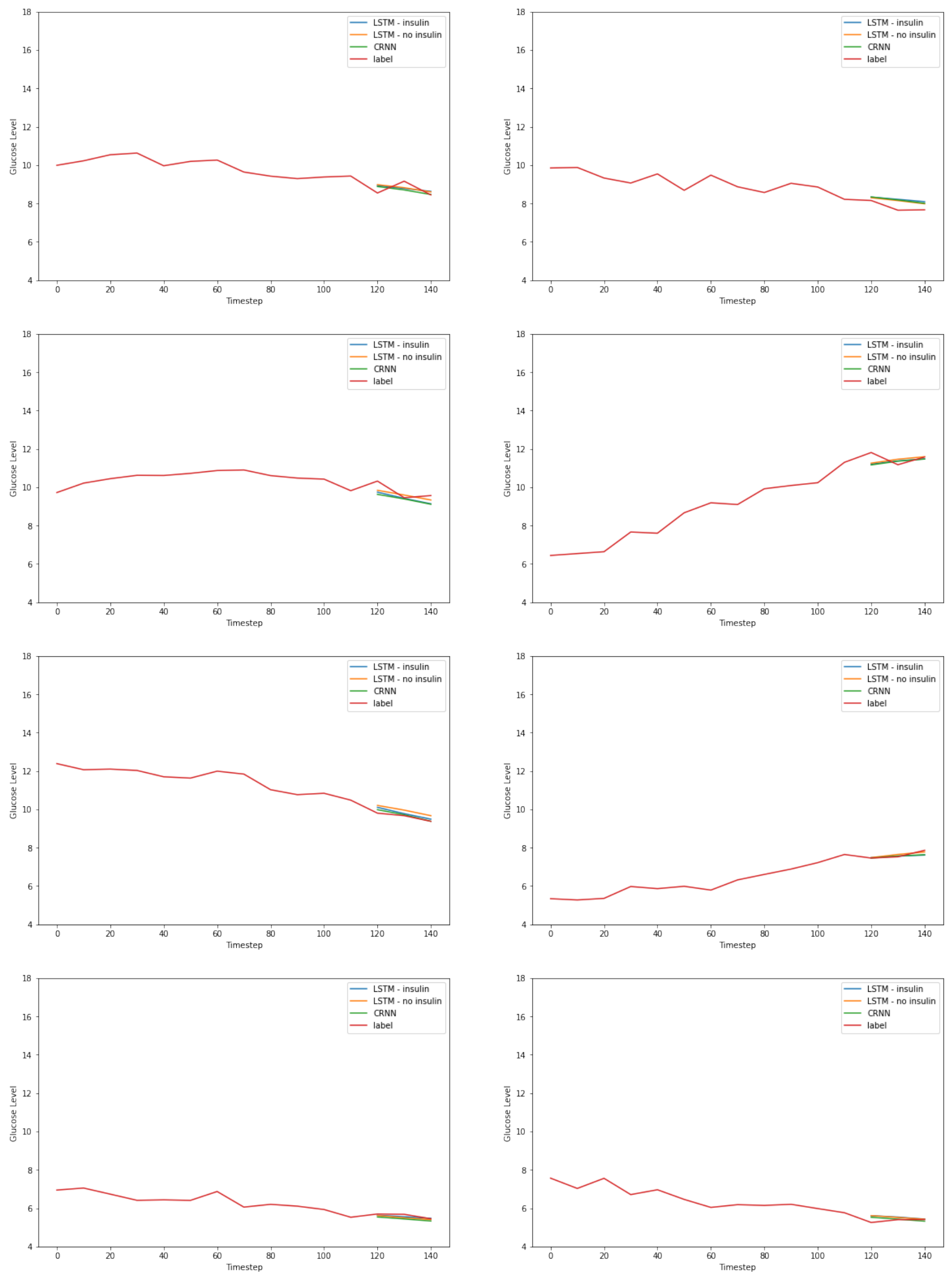


Figure 13: Glucose Level vs Timestep, for the LSTM (with and without insulin) and CRNN Models

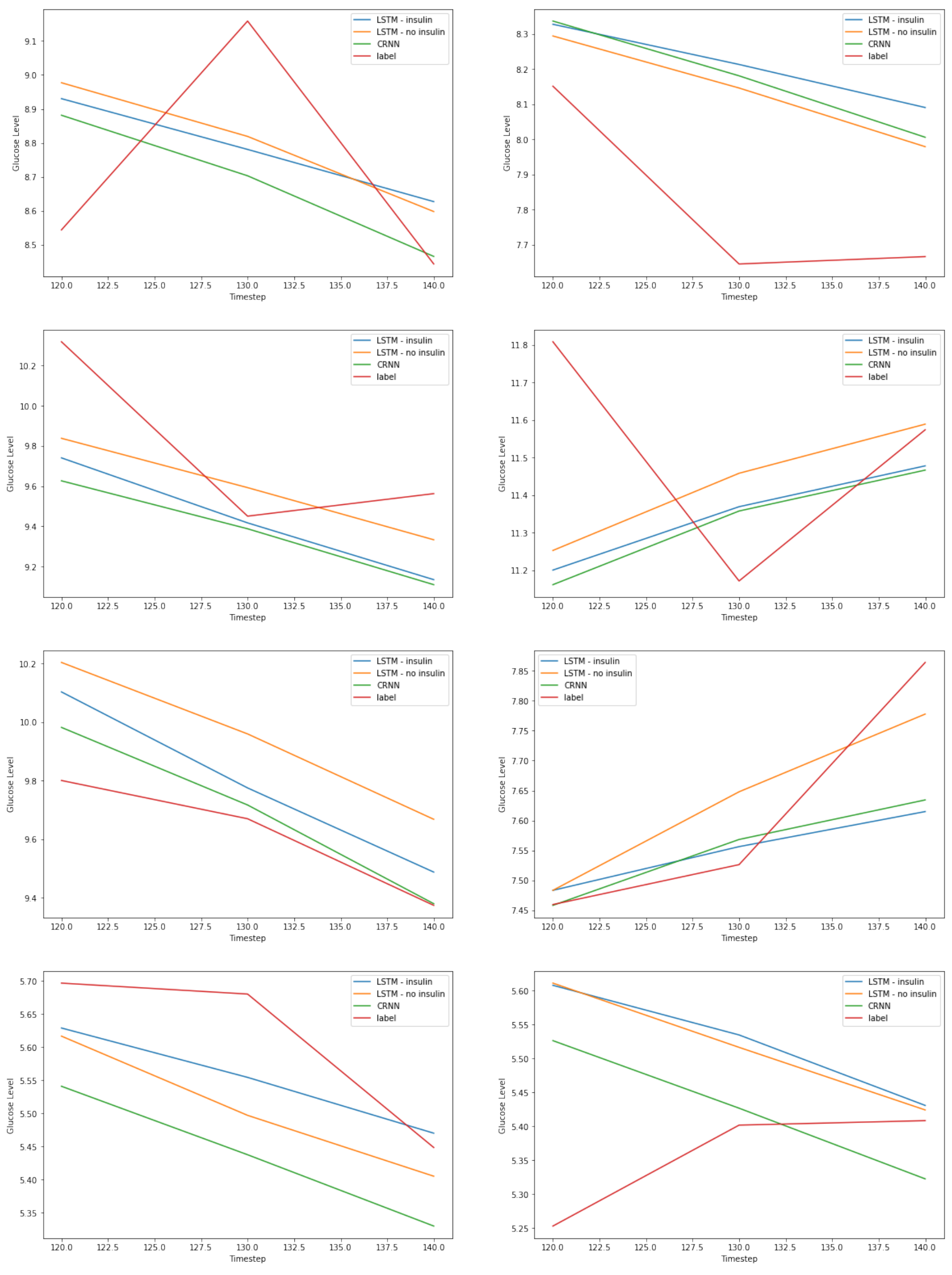


Figure 14: Glucose Level vs Timestep, for the LSTM (with and without insulin) and CRNN Models; focusing on the final three time steps

5.3 Varying input time series lengths

The minimum input time series length to produce close to optimal results is about 110 minutes; close to 2 hours. It is also worth noting that the RMSE is still quite good even with only 10 minutes worth of data (a single time step).

5.4 CRNN

The RMSE of the CRNN was found to be very slightly worse, about 0.01 mmol/L higher, compared to the simple LSTM model. This shows that a simple model for predicting glucose levels could be sufficient for good performance, though many more advanced models are yet to be tried that could improve on these results. The LSTM is a good base from which a more advanced model could be built around however, so more research needs to be done here.

5.5 Compare all models in one plot

In figures 13 and 14 it is shown that the predictions for all three models are very similar, and there is no obviously better model. It is also worth noting that the predictions from all three models are very linear, with very little change in gradient from point to point. This shows that these models might not be very good at predicting very sudden spikes. Maybe in future a smaller resolution should be attempted in order for the model to be better at predicting the sudden changes and finer details of the true glucose level curve. It could even be an important subject of study, as a sudden spike in glucose levels could cause problems for the patient. Maybe a better measure of success for such NN models is to count the number of failures in detecting hypoglycemic and hyperglycemic events, and to make sure that they are as small as possible. This would align more closely with the overall aims of the project, I believe.

6 Professional Issues

One particular problem in machine learning, particularly when using data involving patient information, is the issue of privacy. Patient data can contain personal information, so it is important that the patients understand and give permission for how their data is used. In this project, all the data was created using artificial patients, in other words via a computer model. Therefore, no real patient data was used in this project, meaning that patient permission was not needed. So privacy was not an issue for this project.

However, privacy can be a very big problem in research projects like this and in machine learning in general. For example, the Facebook – Cambridge Analytica data scandal [2] is known to be the one of the largest data breaches ever. Roughly 50 million Facebook profiles were extracted for Cambridge Analytica, most of which was used without their user's permission. Cambridge Analytica were the company who worked with Donald Trump's election team and the Brexit campaign, and used this data to target voters based on this information in order to sway voters in a more efficient political campaign.

The data was collected through an app, where hundreds of thousands of users participated in paid surveys and agreed to have their data collected for academic use. However, the test taker's Facebook friends also had their information taken which lead to the tens of millions of Facebook profiles having their data taken and stored by Cambridge Analytica. User collection of friend's data is only allowed to improve user experience in the app, and not for being sold on and used for advertising, according to Facebook's platform policy. So Cambridge Analytica clearly crossed this boundary and illegally harvested and stored the data of tens of millions of Facebook users. In summary, information from tens of millions of Facebook profiles were taken and used without their user's permission.

This data was then used to build a system that could profile individual US voters and target them with personalised political advertisements. Politicians have since called for more control over data usage, specifically for political advertising.

A letter was sent by Facebook's lawyers telling Cambridge Analytica telling them that since they acquired data without permission, it cannot be used legitimately and must be deleted immediately. They did not follow up that letter when it went unanswered for weeks, and they did not conduct forensic checks on Cambridge Analytica's computers or storage. Facebook also failed to be open and transparent about the data breach. They said that data was accessed correctly but used incorrectly, so they could have felt not responsible for the actions of Cambridge Analytica. All this shows that Facebook felt a lack of accountability for their user's data, and thus betraying the trust of these users.

There has been a failure of accountability from Facebook and also bad ethical conduct from Cambridge Analytica throughout the events of this scandal. Clearly there needs to be a sense of responsibility from both the group holding the data for how it is distributed, and the group extracting the data and then using it responsibly. If everyone is taught and understands the importance of data privacy, along with stricter regulations and more government control, then big data breaches like this could become less common.

In the example above the data was used for political campaigns and targeted advertising, which could be thought of a particularly self-serving way to use data that is taken and used illegally. What about if that data is used for medical research, and thus would aim to serve the patients it is taking the data from, but without their permission?

This is a particularly interesting issue, especially since data acquisition is the biggest problem of creating machine learning algorithms in medicine which is due to the strict privacy and security rules in place about such data [6]. Also, the efficient use of patient data would improve medicine and help to save lives, so one could say that they could breach the rules of legal data usage for the greater good. It is however still unethical to access people's private information. What really needs to be done is to get permission from patients to use their data, explaining exactly what it will be used for. This could be tricky though as there could be limited opportunity for patients to give permission for data usage. Maybe a regular email sequence or programme that allows researchers to ask for data permission and users can then answer after reading where their data will be used. There must be a rigorous system in place for both the group that stores the data and the group that extracts the data so that no big data breaches occur like in the Facebook – Cambridge Analytica data scandal. This would take some time to set up, so therefore there is no guarantee that big data breaches would not occur in the near future. This means that making legal access for large amounts of data for medical research is still difficult. Work must be done to improve the control of data so that it can be used ethically and for the good of society.

For now, small data sets from real patients or data sets from patient models (as used in this project) must be used in order to conduct medical research involving data and machine learning.

In conclusion, I think it is important to keep restrictions on data. Data can be used for projects, but we should state clearly to the user how their data can be used for specific tasks such as medical research and to ask for their permission in a quick and easy way. There must be strict regulations on data usage that must be enforced properly by the government.

7 Self Assessment

Overall the project did not go as well as I would have hoped, but it is still a decent project. I was expecting myself to have done more work in creating and improving the NN architecture and to have done some extra work, such as using inductive conformal predictors and deriving optimal insulin control policies.

I did learn to break up a large project like this into smaller tasks and I set myself deadlines in my project plan. However, I failed to stick to my self-set deadlines within my project plan, which lead to this problem of running out of time. The reason for this is that I did not regulate my day-to-day behaviour and stick to a strict schedule, such as working from 9am until 5pm. This is essential when there is no definite, short-term deadline like there is for most of my assignments. These deadlines push me on and motivate me, but for some reason if I set them myself it does not have the same effect. It was probably because there was always a chance to finish things after the deadlines, since they were not real. So I have to consider how else I should motivate myself. Maybe I should have printed out my project plan and had the deadlines in clear view, so that they could stay more clearly in my mind as dates that I should really stick to.

Additionally, the lack of a constant working environment meant I could never maintain a constant work flow, since I was moving to a lot of different accommodations. I believe that this lack of consistency meant that I procrastinated too much, and thus I could not complete the project work to the level that I was originally planning. In future I should make sure that I have a better schedule, for example by timing how long I should work for. Maybe I should've timed 30 minutes and then I would be allowed to take a break afterwards. This would get me to start working and stop procrastinating since it is such a short time to work. It is something that I have started doing towards the end of this project, in fact.

I am quite pleased with the background reading of this project, particularly with what I have written in this dissertation. I think I have interpreted and explained the sources well enough for a lot of people to understand, while still trying to be concise enough. I am also reasonably happy with my model, as it gives a reasonably good performance despite its simplicity.

I think I should have presented the results in a better way, particularly the plots comparing glucose prediction against glucose labels. I think that I could have done a series of predictions along the plot to show the results over a longer period of time than just a 30 minute prediction at one time step.

If I had more time, I would have continued trying to improve my model and experimenting with different architectures, and then I would have tried to either analyse using inductive conformal predictors or to create a system to derive optimal insulin control policies.

8 How to use my project

First of all, Tensorflow 2.0 is required as the project uses Keras to create the NN models. Numpy, Pandas, SciKit-learn, and Matplotlib are all required. All files are written in Python 3.7. Additionally, the data will have to be downloaded from $https://github.com/nicopao/CGM\_prediction\_data$. All the files needed to run the code for this project are located in the /Program directory. Note that the results of this program can be found in the /figs directory.

The very first file that needs to be run is the *init.py* file, which imports all necessary libraries, initialises a random seed to retain some reproducibility, defines all custom functions, imports the data, initialises some important variables, and preprocesses the data. Note that the file path will need to be set to wherever the data has been saved to when the data is read in, in the Import Data section of the python file. For my computer it was $./CGM\_prediction\_data - master$.

After this the models can be trained using the $train\_*.py$ files, which saves the trained model and its weights into a JSON and HDF5 file respectively. Some pretrained models, along with their respective weights, have been provided to avoid the training step if desired.

The saved models can be loaded and tested using the $test\_*.py$ files. When run they output a series of plots comparing the predictions with the true labels of a few samples, along with the evaluation of the model on the test set.

There is a python file, *overall_plots.py*, which outputs plots comparing the predictions of all three models with the true labels for a few samples. One set of plots includes the timesteps preceding the timesteps to predicict, the other does not.

Finally there is a file called $different\_time\_series\_lengths.py$, which experiments with different input time series lengths. It then goes on to produce the test RMSE against input time series length plot.

References

[1] Charlotte K. Boughton and Roman Hovorka. Advances in artificial pancreas systems. *Science Translational Medicine*, 11(484), 2019.

[2] Carole Cadwalladr and Emma Graham-Harrison. Revealed: 50 million facebook profiles harvested for cambridge analytica in major data breach. `https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election`, March 2018. Accessed: 2020-08-30.

[3] Nagesh Singh Chauhan. Introduction to artificial neural networks (ann). `https://towardsdatascience.com/introduction-to-artificial-neural-networks-ann-1aea15775ef9`, October 2019. Accessed: 2020-08-24.

[4] Hongkai Chen, Nicola Paoletti, Scott A. Smolka, and Shan Lin. MPC-guided Imitation Learning of Neural Network Policies for the Artificial Pancreas. *arXiv e-prints*, page arXiv:2003.01283, March 2020.

[5] Editor. Treatment for type 1 diabetes. `https://www.diabetes.co.uk/treatment-for-type1-diabetes.html`, January 2019. Accessed: 2020-07-09.

[6] Hayk Gharagyozyan. A practical application of machine learning in medicine. `https://www.macadamian.com/learn/a-practical-application-of-machine-learning-in-medicine/`, February 2019. Accessed: 2020-08-30.

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

[8] K. Li, J. Daniels, C. Liu, P. Herrero, and P. Georgiou. Convolutional recurrent neural networks for glucose prediction. *IEEE Journal of Biomedical and Health Informatics*, 24(2):603–613, 2020.

[9] Katrin Lunze, Tarunraj Singh, Marian Walter, Mathias D. Brendel, and Steffen Leonhardt. Blood glucose control algorithms for type 1 diabetic patients: A methodological review. *Biomedical Signal Processing and Control*, 8(2):107 – 119, 2013.

[10] Chris Nicholson. A beginner's guide to convolutional neural networks (cnns). `https://wiki.pathmind.com/convolutional-network`. Accessed: 2020-08-24.

[11] Christopher Olah. Understanding lstm networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, August 2015. Accessed: 2020-08-17.

[12] B. Sohlberg and E.W. Jacobsen. Grey box modelling – branches and experiences. *IFAC Proceedings Volumes*, 41(2):11415 – 11420, 2008. 17th IFAC World Congress.

[13] G. Sparacino, F. Zanderigo, S. Corazza, A. Maran, A. Facchinetti, and C. Cobelli. Glucose concentration can be predicted ahead in time from continuous glucose monitoring sensor time-series. *IEEE Transactions on Biomedical Engineering*, 54(5):931–937, 2007.