Predicting Blood glucose levels with Phased LSTM

Jimmy Fagerholm 40853
Software engineering: Predicting blood glucose levels using Phased LSTM
Supervisor: Sepinoud Azimi
Faculty of Science and Engineering
Åbo Akademi University
2021

Abstract

In this thesis, a phased long short-term memory model is implemented to predict the blood glucose level in patients with type-1 diabetes with a 30-minute forecast. We will continue previous work by extending the standard long short-term memory deep neural network model with a phased LSTM cell. The model is trained on the OhioT1DM dataset from the BGLP challenge. This study will try to solve a standard LSTM model's bottlenecks by using a phased LSTM model. Furthermore, an attention-based phased LSTM model will be implemented to achieve explainability to this research topic's models. An attention-based phased LSTM model performs best if trained on a larger dataset. The performance is on par with previously implemented methods for predicting blood glucose levels from the BGLP dataset.

Keywords: Blood glucose level prediction (BGLP) challenge, Long short-term memory (LSTM), Phased long short-term memory, Explainable AI (XAI)

Contents

1 Introduction **1**

1.1 Previous work 2

1.2 Problem statement 3

2 Deep learning **4**

2.1 Machine learning 4

2.1.1 Regression 4

2.1.2 Deep learning 5

2.2 Recurrent neural network 6

2.2.1 LSTM — Long Short-Term Memory 7

2.2.2 Phased LSTM 9

2.3 Explainable AI 10

2.3.1 Attention Layer 11

3 Method **13**

3.1 OhioT1DM Dataset 14

3.2 Data preprocessing 15

3.3 Long Short-Term Memory 16

3.4 Phased Long Short-Term Memory 17

3.5 Attention-based Phased LSTM 19

3.5.1 Attention layer output 19

3.6 Training the model and updating weights 20

4 Results **21**

4.1 Benchmark model 21

4.2 Phased LSTM model 24

4.3 Attention Phased LSTM model 25

4.4 Interpretation 31

5 Discussion **34**

6 Conclusion **37**

7 Svensk sammanfattning **40**

7.1 Inleding 40
7.2 Metod 41
7.3 Resultat 42
7.4 Diskussion och slutsats 43

Chapter 1

Introduction

With a growing demand for solving complex problems with deep neural networks, the University of Ohio created a Blood Glucose Level Prediction (BGLP) challenge. The BGLP challenge was initially released in 2018 to facilitate research on blood glucose level prediction. This challenge aims to create a powerful machine learning model to forecast blood glucose levels. Alongside this challenge, Ohio University released the OhioT1DM dataset, a dataset with real-time data from 6 different patients with type 1 diabetes. The dataset contains data from eight weeks of continuous blood glucose level, insulin level, physiological sensor, and self-reported life-events monitoring [1]. In 2020 the university added data from six more patients to the OhioT1DM dataset. These six new patients have more features recorded, which the University of Ohio hoped would spark interest in the research topic.

This master's thesis will address a long short-term memory model's issues when training and predicting on sparse data. We hypothesize that a phased long short-term memory model will achieve the same or better level of accuracy on a dataset that has not been preprocessed. Previous research has shown that a phased long short-term memory model converges faster than a standard long short-term memory model and works better on sparse data. With the help of an attention layer in our model, we hope to introduce explainable artificial intelligence (XAI) to this research topic. XAI could enhance feature extraction and model tuning, which would speed up research on this topic. There are few published research papers as of early 2021 that would explain which features are the most important in the OhioT1DM for training a Recurrent neural network.

This study will implement three different model architectures and train models from these architectures on the six original patients from the OhioT1DM dataset. The first model architecture is a copy from earlier research. This model will become our benchmark model to verify that our preprocessing has been done correctly and verify our data extraction method. The second model will have the same number of layers as the first model except that the long short-term memory layer will be swapped for a phased long short-term memory layer. The second model should have a better result than the benchmark model and converge faster while training. The third and final model architecture will extend the second architecture by adding an attention layer as the first layer to the models. The attention layer produces an attention vector from which observations can be made to what feature correlates best to the predicted output. With the help of the attention vector, we hope to provide this research topic with explainability and speed up future research in this area.

1.1 Previous work

There is a wide variety of previous work on this dataset, where new methods are tried out to creating the most powerful model for predicting BGL. From classical time-series approaches to more sophisticated techniques like the long short-term memory (LSTM) model.

In a research paper written by Jinyu Xie and Qian Wang, two types of methods were tested to benchmark machine learning approaches with time-series approaches on the BGLP challenge. Their results suggest no significant difference between the machine learning models and the classic ARX model. Their methods included comparing the classical autoregression with exogenous inputs (ARX) model, Huber Regression, Ridge Regression, Elastic Net Regression, Lasso Regression, Support Vector Regression with Linear Kernel, Support Vector with Radial Basis Kernel, Random Forest, Gradient Boosting Tree. Furthermore, they used two deep learning methods: a standard LSTM model and a Temporal Convolution (TCN) model [2].

In another research paper proposed by El. Idriss et al., an LSTM neural network is proposed with two fully connected layers and a single LSTM layer for this same challenge. This model was compared with another existing LSTM and an autoregression (AR) model. The results show that their LSTM outperforms the contending LSTM model for all the patients and 9 of 10 patients for the AR model [3]. In the research paper, El Idriss et al. aimed to identify the best configuration of parameters for the LSTM model by doing a grid search. In grin search, multiple parameter values are chosen, and a model is created from each of the possible combinations. The resulting models are then trained and compared with each other to find the best set of parameters; read more about the parameters selected in Chapter 3. The performance of this model is the best we have found in a research paper. Therefore, this will be the first model architecture and our benchmark for this thesis study.

A research paper by Robert Bevan and Frans Coenen investigated the need to train individual models (i.e. per patient) to predict blood glucose levels with the help of the OhioT1DM dataset. This research paper compared various model classes such as linear models, feed-forward models and LSTM models. Furthermore, they incorporated an LSTM model with an attention mechanism to predict blood glucose levels. To find the best possible model, they tuned the model with multiple hyperparameters and experimented with the history length passed to the model while training. The best result was achieved with a history of 30 minutes (six samples). Furthermore, the attention mechanism provided better results than the other models and models trained on all of the data outperformed models trained individually. [4] The same data processing techniques will be used in this study as Bevan and Coenen used in their experiments. *(https://github.com/robert-bevan/bglp)*

1.2 Problem statement

Previous work shows an extensive job of preprocessing the data. The LSTM model requires the data to be processed, and if possible, without missing values. With our approach, we do not need to process the data as much. Therefore, making the model much more valuable to real-life scenarios and speeding up the implementation of the models. The benchmark model achieved a root mean square error (RMSE) value of the predicted blood glucose level down to as low as $12.38 mg/dl$ [3]. In this study, we aim to achieve the same level of accuracy with less to no preprocessing.

Previous research used an attention mechanism to perform better when predicting blood glucose levels for the BGLP challenge. Another benefit of the attention mechanism is the attention vector provided by the attention layer. The attention vector describes which features are considered when the prediction is made. Previously explainable AI (XAI) has been a black box for this kind of research. By implementing an attention-based PLSTM model, we hope to understand better which features affect the prediction the most. With explainability, future research can be weighted towards a more narrow viewpoint, achieving even better results.

Chapter 2

Deep learning

2.1 Machine learning

Deep learning has evolved into the new norm for machine learning and artificial intelligence thanks to decreased computational power. Deep learning is a form of machine learning where a deep neural network, a model, is trained on a dataset to predict or estimate values or classifications for new unexplored data that resembles data from the dataset trained on. These models can be used in audio recognition, computer vision, natural language processing, and much more. In this paper, we are using a deep learning model in a regression problem. The model will predict the blood glucose levels with the help of a dataset on patients with type-1 diabetes.

2.1.1 Regression

Regression is a statistical method used to estimate a dependent variable's relationship to one or more independent variables. The most common type of regression method is linear regression. In linear regression, the prediction is computed by the weighted sum of the input features, plus a bias. The linear regression can be written in many ways. However, the most common is the vector form, as shown in equation 2.1, where $\hat{y}$ is the predicted value, θ the angle and x the feature variable.

$$\hat{y} = h_\Theta(x) = \Theta * x \tag{2.1}$$

The angle of θ is referred to as the weight. When training a model, the weight is tweaked to fit as close as possible to the actual value; Figure 2.1 is a visual example of linear regression.

Computers are accurate at producing the correct weights in regression, given the input features and output value, even with polynomial regression and multi-dimensional regression. Therefore, regression is used as the building block for many machine learning approaches.

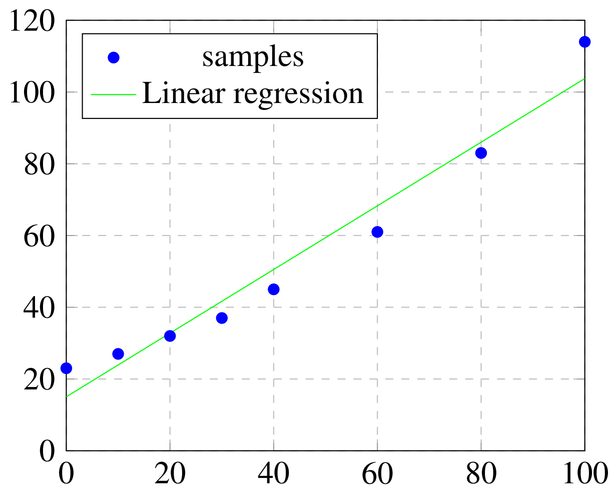


Figure 2.1: Linear regression example

2.1.2 Deep learning

A subsection of machine learning is deep learning. In deep learning, a neural network is created with multiple layers and thousands of neurons. Each layer consists of one or more neuron, and each neuron is assigned a weight. The weight and the input value is passed through an activation function, which determines if the neuron is active or not. An active neuron passes its value to the next neuron as the input value, and a deactivated neuron will block the value to be passed forward. The neuron value is calculated with the help of the input value, the neuron weight, plus a bias.
As stated earlier, computers are very efficient at tuning these weights to produce the desired output. The desired output is the output of the last layer. An error is calculated with the help of a loss function; the error is the average difference between the last layer outputs and the actual values of a batch of samples. By tracking the loss function over time, we can determine the rate at which the model is approaching the optimum value.
If a graph is drawn from the loss function, the angle of the slope would be the gradient. The gradient determines the amount each weight is adjusted when going through the model backwards, backpropagation. The steeper the gradient is, the more each weight is adjusted. Adjusting these weights based on the known output value given the input values is called training the network.

When training, the network is sent samples and given the correct output value, known

as the training data. These samples are passed through the network multiple times, each time calculating the new error value with the help of a loss function. The gradient of the loss function is then backpropagated through the model, which tunes the weights in each layer. This sequence is called an epoch. Once the training data have been used to tune these weights, the network can predict an unknown value given new, never seen before input values. This is the core function of a neural network. The more a network is trained, the better it will become at predicting the output.
If the number of samples is too low, the network will learn the training data too well, also known as overfitting the data. If the number of samples is too high, and the network is not trained for enough epochs, the model has not learned enough to predict a reasonable output; this is called underfitting the data. A balance is found through trial and error.

2.2 Recurrent neural network

Recurrent neural networks introduce an extension to feedforward networks, where the previous step is fed as an input to the current step. This allows information to persist and the network to consider earlier data. With the help of this addition, neural networks can build connections between data in time—for example, a network with relations between words in language modelling and translation. Recurrent neural networks, or RNNs, allow the information to be passed inside the node from one step of the network to the next, as shown in Figure2.2.

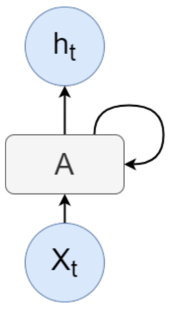


Figure 2.2: Recurrent neural network node

The drawback of a recurrent neural network is that it suffers from short-term memory. When a sequence is too long, connections between input values from the beginning are lost due to vanishing gradients. Each time the gradient is passed through a layer, it becomes smaller, and when it is passed through enough layers, it vanishes. The earlier layers' weights are not updated because of the vanishing gradients, thus forgetting the first values passed to the model.

2.2.1 LSTM — Long Short-Term Memory

A long short-term memory cell, or an LSTM cell, is a recurrent neural network cell. This cell was created to solve vanishing gradients' problem and allow long-term connections to persist throughout the model. These cells have internal gates that regulate the flow of data from cell to cell. Figure 2.3 shows a graphical overview of how an LSTM cell is built.

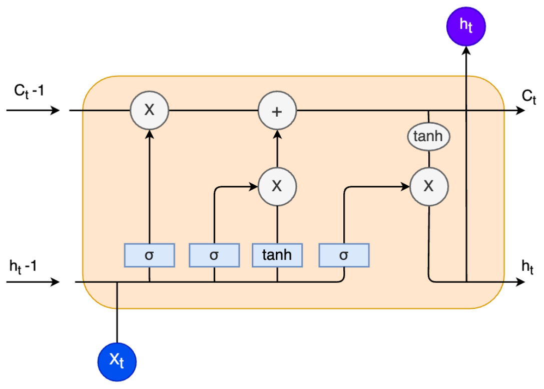


Figure 2.3: LSTM Cell

Where X_t is the input value and h_t the hidden state of the cell, the cell state is named C_t. h_t and C_t are passed to the next neighbouring cell. h_{t-1} is concatenated with the input value X_t to form a vector pair used throughout the cell; let us call this the input vector.
The input vector is first passed through the forget gate, as shown in Figure 2.4. The information is passed through a sigmoid function that outputs a value between 0 and 1; 0 means to forget, and 1 means to keep the value. This gate allows information to be forgotten, which helps with long-term connections between data.

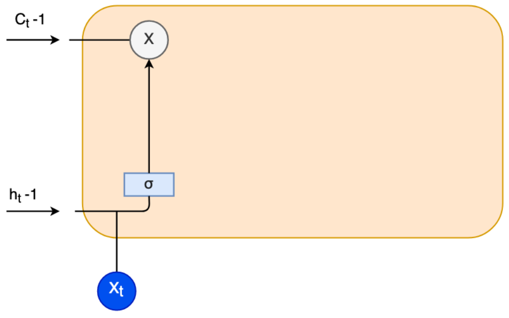


Figure 2.4: LSTM - Forget gate

The next gate in the LSTM cell is the input gate. The input gate is used to update the state of the cell, shown in Figure 2.5.

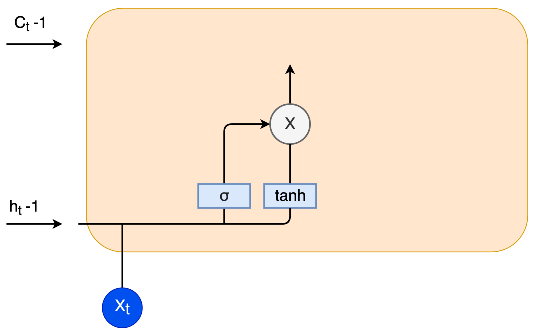


Figure 2.5: LSTM - Input gate

The input gate consists of two functions. The first function is the same as the forget gate, and it works the same way, dictating if the value should be kept or forgotten. The second function is a tanh function which transforms the vector into a value between -1 and 1. This function is used to help regularize the network. The output of these functions is then multiplied by each other. Figure 2.6 is a representation of the tanh function.

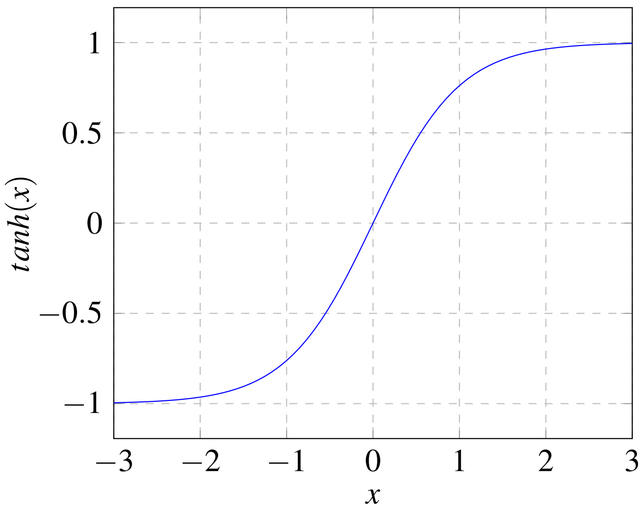


Figure 2.6: Tanh function

The cell state is updated with the help of the forget gate and the new input value. The old cell state, C_{t-1}, is first multiplied by the forget gate; values closer to 0 will drop the current state. Then a pointwise addition is done with the input vector and the current state. The output of this is the new cell state C_t.
The last gate in the LSTM cell is the output gate, shown in Figure 2.7. The next hidden state is calculated through the output gate. The input vector is passed through a sigmoid

function (same as the forget gate), the current state is passed through a tanh function. The output of the two functions is then multiplied to form the new hidden state h_t. The output gate is also used for predictions.

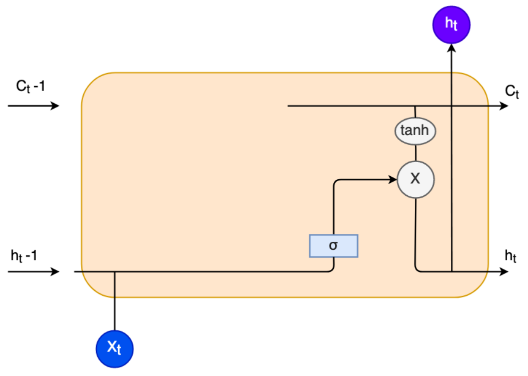


Figure 2.7: LSTM - Output gate

2.2.2 Phased LSTM

The phased LSTM cell extends the LSTM cell by adding a new time gate k_t, sometimes called the Kronos gate. A rhythmic oscillation controls this gate, much like a sinus wave, and the oscillation determines if the cell states C_t and h_t can be updated. The oscillation is determined with three different parameters; The first parameter, τ, controls the real-time period of the oscillation (how fast it oscillated). The second, r_{on}, controls the on/off ratio of the gate. The third, s, controls the phase shift of the oscillation to each phased LSTM cell. All these parameters can be trained during the training of the model. [5]. Figure 2.9 contains a schematic overview of the phased LSTM along with a graph describing how the state is updated only when k_t is active. The internals of a phased LSTM cell resembles the standard LSTM cell, with the addition of a gate at the end of the cell, the Kronos gate. Figure 2.8 represents the phased LSTM cell, where the Kronos gate is added to the end of the cell compared to the standard LSTM cell.

The advantage of a phased LSTM cell is that it converges faster on long input sequences, and it learns faster than regular LSTM or other RNNs. The normal LSTM cell struggles with sparse computation, while the phased LSTM cell allows for sparse computation [5]. Furthermore, the sleep period acts like a dropout function, regularizing for robustness. A normal dropout function will randomly ignore a percentage of the input values. Thanks to the sleep period, backpropagated gradients are preserved across many time steps, resulting in fast learning on long sequences. With the correct loss function, the phased LSTM layer

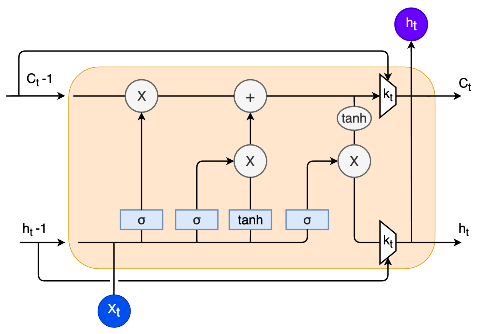
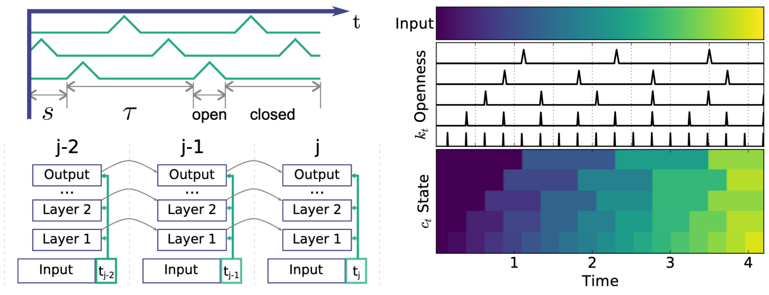


Figure 2.8: Phased LSTM Cell



(a) (b)

Figure 2.9: Diagram of Phased LSTM behaviour. (a) Top: The rhythmic oscillations to the time gates of three different neurons; the period $/tau$ and the phase shift s are shown for the lowest neuron. The parameter r_{on} is the ratio of the open period to the total period τ. Bottom: Note that in a multilayer scenario,the timestamp is distributed to all layers which are updated at the same time point. (b) Illustration of Phased LSTM operation. A simple linearly increasing function is used as an input. The time gate k_t of each neuron has a different $/tau$, identical phase shifts, and an open ratio r_{on} of 0.05. Note that the input (top panel) flows through the time gate k_t (middle panel) to be held as the new cell state c_t (bottom panel) only when k_t is open.
Figure and caption copied from [5].

solves the issues with vanishing or exploding gradients with the standard LSTM layer.

2.3 Explainable AI

Explainable AI or XAI is a subsection of machine learning where the goal is to explain a model's decisions. XAI is essential to not only the end user but also the model developer. With the power of explaining the model's behaviour, design decisions can enhance the model's ability to predict the correct output. The end user should also know the underlying reason why a model predicts a given output to utilize the model in a better way [6]. One way of achieving explainability with deep learning is to implement an attention layer. Explainability is the ability to explain a prediction or an action of a model from a more technical perspective to a human.

2.3.1 Attention Layer

There are two major types of attention; the first type was introduced by Dzmitry Bahdanau and is usually referred to as the Bahdanau type, sometimes called additive attention [7]. The second type, referred to as the Luong type, was introduced by Thang Luong, sometimes called multiplicative attention, and was built on top of the first attention mechanism type proposed by Bahdanau [8].
The attention mechanisms were initially built as a method to improve the accuracy of machine translation. Research done by Bahdanau and Loung shows that the attention mechanism improves performance on machine translation tasks while also interpreting the model. The first benefit was achieved from the attention layer building global attention over the whole sequence. Global attention is something LSTM and Phased LSTM layers do very well already. The second benefit, interpretation of the model, where the attention mechanism provides explainability of the dataset when training and predicting.
The attention layer produces an attention vector a_k, where the input features x_k are multiplied by the attention weights W_k across time steps:

$$a_k = softmax(W_k x_k)$$

where every x_k represents a single feature over the entire dataset. Furthermore, the input features x_k passed through the attention layer are weighted down with the attention vector a_k before it is available to the next layer:

$$y_k = a_k * x_k$$

Explainability can be achieved by interpreting the extracted output vector a_k. The attention vector describes how much each of the input features provides to the predicted values of our model. The Bahdanau type derive the attention at $t-1$ and concatenated that with attention at t, while the Luong type only considers the attention at t.

Figure 2.10 describes the process where five input features are sent to a machine learning model, and the attention layer calculated the attention vector from the input features. In this example, the first and last features have the highest attention value, 0.42 and 0.31. This example shows that these two features are most important when predicting the output value.

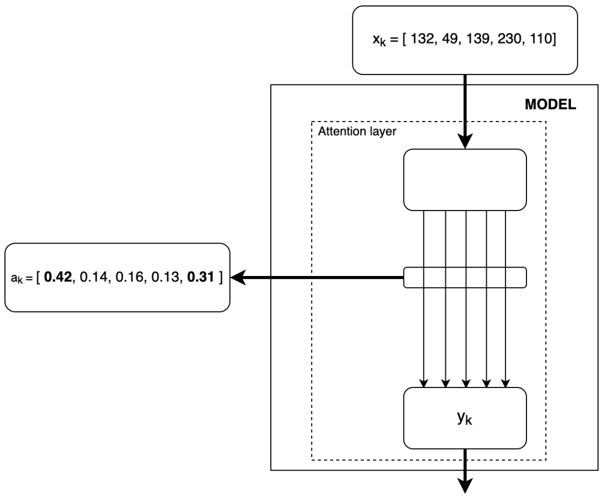

x_k = [132, 49, 139, 230, 110]
MODEL
Attention layer
a_k = [0.42, 0.14, 0.16, 0.13, 0.31]
y_k

Figure 2.10: An attention vector example. Five feature values are passed to a machine learning model where the model's attention layer sets the weights on which features are important when predicting the output value. A higher value indicated more attention is used on these features when predicting.

Chapter 3

Method

Previous research has proven the phased long short-term memory (PLSTM) model to be more effective in converging than the standard long short-term memory (LSTM) model. By implementing a phased gate to the LSTM cell, the PLSTM model will converge faster on sparse data; this was discussed in Section 2.2.2. This section will describe the implementation and the design decisions and their impact on the final results.

In this study, a PLSTM model is used, which differs from the standard LSTM model in that it should not require perfectly sequential data, thanks to the forget gate discussed in Section 2.2.2. Multiple previous research papers state the importance of data processing for the standard LSTM model before the dataset is ready to be used if the dataset contains missing values [3] [5]. When using a PLSTM model, the amount of data processing required is reduced. Choosing a PLSTM model thus simplifies the preprocessing of the data. For the standard LSTM model to perform well, the dataset should not be sparse or contain missing data in-between samples. For many of the patients' data in the OhioT1DM dataset, the blood glucose level is sparse and requires processing for the LSTM model to perform well. The dataset also contains many features which do not have data at all for many of the patients. The second model architecture uses a PLSTM model that does not depend on the data to be perfectly sequential, without any missing samples in between.

Furthermore, an attention layer will be added to the PLSTM model. With the help of an attention layer, we can extract the importance of the single features used from the dataset. The attention layer will output a layer vector with values that correspond to the significance of every feature. The attention layer will add explainability to both phased long short-term memory models for this dataset.

For this study, three different model architectures will be trained. The first model architecture is a benchmark model with the same architecture as previous research. The latter two models architectures extend the benchmark model with a different LSTM layer, and the last model architecture adds explainability to the models.
Previous research has shown excellent accuracy on the OhioT1DM dataset with extensive data mining and hyperparameter tuning [3]. Our suggestion for reducing the need of preprocessing is to replace the long-short term memory layer with a phased long short-term memory layer, discussed in Section 2.2.2. *Hyperparameters* are the parameters used to control the learning process of a machine learning model. The same parameters will be used as the ones selected for our benchmark model.

3.1 OhioT1DM Dataset

The University of Ohio released the OhioT1DM dataset in 2018. As of 2020, this dataset contains eight weeks (about two months) of continuous data from 12 patients with type 1 diabetes. *Diabetes* is a chronic illness caused by a disorder in glucose metabolism. There are mainly two types of diabetes. Type 1 Diabetes Mellitus (T1DM), when the pancreas does not produce enough insulin, and Type 2 Diabetes Mellitus (T2DM), which results from ineffective use of insulin [3].
For anonymity, each patient is assigned a random id; the id is referenced in the thesis for replicability. Each patient has self-reported events such as sleep quality, stress levels, and meals, with estimated carbohydrate intake and sensory data. There are 19 features recorded. Features with low levels of data and completing missing data is excluded, leaving us with five features. For many of the patients, some of these features contain no data, and other features have sparse data on which our neural network model will be trained. The five features selected for this study is the following; The blood glucose level (BGL), the self-monitored value of BGL through finger stick samples, background insulin infusion rate (basal), insulin delivered to the patient (bolus), and the meals the patient has eaten with the estimated carbohydrates, which affects the blood glucose level. Figure 3.1 visualizes the features in the dataset for patient 544. This figure shows the recorded BGL, finger-stick level, basal, and bolus values over eight weeks for patient 540.
Each patient used either a Medtronic 530G or 630G insulin pump for monitoring the insulin values. These insulin pumps allow the bolus to be either injected in one dose, a regular injection—alternatively, dual wave, where insulin is first injected in one dosage and then injected over time [9]. These different modes are stated as a type in the dataset and accounted for during the preprocessing of the dataset.

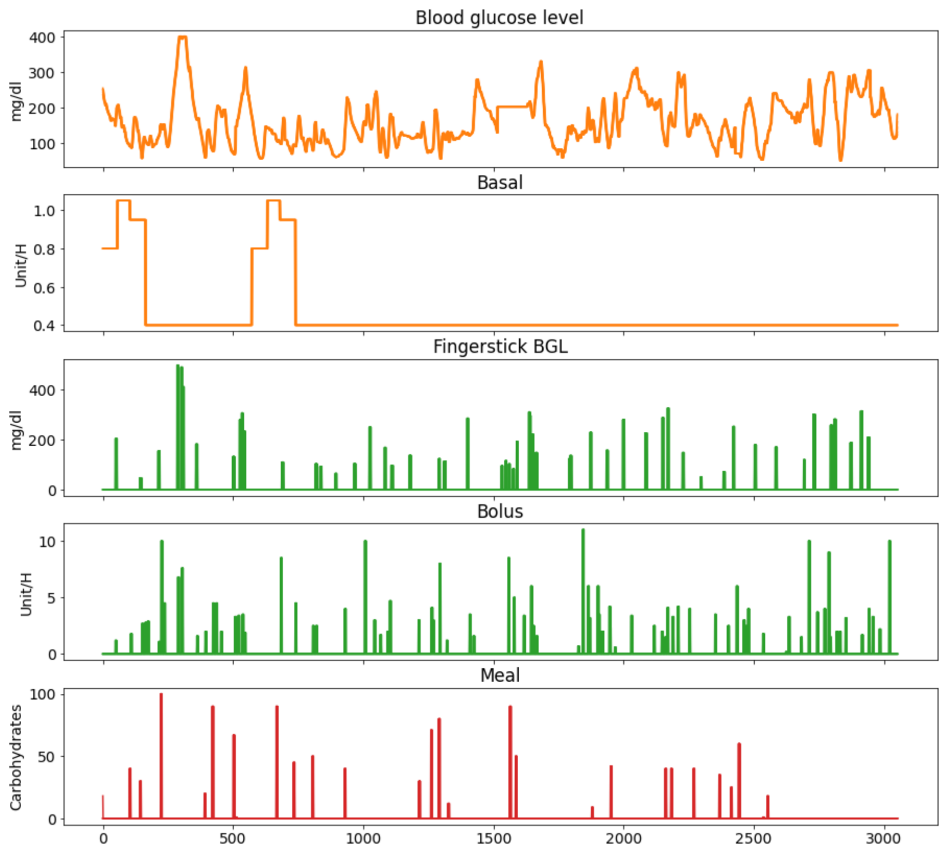


Figure 3.1: Patient 544, Dataset visualized

Later a sixth feature was created, a time-of-day feature. This feature was used in earlier research with good results . This feature is created by looking at the recorded time of the BGL level. Table 3.1 describes how the feature was encoded to the dataset. Timestamps between 06:00 and 11:59 are considered in the morning, 12:00 to 16:59, the afternoon, 17:00 to 21:59 in the evening and 22:00 to 05:59 at night.

Time	timeslot	Number seen in the dataset
Between 06:00 and 11:59	Morning	1
Between 12:00 and 16:59	Afternoon	2
Between 17:00 and 21:59	Evening	3
Between 22:00 and 05:59	Night	4

Table 3.1: Description of how the virtual time-of-day feature was encoded

3.2 Data preprocessing

For a deep neural network to work with significant results, some preprocessing of the dataset is conducted. The dataset is provided as a zip file with a single file for each pa-

tient. The first step to preprocessing is to extract the XML files into NumPy arrays for the Python program to interpret them easily. Each event contained a timestamp as a reference to when the measurement was taken. The output of our model will be blood glucose levels in the future based on the values from the different features in the dataset. The first feature that is extracted is the blood glucose level. The timestamps from this feature will be the reference that is used to match the other features. We know that the blood glucose level is measured at a 5-minute interval.
Figure 3.2 represents samples with only the blood glucose level as a feature.

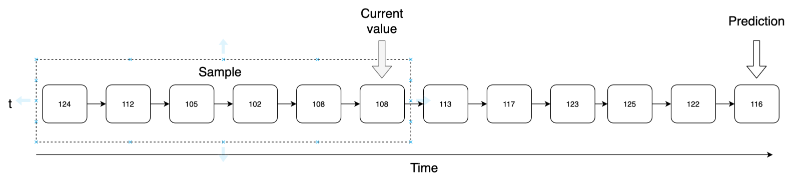
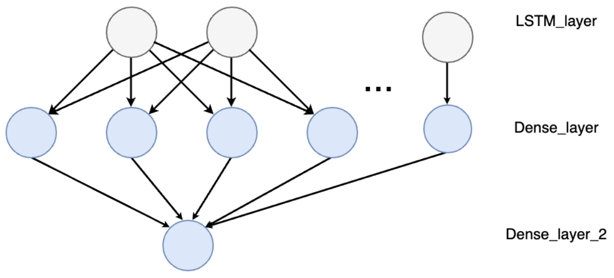
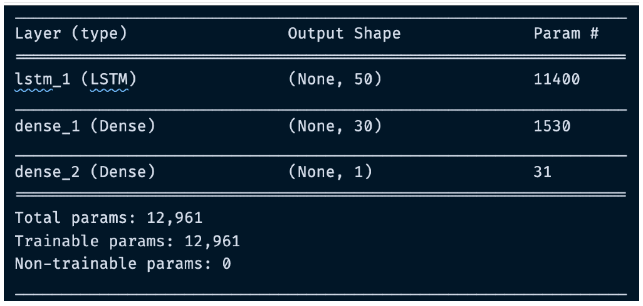


Figure 3.2: Preprocessed sample

3.3 Long Short-Term Memory

El Idriss et al. have done extensive research on data mining for diabetes self-management [10]. With the help of their research El Idriss, Ali Idri, and Ibtissam Adnane researched the OhioT1DM dataset with a long short-term memory (LSTM) model. They performed a grid search with different parameters to identify the best configuration for this neural network. Grid search is usually done to find the best performing model.
In grid search, multiple parameters are chosen, and a model is created with every combination possible to be compared with each other to find the best possible match. Parameters used in the grid search were the LSTM units, Dense units, and the sequence length [3].
In the reference implementation of this thesis, we followed the same implementation architecture as El Idriss et al. did to construct a benchmark that will be used later as a reference. The reference model has one *input layer*, one *LSTM layer* and two *Dense layers*. Based on results from the grid search, ten units were chosen for the LSTM layer. Thirty units were used for the first Dense layer. The last Dense layer has one unit, which is used to obtain the desired output format. The output of this model is a single blood glucose value. Figure 3.3 visualizes the benchmark model architecture.



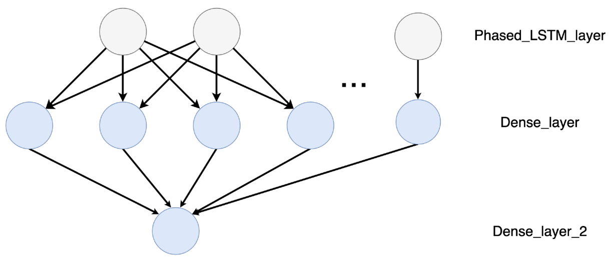


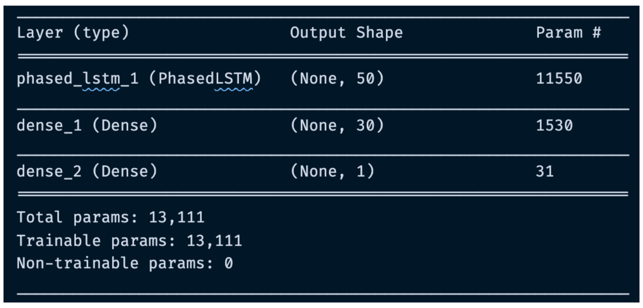
```
Layer (type)                 Output Shape              Param #
================================================================
lstm_1 (LSTM)                (None, 50)                11400
________________________________________________________________
dense_1 (Dense)              (None, 30)                1530
________________________________________________________________
dense_2 (Dense)              (None, 1)                 31
================================================================
Total params: 12,961
Trainable params: 12,961
Non-trainable params: 0
________________________________________________________________
```

Figure 3.3: Benchmark model architecture

3.4 Phased Long Short-Term Memory

The next step was to swap out the LSTM layer with a phased LSTM layer, referred to as the phased model with the benchmark model created and trained. Research has shown that phased LSTM layers will accelerate the training of RNN models [5]. Furthermore, the phased LSTM layer will work well with asynchronous data, meaning that the input data does not have to be processed. A more advanced model will be used when a base has been implemented to counter the sparse data found in our dataset, see figure 2.8.
The phased LSTM model was used, following the same architecture as the base LSTM model: one *phased LSTM layer* and two *dense layers*, as described in figure 3.4. The model should converge at the same rate or faster than the benchmark model with the help of the phased LSTM cell. Another advantage the phased LSTM layer will provide is less preprocessing. With the help of the *phased* cell, the model handles missing data better.



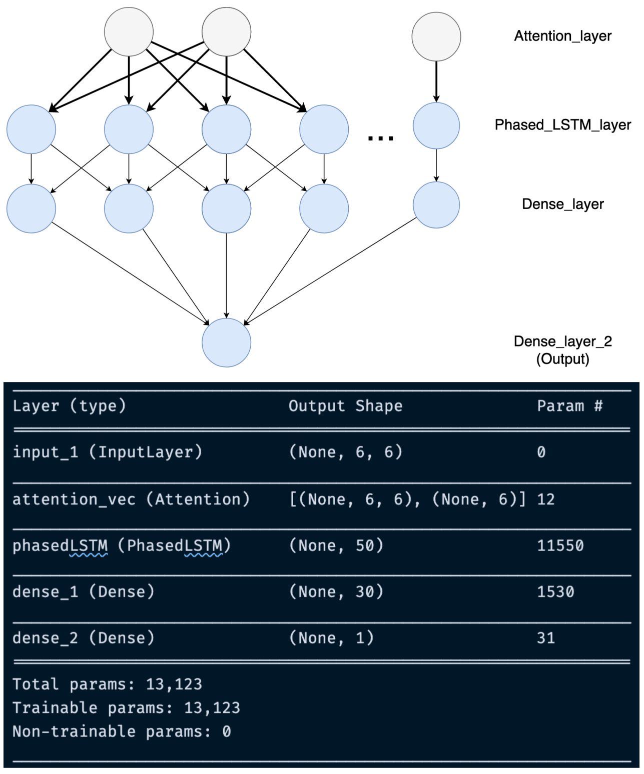


```
Layer (type)                 Output Shape              Param #
==============================================================
phased_lstm_1 (PhasedLSTM)   (None, 50)                11550
______________________________________________________________
dense_1 (Dense)              (None, 30)                1530
______________________________________________________________
dense_2 (Dense)              (None, 1)                 31
==============================================================
Total params: 13,111
Trainable params: 13,111
Non-trainable params: 0
______________________________________________________________
```

Figure 3.4: Phased LSTM model architecture

3.5 Attention-based Phased LSTM

The attention layer will provide us with beneficial information that can be used for explainable AI. Our phased LSTM model with one phased long short-term memory layer and two fully connected layers is extended with an attention layer.
For this paper, the Bahdanau-based attention layer is chosen. This layer created a vector of values, where a value corresponds to one feature. A value is the input value times the layer weight plus a bias, and this value is scaled with the hyperbolic tangent function. The attention is then calculated with the help of softmax on the vector.
The attention layer is placed before the phased LSTM layer, and the attention output is passed to the phased LSTM layer. The model is written with the help of the functional API from Keras, so the output of the middle layer can be extracted and used later. Figure 3.5 visualizes the model architecture.



```
Layer (type)                  Output Shape                Param #
==================================================================
input_1 (InputLayer)          (None, 6, 6)                0
------------------------------------------------------------------
attention_vec (Attention)     [(None, 6, 6), (None, 6)]   12
------------------------------------------------------------------
phasedLSTM (PhasedLSTM)       (None, 50)                  11550
------------------------------------------------------------------
dense_1 (Dense)               (None, 30)                  1530
------------------------------------------------------------------
dense_2 (Dense)               (None, 1)                   31
==================================================================
Total params: 13,123
Trainable params: 13,123
Non-trainable params: 0
------------------------------------------------------------------
```

Figure 3.5: Attention based phased LSTM architecture

3.5.1 Attention layer output

The model required us to remodel our implementation to use the functional API instead of the sequential API used previously. The functional API provides us with the ability to specify the output of each layer. In this case, we are interested in the output vector of our attention layer. The attention layer outputs the input times an attention vector and the attention vector itself. We are interested in the attention vector. Therefore, this is also outputted. We can then specify that the model should output this attention vector to process the information later. This information is interpreted into multiple graphs to watch the attention shift from a random initialization towards a generalized attention vector describing which feature our model is interested in when predicting the blood glucose level for a patient.

3.6 Training the model and updating weights

The dataset was provided with training and testing data. The model is trained on the training data and evaluated with the help of the testing data. During the training phased, the training data is split into 80% training data and 20% validation data. The test data is only used after the training phased, during the evaluation phase. The evaluation is done by calculating a loss value. The model is updated with the help of the loss function. The predicted value is backpropagated through the model, and the layer weights are updated based on the loss function. The loss function used for the benchmark model is the root mean squared error, equation 3.1.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i)^2} \tag{3.1}$$

For the other models, the Huber loss function, equation 3.2, is selected.

$$Huber = \delta^2(\sqrt{1+(a/\delta)^2}-1). \tag{3.2}$$

The Huber loss function is less sensitive to outliers in our dataset than the RMSE function. The RMSE loss function will result in exploding/vanishing gradients when the phased LSTM model is learning, observed while training the model. The Huber loss function will ignore more of the outliers, resulting in fewer exploding/vanishing gradients.

Chapter 4

Results

This section will present the results from gathered information during our models' training and evaluation phase. Three different model architectures were created: one standard LSTM model as a benchmark, one phased LSTM model and one phased LSTM model with an attention layer. All these models were created to forecast blood glucose levels. The models were trained on the OhioT1DM dataset provided by the University of Ohio. Each of the models was trained for a maximum of 100 epochs. The models were equipped with a callback method for early stopping. Early stopping monitors the loss function value, and the callback method will halt the training when the gradient of the loss function approaches zero. By stopping the training early overfitting the data can be prevented.

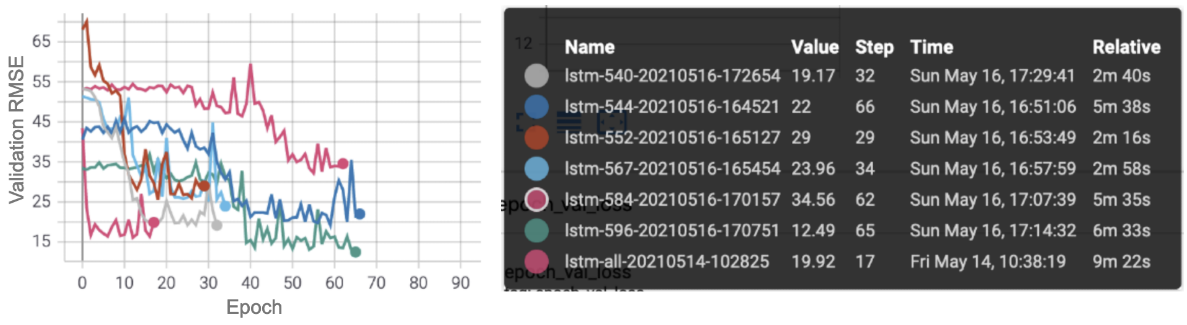
4.1 Benchmark model

The performance of the benchmark model is as close as we think it can be to the previous research results of 12.38 mg/dl [3] and 18.23 mg/dl [4]. The model performance almost matches the official results recorded for the BGLP challenge [11], which is enough to proceed with the following model architecture. Multiple LSTM models were trained on data from each patient, and Table 4.1 displays the RMSE value of the predicted values from the training data, validation data, and testing data of the models. The best performance was recorded for the patient with id 552, with an RMSE value of 17.04 on the testing data. The *combined* model referred to in Table 4.1 is a model trained on a dataset of all the combined patients' data. The combined model performed similarly to the individual models with an RMSE value of 20.63 on the testing data.
Figure 4.1 is a visual overview of how the root mean square error value decays towards the optimum, a zero value, on each elapsed epoch during the training phase. As discussed

Table 4.1: Benchmark model performance

Patient id	RMSE on training data	RMSE on validation data	RMSE on testing data
540	19.07	20.70	23.03
544	25.35	24.41	23.15
552	20.24	30.10	17.04
567	22.12	27.28	26.06
584	32.04	40.35	30.47
596	19.91	18.51	21.14
combined	22.16	20.84	20.63

earlier, the LSTM is a fast-converging RNN model, which can also be interpreted from the steep gradient in Figure 4.1, where the X-axis corresponds to the epochs and the Y-axis the RMSE value from the validation data. In Figure 4.1 a table of reference is found to the right. Every model has an identifying name, the validation root mean square error value (Value), how many epochs the model was trained for (Step), when it was trained (Time), and the duration of the training (Relative). When comparing the individual trained models with the model trained on a combined dataset in the list (lstm-all-20210514-102825), a similar RMSE value can be observed. The only thing that this model stands out with is the training time. At the same time, the other models were trained for two to six minutes. The combined model was trained for nine and a half minute. This is due to the size of the dataset used while training.



	Name	Value	Step	Time	Relative
●	lstm-540-20210516-172654	19.17	32	Sun May 16, 17:29:41	2m 40s
●	lstm-544-20210516-164521	22	66	Sun May 16, 16:51:06	5m 38s
●	lstm-552-20210516-165127	29	29	Sun May 16, 16:53:49	2m 16s
●	lstm-567-20210516-165454	23.96	34	Sun May 16, 16:57:59	2m 58s
●	lstm-584-20210516-170157	34.56	62	Sun May 16, 17:07:39	5m 35s
●	lstm-596-20210516-170751	12.49	65	Sun May 16, 17:14:32	6m 33s
●	lstm-all-20210514-102825	19.92	17	Fri May 14, 10:38:19	9m 22s

Figure 4.1: Benchmark model training callback data.
Left: a graph over the elapsed training, where the X-axis is the epoch and Y-axis the validation root mean square error value.
Right: A table with the individualizing name of the model, the final RMSE value, number of epochs before the early stopping callback was called, when the model was trained and the elapsed training time for the individual model.

To further evaluate the model and ensure that it is making sensible predictions, a graph was produced where the predicted values are compared to the actual values of the dataset with the models trained on the individual patient's data. Figure 4.2 represents a comparison of the two values, where the red line is the predicted values and the blue line the actual blood glucose level from the dataset.

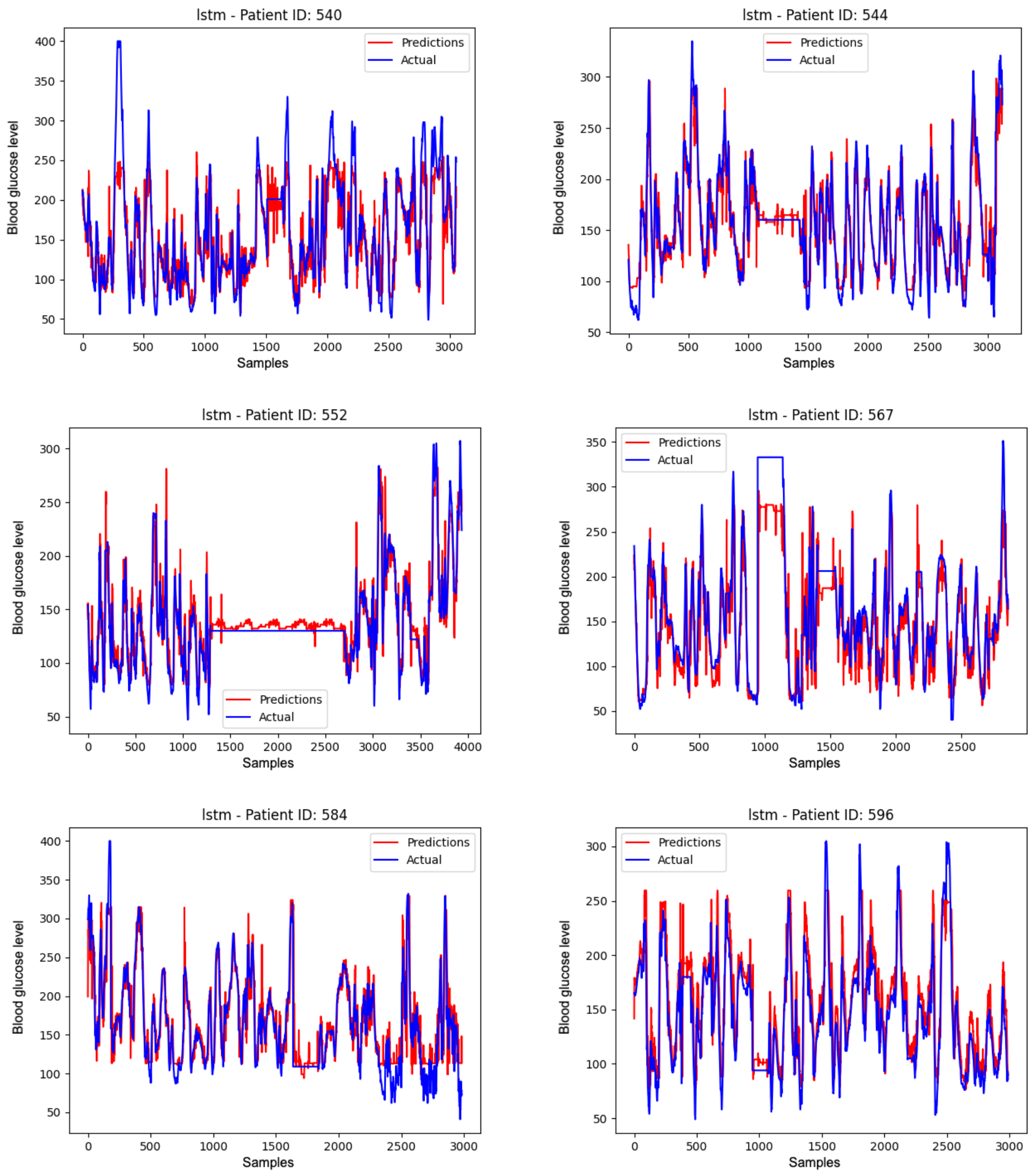


Figure 4.2: Benchmark (LSTM) - Comparison of the predicted value and the actual BGL level for the individual trained models.

4.2 Phased LSTM model

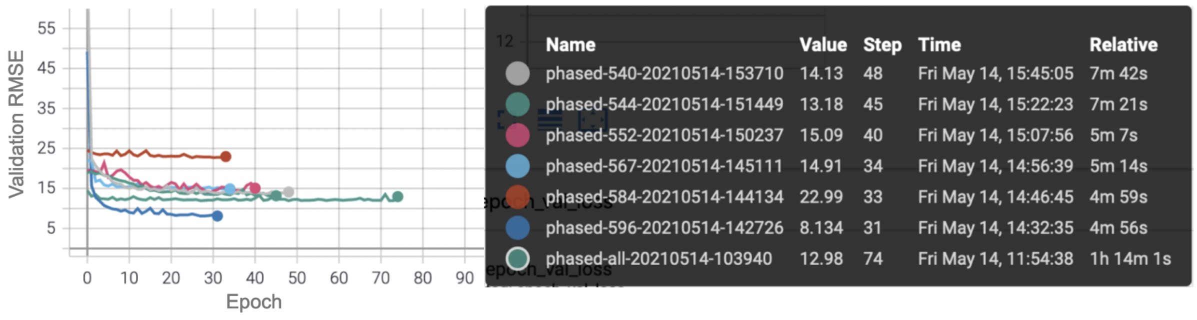
Likewise, multiple phased LSTM models were created, one for each patient and one for the combined dataset. The phased LSTM models produced a better RMSE value for each of the models, as seen in Table 4.2, with a 41% increased performance. Like with the benchmark model, the combined model performed similarly to the individual trained model.

Table 4.2: Phased LSTM model performance comparison.

Patient id	RMSE on training data	RMSE on validation data	RMSE on testing data
540	19.17	18.04	20.89
544	15.30	17.79	14.67
552	16.17	18.65	10.12
567	18.21	19.83	18.69
584	22.00	30.60	20.23
596	14.61	11.33	15.30
combined	18.81	16.79	16.63

Furthermore, the phased LSTM model converges faster compared to the benchmark model. This can be observed in Figure 4.3, where each model has a very steep decreasing gradient during the first five epochs and drastically diminished after epoch 5. The phased LSTM model required less training time to achieve similar performance as an LSTM model. The phased LSTM model also converges better on sparse data, as discussed in section 2.2.2. An observation can be made on the duration of training for the combined model, where the model was trained for 1 hour and 14 minutes before the early stopping callback function stopped it. This averages out to a training time of 60 seconds per epoch, which is almost six times longer than when the benchmark model was trained on the combined dataset.

A graph was produced for the phased LSTM model where the actual value is compared to the predicted value to ensure that the phased LSTM model produces sensible predictions, Figure 4.4. An observation is that the phased LSTM models produce fewer spikes in the predicted values and the predictions are closer to the actual value than the benchmark models predictions seen in Figure 4.2.



	Name	Value	Step	Time	Relative
●	phased-540-20210514-153710	14.13	48	Fri May 14, 15:45:05	7m 42s
●	phased-544-20210514-151449	13.18	45	Fri May 14, 15:22:23	7m 21s
●	phased-552-20210514-150237	15.09	40	Fri May 14, 15:07:56	5m 7s
●	phased-567-20210514-145111	14.91	34	Fri May 14, 14:56:39	5m 14s
●	phased-584-20210514-144134	22.99	33	Fri May 14, 14:46:45	4m 59s
●	phased-596-20210514-142726	8.134	31	Fri May 14, 14:32:35	4m 56s
●	phased-all-20210514-103940	12.98	74	Fri May 14, 11:54:38	1h 14m 1s

Figure 4.3: Phased LSTM model training callback data.
Left: a graph over the elapsed training, where the X-axis is the epoch and Y-axis the validation root mean square error value.
Right: A table with the individualizing name of the model, the final RMSE value, number of epochs before the early stopping callback was called, when the model was trained and the elapsed training time for the individual model.

4.3 Attention Phased LSTM model

The third and final model architecture is a phased LSTM model with an attention mechanism. Like previously, multiple models were trained, one from each patient's data plus a model from the combined data from all the patients. The attention vector was extracted from the attention layer, which is the first layer of our model, to achieve the explainability of our models' behaviour. The performance of these models is comparable with the phased LSTM models, with a slightly better RMSE value overall. The models outperformed the benchmark models and the phased LSTM models. Table 4.3 is a summary of the model's RMSE values from the different parts of the dataset, with an average of 16% increased performance compared to the phased LSTM model.

Table 4.3: Attention PLSTM model performance comparison

Patient id	RMSE on training data	RMSE on validation data	RMSE on testing data
540	15.99	15.03	17.94
544	13.10	14.71	12.94
552	14.99	19.23	9.57
567	15.26	16.59	16.31
584	18.96	25.00	17.77
596	11.85	9.38	12.38
combined	14.88	13.37	13.38

While the attention layer adds complexity to the model by introducing an extra layer, it does not increase the training time compared to the previously mentioned phased LSTM

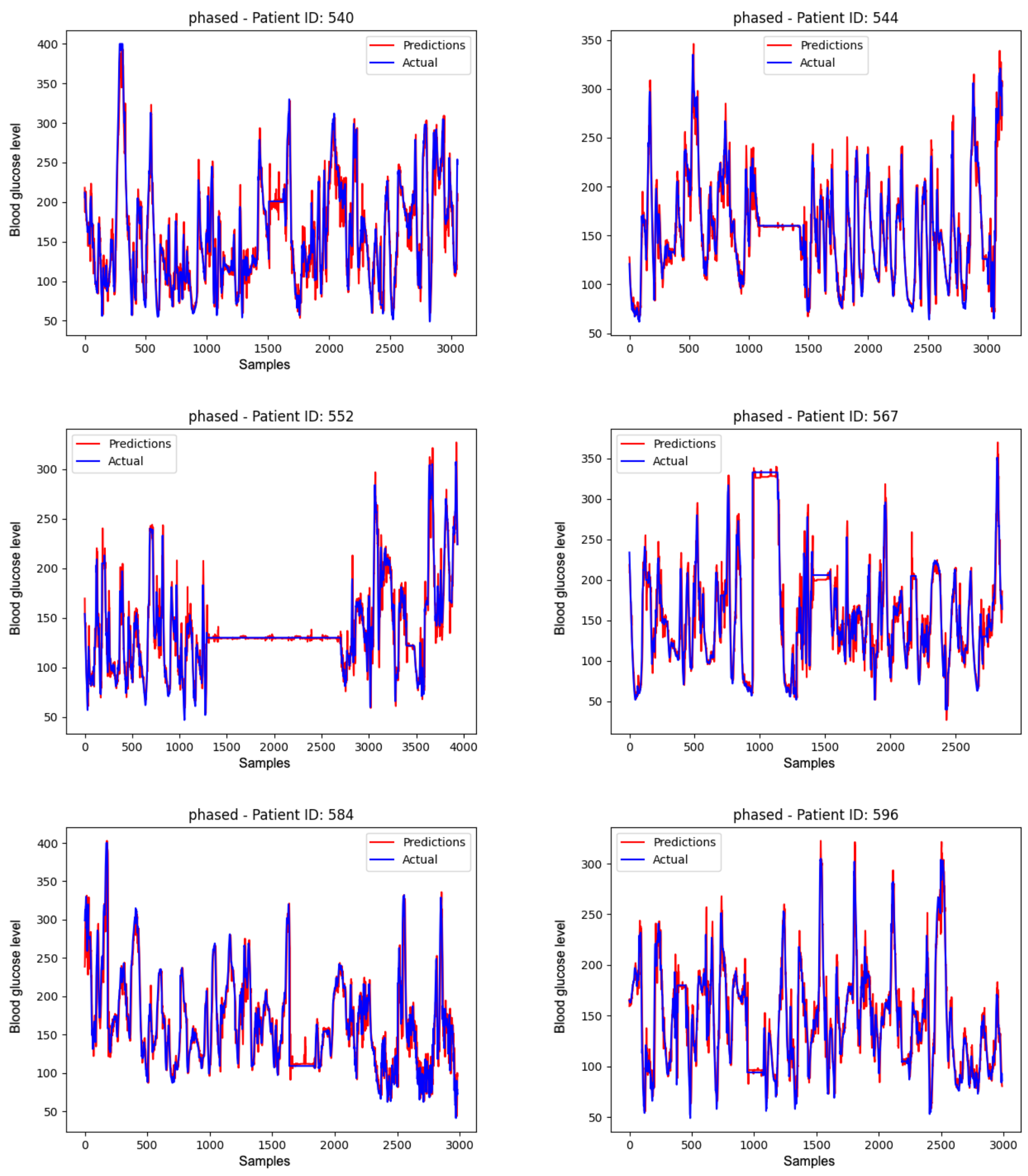
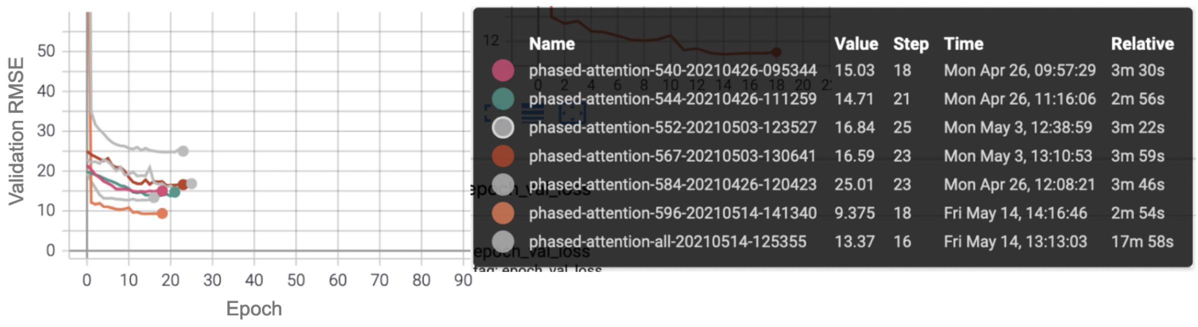


Figure 4.4: Phased LSTM - Comparison of the predicted value and the actual BGL level for the individual trained models.

model architecture. A similar steep gradient can be observed when visualizing the RMSE value over time from the training phase in Figure 4.5. Compared to the phased LSTM model without an attention mechanism, this model performed slightly better with an average RMSE value of 14.98.



	Name	Value	Step	Time	Relative
●	phased-attention-540-20210426-095344	15.03	18	Mon Apr 26, 09:57:29	3m 30s
●	phased-attention-544-20210426-111259	14.71	21	Mon Apr 26, 11:16:06	2m 56s
●	phased-attention-552-20210503-123527	16.84	25	Mon May 3, 12:38:59	3m 22s
●	phased-attention-567-20210503-130641	16.59	23	Mon May 3, 13:10:53	3m 59s
●	phased-attention-584-20210426-120423	25.01	23	Mon Apr 26, 12:08:21	3m 46s
●	phased-attention-596-20210514-141340	9.375	18	Fri May 14, 14:16:46	2m 54s
●	phased-attention-all-20210514-125355	13.37	16	Fri May 14, 13:13:03	17m 58s

Figure 4.5: Attention phased LSTM model training callback data.
Left: a graph over the elapsed training, where the X-axis is the epoch and Y-axis the validation root mean square error value.
Right: a table with the individualizing name of the model, the final RMSE value, number of epochs before the early stopping callback was called, when the model was trained and the elapsed training time for the individual model.

A similar result can be observer in Figure 4.6 as in Figure 4.4 where the predicted value is compared against the actual blood glucose level from the dataset. When comparing the spikes produces by this model, they are more noticeable than the other phased LSTM model, without an attention mechanism. It is hard to explain why the spikes appear with the attention layer included, even when analysing the attention vector.

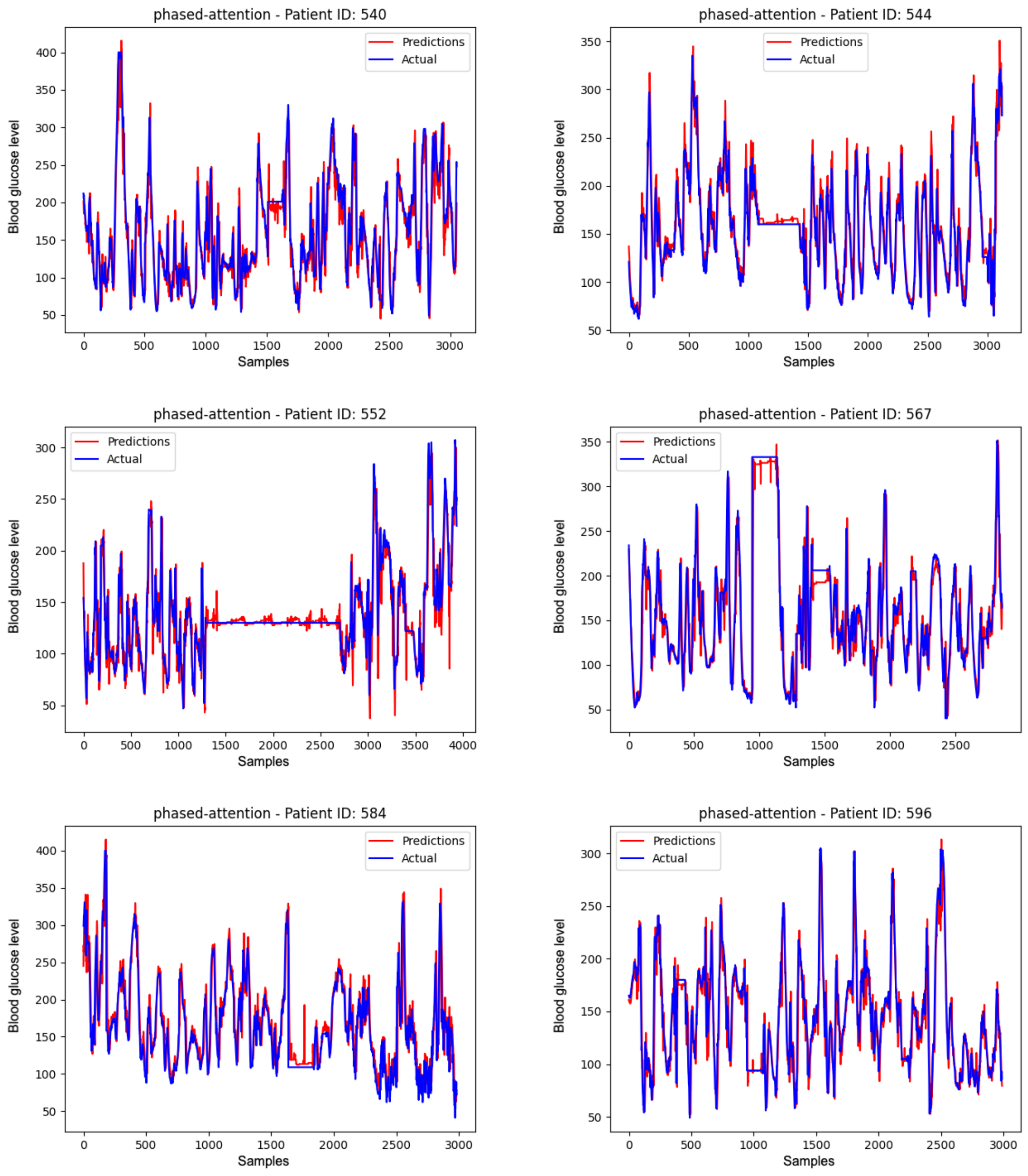


Figure 4.6: Attention PLSTM - Comparison of the predicted value and the actual BGL level for the individual trained models.

The attention vector is extracted from the attention layer and is visualized in Figure 4.7, where each bar corresponds to the feature in the dataset. A higher value means that the feature has more weight when predicting the next value. The weight is how much the model considers the feature when making a prediction. The blood glucose level has the highest value, with a weight of 0.35. The lowest weight is from the bolus feature, with a value of 0.12. This attention vector shows the average attention values for the individually trained models. Analysing the model trained on a combined dataset, *phased-attention-all-*

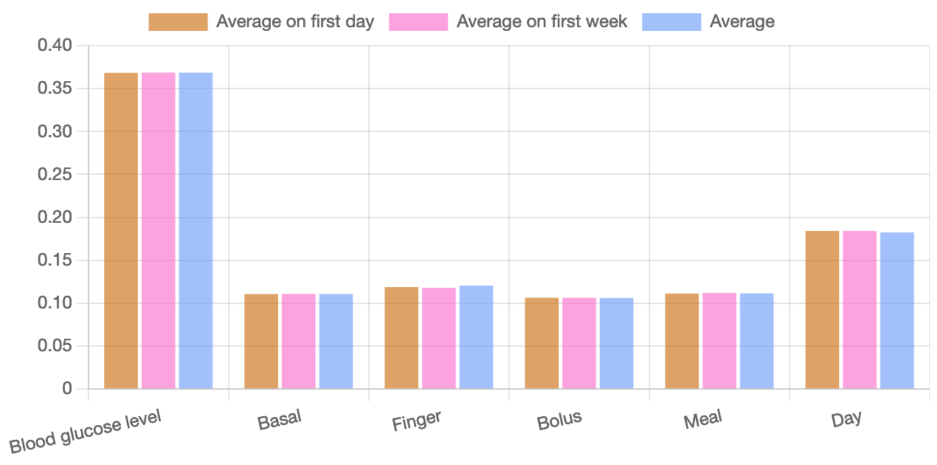


Figure 4.7: Attention vector for a model trained on a single patient's data, average from the first day, the first week and over the whole dataset (8 weeks).

20210514-125355, which had a similar training time (average 67 seconds per epoch) as the combined model without the attention mechanism. When further inspecting the attention vector, it did not resemble the attention vector from the models trained individually. The feature which had the highest attention value from the other models had the lowest, and the other feature values were almost twice as high as the other models. Figure 4.8 shows the average attention value from the combined dataset.

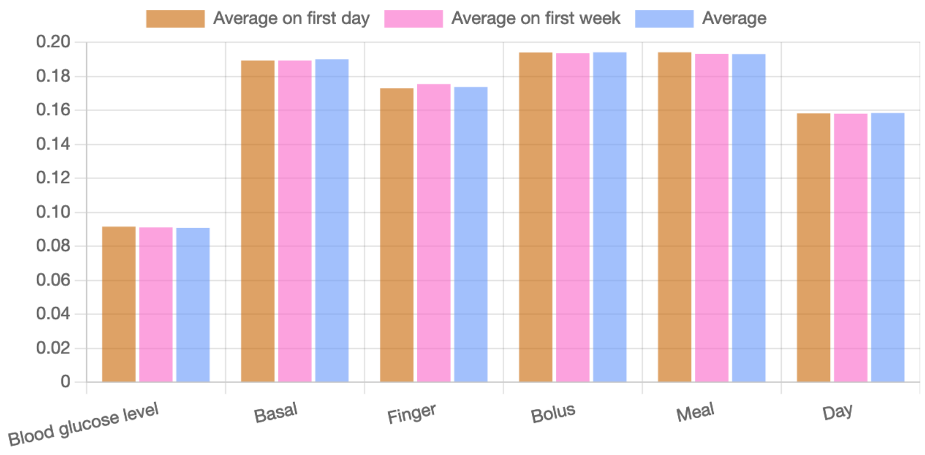


Figure 4.8: Attention vector average from the first day, the first week and over the whole dataset (8 weeks).

4.4 Interpretation

A comparison is made on the different models on each patient's test data to understand the model's performance better. Training a model on one patient's data should not affect the model's ability to predict blood glucose levels from any other source. The general interpretation of a machine learning model is, the more training data there is, the better the model will perform. Table 4.4 shows all the different models performance on the patients testing data with the benchmark model architecture.
With the results from Table 4.4 a general assumption can be made that training a model for a specific case is not necessary. This result matches what R. Bevan and F. Coenen stated in their study [4]. The combined model (the model made from a dataset combined from all of the different patients in OhioT1DM) performed, in general, better than the other models. A model trained for a specific model did not perform better than the combined model. Training the combined model took about six times longer (an average of 60 seconds per epoch) than a model trained on only one patient (an average of 10 seconds per epoch). There was no added benefit from training on a combined dataset with this model architecture. The best performing model on a specific patient's test data is highlighted in Table 4.4.

Table 4.4: Benchmark model performance comparison, where the value is the calculated RMSE value from the model, given the patient's test data. e.g. Model 540 was trained on patient 540's training data, and the RMSE value of 23.52 was achieved from calculating the error when predicting patient 540's test data.

Test data	Model 540	Model 544	Model 552	Model 567	Model 584	Model 596	Model combined
Patient 540	**23.52**	32.71	30.93	26.52	42.77	28.71	23.97
Patient 544	21.37	23.15	20.53	**17.67**	24.47	23.04	18.50
Patient 552	**13.18**	19.52	17.04	13.28	23.50	16.27	15.51
Patient 567	29.46	30.18	30.30	25.56	35.42	31.04	**24.04**
Patient 584	30.31	27.06	27.04	26.73	27.83	29.28	**25.22**
Patient 596	17.28	26.43	23.70	19.76	34.22	**16.48**	18.28

A similar table is produced from the phased LSTM models, where the models are tested on all the individual patient's test data, see Table 4.5. Like with the benchmark model, the phased LSTM performance was neutral across all the patients testing data, independent of what training data was used. The training time of the combined model with the phased LSTM layer is more than ten times slower than the models trained on the individual level, which was discussed in the previous section.

Table 4.5: Phased LSTM model performance comparison, where the models produced RMSE value of the given patient's training data. e.g. Model 540 was trained on patient 540's training data, and the RMSE value of 20.89 was achieved from calculating the error when predicting patient 540's test data.

Test data	Model 540	Model 544	Model 552	Model 567	Model 584	Model 596	Model com-bined
Patient 540	20.89	23.39	21.08	22.17	22.19	22.88	**20.72**
Patient 544	15.78	14.67	**14.56**	14.76	15.08	15.54	15.25
Patient 552	10.75	12.49	**10.12**	10.34	11.62	11.28	10.94
Patient 567	19.92	21.67	20.07	**18.69**	20.33	20.28	19.17
Patient 584	20.84	20.78	20.50	20.80	20.23	**20.05**	20.44
Patient 596	16.32	16.69	**14.96**	15.26	15.68	15.30	15.00

A table was produced to compare the models produced from the phased LSTM model architecture with an attention mechanism, see Table 4.6. The best performing model for a given patient is highlighted. An interesting observation can be made that the model trained on the combined data from all the patients performs better than any other model. This result is only seen with the attention layer model architecture, and this result might indicate that the attention layer will improve the model when there is enough training data to train the models on.

Table 4.6: Attention Phased LSTM model performance comparison, where the value is the models produced RMSE value of the given patient's training data.

Test data	Model 540	Model 544	Model 552	Model 567	Model 584	Model 596	Model combined
Patient 540	17.95	18.93	20.14	18.71	22.63	18.77	**16.98**
Patient 544	12.84	12.94	13.53	12.59	15.43	12.77	**12.49**
Patient 552	9.28	10.27	10.40	9.19	12.56	9.37	**8.89**
Patient 567	17.77	17.35	19.81	16.31	19.29	17.42	**15.34**
Patient 584	16.73	15.95	17.77	16.71	17.77	16.25	**15.84**
Patient 596	12.20	13.32	13.92	13.14	15.28	12.38	**12.15**

What makes the attention model perform best when trained on a combined dataset? One answer to this question is the number of training samples, which gives the attention layer enough samples to adjust the weight, resulting in a lower RMSE value from the predictions without an attention mechanism.
Another answer can be found when interpreting Figure 4.8, where the average attention value is calculated from the vector during the first day, the first week and the whole dataset. This figure shows that the model pays more attention to the other features compared to when it is only trained on a single patient's data, seen in Figure 4.7. Thanks to the attention layer, a more balanced weight distribution is achieved over all the features in our dataset. Furthermore, the blood glucose level is not as important as the other features when using an attention layer and training with enough data samples.

Chapter 5

Discussion

Many models were created from three different model architectures: from each architecture, one model for each patient and one from a combined dataset was created. The phased LSTM model converges faster than a standard LSTM model, and fewer samples are required for the phased LSTM model to converge. The attention layer adds explainability to the research topic. While the attention layer adds complexity to the model, the training time stays the same. A virtual feature, "time-of-day", was created. Blood glucose level had the highest attention value when the model was trained on a small dataset. When the model was trained on a larger dataset, the BGL feature had the lowest attention value.

As previous research has stated, a model trained on combined data from multiple patients performs better than a trained model for individual patients. A model trained on one patient performs equally good for data from another patient; this can be seen in the previous section. Strengthening previous research results where individually trained models, specific for a patient, were compared to one model where all patients data were used. The previous research stated that training individual models is unnecessary and that a general model will perform equally well; this is true [4]. While the model trained on a combined dataset performs equally well, the average training time for an individual model is far less than when trained on a combined dataset. A preprocessed dataset is superior when predicting blood glucose levels with a standard LSTM model. While our benchmark model achieved an average RMSE value of 24.57 it did not achieve the same level as previous studies [3].

While the benchmark model did not achieve the same level of accuracy as the same model with a more preprocessed dataset, as seen in previous research [3], the phased LSTM model did achieve similar results. The phased LSTM model achieved an RMSE value of 17.37, which is an 41% improvement compared to the benchmark model trained in this study. The training time for the phased LSTM model is twice as long as the benchmark model, averaging at around 6 minutes per model. The model's training time should not be used as a parameter but should guide when considering architecture and the number of samples. Furthermore, an early stopping callback function was used when trained in these models, directly affecting how long the model training takes. To make the model's training time comparable with each other, an average time per epoch has been calculated. When calculating an average training time per epoch, the benchmark model had an average training time of 5.4 seconds per epoch, and the phased LSTM model 9 seconds per epoch. The attention phased LSTM model had an average training time similar to the phased LSTM. The combined model was not taken into consideration when calculating the average training time. For the combined model, the training time is over six times slower than when trained on an individual level. The combined dataset contained data from six individual patients'. The training time correlates with the number of samples used in the dataset.

The combined model (trained on a combined dataset) did not achieve better performance for the benchmark model or the phased LSTM model. The combined model outperformed the other models when trained with the attention mechanism. The attention phased LSTM model achieved the best performance, with an average RMSE of 14.98, a 16% improvement from the phased LSTM model without an attention mechanism. The conclusion is that the attention mechanism needs more training data than the other layers in the model; therefore, it outperformed the other models. Furthermore, the attention layer had a balanced value across all the used features when trained on enough data, which results in the model considering all the features when making a prediction. The attention phased LSTM model trained on an individual level had a higher value for the blood glucose level feature than the rest. When trained on a larger dataset, the attention mechanism pays less attention to the blood glucose level and more to the other features in the dataset; this can be seen in Figure 4.8.

For the models trained on an individual level, independent of the architecture, patient 552 had the best result. When analysing the data, an observation can be made that over 30 per cent of the blood glucose level is missing. This can be observed as a straight line in Figure 4.2, Figure 4.4 or Figure 4.6 in Chapter 4.

To summarise, the model with an attention mechanism performed the best, and with enough training data, the attention layer is utilised the most. The preprocessing of the dataset affects the RMSE when using a standard LSTM layer. This result can be seen

from the benchmark model used compared to previous research, where the preprocessing was more comprehensive than used in this study.
The phased LSTM model and the phased LSTM model with an attention mechanism converged faster than the benchmark model. While the attention mechanism introduced more complexity to the model, it did not increase the training time. More features could be introduced to decrease the RMSE value further, and the dataset could be augmented to fill in the missing values.

Chapter 6

Conclusion

In this thesis, previous research was reviewed on how long short-term memory (LSTM) models were implemented for the blood glucose level prediction (BGLP) challenge. The University of Ohio published the BGLP challenge in 2018, along with the OhioT1DM dataset. A few other model architectures were reviewed to compare the already existing LSTM models result. Three different model architectures were selected; One benchmark model with the same architecture as one of the previous research, one phased LSTM model architecture, and one phased LSTM model architecture with an attention layer. The models were trained individually on multiple patients from the OhioT1DM dataset. A model was trained on a combined dataset from all the patients' training data for each model architecture model.
The OhioT1DM dataset was initially published in 2018 with eight weeks of continuously recorded data from six patients with type 1 diabetes. In 2020 the dataset was extended with six more patients. These patients were released to aid the research in predicting the blood glucose level. In this study, the data from the six original patients were used for training the models.

We briefly discussed the state of machine learning and deep learning. How deep learning has become the new standard for machine learning. We discussed what regression is and how regression is used in deep learning and long short term memory cells, and why it is essential for this study.

The architecture of the LSTM cell was reviewed and compared to the phased LSTM model. We went through each of the components of the LSTM cell. We compared the internal difference between the phased LSTM model and the normal LSTM cell. We discussed the Kronos gate introduced by the phased LSTM cell and how it aids in training on sparse data. The Kronos gate implements a phase to the LSTM cell, which dictates if

the LSTM cell can be updated or not.
The final model architecture, a phased LSTM model with an attention mechanism, introduces explainable AI (XAI) to the research topic. This thesis reviewed the two main attention mechanism types; The Bahdanau type and the Luong type, which builds on the Bahdanau type.
The attention mechanism introduces a neural network layer that produces an attention vector; the attention vector contains values corresponding to the input features in the data samples. A high attention value equals great attention towards that feature, and a low attention value equals low attention towards that feature. The Luong type scores the attention vector differently than the Bahdanau, but the mechanism stays the same. The attention mechanism helps us understand which feature is essential when predicting blood glucose level with the help of the features in the OhioT1DM dataset.

The benchmark model was implemented first for the study, then the phased LSTM model and last, the phased LSTM model with an attention mechanism. The benchmark model was implemented to verify the data processing. The benchmark performance almost matches the reference studied performance, so the data processing was a success.
The phased LSTM model achieved a better result than the benchmark model, which was expected due to the missing values in the dataset. The performance increased by X% compared to the benchmark model. Furthermore, the phased LSTM model converged faster thanks to the Kronos gate — The phased LSTM model is faster at learning than the standard LSTM model. The RMSE loss function produced exploding/vanishing gradient, so the Huber loss function was selected for the phased LSTM models. The Huber loss function is less sensitive to outlying values in a dataset, which results in fewer problems with exploding/vanishing gradients.
By extending the phased LSTM model with an attention mechanism, XAI introduces the research topic. A superficial Bahdanau type attention layer was implemented to extract the model's attention with the last architecture in our study. The attention layer does not affect the training time for a phased LSTM model.

The attention vector extracted from the attention layer did not affect the prediction of BGL on an individual level. The model trained on a combined dataset had an increased performance compared to the individually trained models. This result was achieved by calculating the RMSE value from predicting BGL from each patient's test data. For an individually trained model, the BGL feature was the essential feature when predicting BGL. When trained on enough samples, the BGL was the least important feature in our dataset. The other features had an equally high value from the extracted attention vector.

For further research, more features should be selected, and more parameter testing should be done. The attention vector from the combined dataset showed promising results and could be used to increase the performance of the models further.
A single model trained on a single patient's data is enough to predict the blood glucose level of an unknown patient. A combined model takes longer to train but might be more robust.

Chapter 7

Svensk sammanfattning

7.1 Inleding

Till följd av ökat intresse för maskininlärning skapade Ohio universitet år 2018 utmaningen Blood Glucose Level Prediction (BGLP). BGLP kan översättas med estimering av blodsockernivå. Med hjälp av utmaningen hoppades universitetet på att skapa ett ökat intresse för forskning kring blodsocker och diabetes.

För att användas till utmaningen lanserades OhioT1DM, som är en datamängd som innehåller åtta veckor kontinuering data från patienter med typ-1 diabetes. Datat är insamlade med fem minuters intervaller och innehåller bland annat blodsockernivån, insulinnivån, fysiska mätningar samt självrapporterade händelser, såsom matvanor, sömn och stress, med mera. År 2020 lades ytterligare sex patienter till i datamängen, med mera data per patient. Universitetet hoppades detta skulle öka intresset för forskningsämnet.

Tidigare forskning inom ämnet har med hjälp av datamängen OhioT1DM lyckats utmärkt med att estimera blocksockernivån. I en studie av Junyu Xie och Qian Wang jämfördes flera maskininlärnings metoder med en autoregressions metod med hjälp av BGLP. Forskarna fann ingen betydande skillnad på prediktion av blodsockernivå mellan maskininlärningsmetoderna och autoregressionsmetoden.

I en annan studie av El Idriss m.fl. skapades en LSTM-modell, som hör till djup maskininlärning, och som har två fullt ihopkopplade neuronnätslager samt ett LSTM-lager [3]. LSTM kommer från egelska *Long Short-Term Memory* och kan översättas med långt korttidsminne. Denna modell jämfördes med en existerande LSTM-modell samt en autoregressionsmetod. Resultaten visade att deras LSTM-modell överträffade en tidigare LSTM-modellen gjord av för alla patienter från OhioT1DM, samt 9 av 10 patienter för autoregressionsmetoden.

LSTM-modellens struktur används som utgångspunkt för detta diplomarbete. Utöver LSTM-modellen skapas en *phase LSTM*-modell, phase kommer från egenskan och betyder fas. En fas i detta sammanhang innebär en sinusvåg. Tidigare forskning visar att med hjälp av en fas i LSTM-modellen kan mycket av förbehandlingen uteslutas, eftersom att fasen hjälper modellen konvergera på gles data.

Utöver phased-LSTM lagret kommer ett uppmärksamhetslager (eng. attention layer) sättas till i modellen, vilket bidrar med information om vilka variabler som beaktas mer av modellen vid predicering av ett värde. Förhoppningen är att detta i sin tur skulle bidra med bidra med förklarbarhet till LSTM-modeller i BGLP. Genom att implementera modellen med ett uppmärksamhetslager kommer förklarlig maskininlärning, (eng. Explainable AI (XAI)) att tas in i forskningen.

7.2 Metod

Först implementerades den normala LSTM-modellen som utgångspunkt, som verifiering för att dataprosesseringen gjorts rätt. LSTM-modellen implementerades i enlighet med tidigare forskning: med ett LSTM-lager och två fullt ihopkopplade lager. I den tidigare forskningen utfördes rutnätssökning, vilket innebär att man väljer flera parametrar för modellen. Av permutationer av parametrarna skapas modeller, vilka tränas med datamängden, och modellen som har bäst prediktabilitet väljs. Med hjälp av rutnätssökningen valdes följande parametrar till modellen: 10 LSTM-enheter, 30 enheter i den första fullt ihopkopplade lagren, och en sekvens läng på 10.

Efter att den normala LSTM-modellen uppvisade liknande prediktabilitet som i tidigare forskning implementerades den nya modellen, LSTM-modellen med en fas, (eng. *phased LSTM* eller kort PLSTM. Tack vare PLSTM-modellen kan förbehandlingen av datamängden förkortas till att endast bestå av att de enskilda samplen sätts ihop till en stor datamängd, från formatet XML till Numpy data. Det senare dataformatet kan avläsas snabbare i Python. PLSTM-modellen följer samma uppbyggnad som LSTM-modellen, och består av ett PLSTM-lager och två fullt ihopkopplade lager. Samma parametrar valdes för PLSTM-modellen som för LSTM-modellen. PLSTM-modellen visade tidigt prediktabilitet som påminner om den normala LSTM-modellen. I enlighet med tidigare forskning konvergerade PLSTM-modellen därtill snabbare än LSTM-modellen.

Utöver PLSTM-modellen skapades den sista modellen, en PLSTM-modell med ett uppmärksamhetslager. Uppmärksamhetslagret placerades som första lager i modellen och användes främst för att hämta information från modellen, om vilka mätningar modellen tar mest hänsyn till vid estimering av blodsockernivån.

7.3 Resultat

För alla tre modellstrukturerna tränades flertal modeller, en modell för varje patient i datasettet samt en modell från ett kombinerat dataset från alla patienter. Liksom tidigare angetts visade den normala LSTM-modellen liknande prediktabilitet som i tidigare forskning. den normala LSTM-modellen uppvisade ett genomsnittligt RMSE värde på 24,57. Detta värde motsvarade nästan det som tidigare meddelats för samma modell struktur. På grund av att datat inte var förbehandla på samma sät kommer modellen inte att uppnå samma värden som tidigare forskning, men detta var ett tillräkligt bra värde för att implementera följande modellstruktur.

PLSTM-modellen visade ett bättre resultat än den normala LSTM-modellen och hade en förbättring på 41%. PLSTM-modellens genomsnittliga RMSE värde är $17,37$. Detta innebär att PLSTM-modellen kräver mindre förbehandling av datat innan modellerna tränas på det, vilket tidigare forskning visat.

Den sista modellstrukturen, en PLSTM-modell med en uppmärksamhetsmekanism. Den sista modellstrukturen presterade i genomsnitt 17% bättre än PLSTM-modellen utan ett uppmärksamhetslager. Från uppmärksamhetslagret fås en vektor vars värden visar hur mycket uppmärksamhet modellen visar för ett visst värde vid projicering av nya värden. För dom individuelt tränade modellerna indikerade uppmärksamhetsvektorn att blocksockernivån har en stor betydelse och dom övriga egenskaperna en mindre betydelse vid projicering av blocksockernivån. Figur 2.10 är en graf över uppmärksamhetsvektorn från den första dagen, första veckan och hela datasettet, figuren visar att blocksockernivån har störst betydelse vid projicering och dom övriga egenskaperna ett balancerat intresse.

All modeller med uppmärksamhetslagret producerade ett bättre resultat än modellerna utan upmärksamhetslagret. Modellerna testades på alla patienters testdel av datat och resultat visas i Figur 4.7. Som ses i figuren producerade den kombinerade modellen det bästa resultaten för alla patienters data. Utöver att den kombinerade modellen producerade det bästa resultatet hade modellen en helt annan distribution av uppmärksamhetsvektorn. Blocksockernivån var den minst betydelsefulla egenskapen, som tidigare var den mest betydelsefulla egenskapen för modellen vid projicering av blocksockernivån. Dom övriga egenskaperna hade ett balancerat uppmärksamhetsvärde, Figur 4.8 visar en graf över distributionen av uppmärksamhetsvektorn. PLSTM-modellen med ett uppmärksamhetslager hade ett liknande resultat som den modell från en tidigare studie som användes som referens för denna studie.

7.4 Diskussion och slutsats

Genom att jämföra tidigare skapade modellstrukturer med dom som skapades i denna studie har nya synvinklar skapats samt en modell som klarar sig bra jämfört med dom tidigare modellerna vid projicering av blocksockernivån i BGLP utmaningen. Genom att skapa en modell med hjälp av den fasande LSTM-lagret (eng. phased LSTM layer) kan processeringen av data minskas vilket leder till att datat som modellen tränas med påminner mera om det data som fås direkt av patienten. Detta leder till att den modellstruktur som skapas i denna studie kan lättare användas till ett praktiskt syfte.

Utöver en bättre presterande modell producerar modellen en uppmärksamhetsvektor som kan användas för att urskilja vilka egenskaper som modellen tar hänsyn till vid projicering av nya värden. För en modell som tränas med färre sampel, enskillda patienters data, är blocksockernivån den mest betydelsefulla egenskap. En model som tränas på flera sampel, ett kombinerat dataset av alla patienter, är blocksockernivån den minst betydelsefulla egenskapen och dom andra egenskaperna har en balancerad betydelse.

Det är inte nödvändigt att träna en modell på individuell nivå, vilket märktes redan i tidigare studier. En modell med ett uppmärksamhetslager som är tränat på ett kombinerat dataset presterar bättre än en model tränad på en individuell nivå. Studien kan fortsättas genom att utvidga datasettet med flera egenskaper, för att se ifall detta påverkar resultatet vid projicering av blodsockernivån. Genom att justera modellens parametrar kan ytterligare ett bättre resultat uppnås och träningstiden förkortas.

References

[1] C. Marling and R. C. Bunescu, “The ohiot1dm dataset for blood glucose level prediction,” in *KHD(at)IJCAI*, 2018.

[2] J. Xie and Q. Wang, “Benchmark machine learning approaches with classical time series approaches on the blood glucose level prediction challenge,” in *KHD(at)IJCAI*, 2018.

[3] T. El Idrissi, A. Idri, I. Abnane, and Z. Bakkoury, “Predicting blood glucose using an lstm neural network,” Sep. 2019, 35_41.

[4] R. Bevan and F. Coenen, “Experiments in non-personalized future blood glucose level prediction,” in *Proceedings of the 5th International Workshop on Knowledge Discovery in Healthcare Data co-located with 24th European Conference on Artificial Intelligence, KDH@ECAI 2020, Santiago de Compostela, Spain & Virtually, August 29-30, 2020*, K. Bach, R. C. Bunescu, C. Marling, and N. Wiratunga, Eds., ser. CEUR Workshop Proceedings, vol. 2675, CEUR-WS.org, 2020, pp. 100–104. [Online]. Available: `http://ceur-ws.org/Vol-2675/paper17.pdf`.

[5] D. Neil, M. Pfeiffer, and S. Liu, *Phased lstm: Accelerating recurrent network training for long or event_based sequences*, 2016. arXiv: `1610.09513 [cs.LG]`.

[6] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu, “Explainable ai: A brief survey on history, research areas, approaches and challenges,” in. Sep. 2019, pp. 563–574, ISBN: 978-3-030-32235-9.

[7] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, 2016. arXiv: `1409.0473 [cs.CL]`.

[8] M.-T. Luong, H. Pham, and C. D. Manning, *Effective approaches to attention-based neural machine translation*, 2015. arXiv: `1508.04025 [cs.CL]`.

[9] Medtronic, *Minimed® 530gsystem user guide*, English, Medtronic, May 24, 2021, 295 pp. [Online]. Available: `https://www.medtronicdiabetes.com/sites/default/files/library/download-library/user-guides/z10-mp6025813-014-000-a--mp6025813-014_a.pdf`, 2018-3-2.

[10] T. EL Idrissi, A. Idri, and Z. Bakkoury, “Systematic map and review of predictive techniques in diabetes self-management,” *International Journal of Information Management*, vol. 46, pp. 263–277, 2019, ISSN: 0268-4012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0268401218304705.

[11] *Official bglp results*, http://smarthealth.cs.ohio.edu/bglp/bglp-results.html, Accessed: 2021-05-24.

[12] S. Hochreiter and J. Schmidhuber, “Long short_term memory,” *Neural computation*, vol. 9, 1735_80, Dec. 1997.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, 2017. arXiv: 1706.03762 [cs.CL].

[14] Tensorflow, *Tensorflow addons networks: Sequence-to-sequence nmt with attention mechanism*, 2020 (accessed 2020-12-13). [Online]. Available: https://www.tensorflow.org/addons/tutorials/networks_seq2seq_nmt.

[15] D. Neil, *Public_plstm*, 2017 (accessed 2020-12-13). [Online]. Available: https://github.com/dannyneil/public_plstm.

[16] D. A. Kaji, J. R. Zech, J. S. Kim, S. K. Cho, N. S. Dangayach, A. B. Costa, and E. K. Oermann, “An attention based deep learning model of clinical events in the intensive care unit,” *PLOS ONE*, vol. 14, no. 2, pp. 1–17, Feb. 2019. [Online]. Available: https://doi.org/10.1371/journal.pone.0211057.