Early detection of positive blood cultures using recurrent neural networks on time series data

Thomas Peiffer

Supervisors: Prof. dr. ir. Filip De Turck, Prof. dr. ir. Tom Dhaene

Counsellors: Ir. Rein Houthooft, Ir. Joeri Ruyssinck, Dr. Femke Ongenae, Ir. Cedric De Boom

Master's dissertation submitted in order to obtain the academic degree of Master of Science in Computer Science Engineering

Department of Information Technology
Chair: Prof. dr. ir. Daniël De Zutter
Faculty of Engineering and Architecture
Academic year 2015-2016
Early detection of positive blood cultures using recurrent neural networks on time series data

Thomas Peiffer

Supervisors: Prof. dr. ir. Filip De Turck, Prof. dr. ir. Tom Dhaene

Counsellors: Ir. Rein Houthoof, Ir. Joeri Ruyssinck, Dr. Femke Ongenae, Ir. Cedric De Boom

Master's dissertation submitted in order to obtain the academic degree of Master of Science in Computer Science Engineering
Preface

A profound note of gratitude: This dissertation is the icing on the cake of many years of education. The chapters in this dissertation are the last ones in a metaphorical life-chapter. I want to thank everyone who has been part of this life-chapter because it was one of the most exciting and interesting ones in my life.

This dissertation is the result of hard work and dedication which would not have been possible without the help of certain people. I want to thank in particular:

- My supervisors Filip De Turck and Tom Dhaene.
- My counsellors Joeri Ruyssinck, Femke Ongenae, Cedric De Boom and Rein Houthooft for their valuable insights and advice.
- Johan Decruyenaere, Bram Gadeyne and Kirsten Colpaert for providing the dataset and answering all medical related questions.
- All my friends and colleagues for their support and help to keep the spirit high.
- My brother Matthias for the numerous days and nights of studying together.
- My parents for encouraging me and giving me the opportunity to acquire this degree.

Thomas Peiffer, may 2016
Permission for usage

The author gives permission to make this master dissertation available for consultation and to copy parts of this master dissertation for personal use. In the case of any other use, the limitations of the copyright have to be respected, in particular with regard to the obligation to state expressly the source when quoting results from this master dissertation.

Thomas Peiffer, may 2016
Early detection of positive blood cultures using recurrent neural networks on time series data
by Thomas Peiffer

Supervisors: Prof. dr. ir. Filip De Turck, Prof. dr. ir. Tom Dhaene
Counsellors: Ir. Rein Houthooft, Ir. Joeri Ruyssinck, Dr. Femke Ongenae, Ir. Cedric De Boom

Master’s dissertation submitted in order to obtain the academic degree of
Master of Science in Computer Science Engineering

Academic year 2015–2016

Department of Information Technology
Chair: Prof. dr. ir. Daniel De Zutter
Faculty of Engineering and Architecture
Ghent University

Abstract

Bacteria or fungi in the blood stream can mean a life-threatening condition for intensive care unit (ICU) patients. A model is developed to assist doctors in detecting this positive blood culture condition. The model is based on a Bidirectional Long Short-Term Memory (BiLSTM) network and uses 9 different monitored parameter sequences from over 2000 ICU admissions at the Ghent University Hospital.

Index Terms

Machine learning, recurrent neural network, time-series prediction, LSTM, blood culture.
Early detection of positive blood cultures using recurrent neural networks on time series data

Thomas Peiffer

Supervisors: Prof. Dr. Ir. Filip De Turck, Prof. Dr. Ir. Tom Dhaene, Ir. Joeri Ruyssinck, Ir. Rein Houthooft, Dr. Femke Ongenae, Ir. Cedric De Boom

Abstract—Bacteria or fungi in the blood stream can mean a life-threatening condition for intensive care unit (ICU) patients. A model is developed to assist doctors in detecting this positive blood culture condition. The model is based on a Bidirectional Long Short-Term Memory (BiLSTM) network and uses 9 different monitored parameter sequences from over 2000 ICU admissions at the Ghent University Hospital.

Index Terms—Machine learning, recurrent neural network, time-series prediction, LSTM, blood culture.

1 INTRODUCTION

A positive blood culture is defined as a blood sample where bacteria or fungi are present. This growth of organisms in the blood stream can lead to inflammation throughout the body or even organ failure or death [1]. When doctors suspect a patient to have a positive blood culture they can decide to advance to a blood culture test. However, symptoms are not always clear and causes of positive cultures are very complex and not yet fully understood. There are nevertheless suspicions that a link exists between a patient’s physiological data and the outcome of such a test. This paper presents research about how computational models can assist in detecting culture positive patients or even suggest new relationships.

2 RELATED WORK

Several papers suggest learning models to detect the presence of sepsis. Sepsis is a systemic response of the human body to bacteria entering the blood stream. This condition is highly correlated with a positive blood culture. Prediction models in the domain of sepsis might be exportable to positive blood culture detection. E. Henry et al. developed a targeted real-time early warning score (TREWScore) to recognize septic patients [2]. Ho et al. presents a similar system [3] which uses the Multi parameter Intelligent Monitoring in Intensive Care II (MIMIC II) [4] database.

None of the presented systems make use of models that are able to capture temporal information. However, monitored physiological patient data consists of different time series. Using models designed to work with this time series information might improve detection systems. The great advantage of using temporal models is that they can work with sequences of different lengths and they are efficient in learning time dependencies, this in contrast to non-temporal models. In the past, temporal models have meant great added value to multiple applications such as EEG classification [5] or IMDB sentiment analysis [6].

3 DATA

The ICU of the Gent University Hospital provided a database with physiological information from 2177 patients whereof 229 admissions have had a positive blood culture test. The dataset contains more than 14 million values belonging to 29 parameters. Those variables can be subdivided into 10 parameter groups. Table 1 shows these parameter groups and shows which ones are used in the blood culture classification model. Blood lactate is the only parameter not taken up in the model due to lack of enough admissions where values for this parameter are available.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Present</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blood Pressure</td>
<td>Yes</td>
</tr>
<tr>
<td>Temperature</td>
<td>Yes</td>
</tr>
<tr>
<td>Respiratory Rate</td>
<td>Yes</td>
</tr>
<tr>
<td>Lactate</td>
<td>No</td>
</tr>
<tr>
<td>CRP-s</td>
<td>Yes</td>
</tr>
<tr>
<td>Heart Rate</td>
<td>Yes</td>
</tr>
<tr>
<td>Thrombocytes</td>
<td>Yes</td>
</tr>
<tr>
<td>WBC</td>
<td>Yes</td>
</tr>
<tr>
<td>Blood Pressure</td>
<td>Yes</td>
</tr>
<tr>
<td>SOFA</td>
<td>Yes</td>
</tr>
<tr>
<td>INR</td>
<td>Yes</td>
</tr>
</tbody>
</table>

3.1 Preprocessing

The database resides samples that are not properly structured in time. Training a model on this raw data is not possible. The data needs some sort of sampling first to provide data to the system which has equal time between data points. Outliers (values that are biologically
impossible) need to be removed from the database.

Sampling happens by taking the last, mean, minimum or maximum parameter value in a time window. The length of those time windows is defined by the sampling rate (usually one sample per hour) and the sampled period (24 hours). The impact of whether a last, mean, minimum or maximum value is chosen in a sample frame is discussed in the evaluation section. A sampling period ends at the point when a first positive blood culture test is taken. In case of a negative culture patient, where there was no positive test, the sampling period ends at the time when the physiological data flow stops. After sampling, each admission is represented by a label and 9 sequences of 24 samples (depending on the sampling rate & period). This data can serve as input to the network that will be examined in the next section.

4 METHODS

A Recurrent Neural Network (RNN) is a computational model designed to work with temporal features. It’s similar to a normal feed forward neural network by extension that cycles are present in the network. Through those cycles the model obtains a memory effect that lets the network learn on inputs from several time steps in the past. To visualize the data flow through the network, the network can be unfolded over time. Figure 1 shows such an unroll in time for an RNN with one hidden node.

A cost function is defined to quantize how well the network classifies. This is usually a mean squared error (MSE) or logloss function. Training such a network boils down to optimizing all parameters so the cost function reaches its minimum value. For recurrent neural networks the process of learning is called back propagation trough time. For a detailed explanation on how this works, please refer to more specific literature [8].

4.1 LSTM

A commonly recognized problem in recurrent neural networks is the vanishing gradient problem. The influence of inputs from several time steps away fades exponentially. This makes it impossible for those networks to learn long-term dependencies. A solution for this was proposed by Sepp Hochreiter [9] and is called Long Short-Term Memory (LSTM). An LSTM cell is based on the principle of gating. A gate makes it possible for the network to block inputs or outputs and in that way contain the hidden state for longer time periods. How such an LSTM-cell looks like can be seen in figure 2.

Fig. 2. Illustration of an LSTM memory cell

Five types of networks are used for the blood culture prediction task: A normal recurrent neural network, a bidirectional RNN, a normal LSTM network, a bidirectional LSTM (BiLSTM) network and an advanced BiLSTM network with an extra hidden layer. In the advanced BiLSTM network, hidden states from all time steps have direct influence to the output node. While in the more simple BiLSTM network, the hidden state values first have to ripple through the whole LSTM chain. Figure 3 shows an example of an unfolded LSTM network with 2 hidden nodes. $x_t[k]$ represents the $k$'th input feature at time step $t$. Figure 4 shows an advanced BiLSTM network.
This section handles the prediction evaluation of the mentioned networks. Models are compared based on their ability to predict up to the point where an actual blood sample is taken and on their early prediction capabilities. Finally, the temporal models are compared to a non-temporal one.

Table 2 shows the area under the Receiver Operator Characteristic and Precision Recall curve (ROC auc & PR auc) and scores for the different models when predicting up to the point of a positive blood sample or until the last available value in case of a negative patient. The data trained on was a sampled dataset where the last value in a sample window is used.

<table>
<thead>
<tr>
<th>Network</th>
<th>ROC auc</th>
<th>PR auc</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>0.797</td>
<td>0.492</td>
</tr>
<tr>
<td>BiRNN</td>
<td>0.818</td>
<td>0.529</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.837</td>
<td>0.453</td>
</tr>
<tr>
<td>BiLSTM</td>
<td>0.841</td>
<td>0.499</td>
</tr>
<tr>
<td>BiLSTM hl</td>
<td>0.891</td>
<td>0.629</td>
</tr>
</tbody>
</table>

The advanced BiLSTM network (BiLSTM hl) achieved the best performance in ROC auc. This table shows the results when the networks are trained on the last samples in a window. However, predictions are slightly better when the maximal or minimal values for a parameter in a sample window is used. The advanced BiLSTM network with those samples achieved an ROC auc of 0.901 and a PR auc of 0.669. The ROC curve is shown in figure 5. Whether the minimum or maximum value is used was decided in dialogue with physicians. Maximum values are taken for INR, temperature, heart rate, CRP-s, blood leukocyte count and SOFA. Minimum for Blood thrombo- and leukocyte counts. The remaining parameters use the average value over the past time window.

5.1 Early prediction

The sooner a culture positive patient can be diagnosed how higher his chances of survival are. Therefore the model was tested on sequences that end several hours before the actual blood test was performed. The network is first trained on sequences that span 24 hours. Next, the network is tested on sequences that stopped earlier (x hours before the test). Figure 6 shows the scores for early prediction of the advanced BiLSTM model with extremum sample sequences. Upto 10 hours upfront the model still reaches a ROC auc of ±0.8. At 24 hours upfront the ROC auc is around 0.76. The area under the precision recall curve shows a similar evolution with lower values.

5.2 Non-temporal models

Since literature only mentions the use of non-temporal models it is interesting to see if using temporal information actually brings an added value in predicting a blood culture test outcome. Figure 6 shows the early prediction curve of the advanced BiLSTM network versus the one for a feed forward neural network with 100 hidden nodes. The input data for the feed forward neural network is the extremum dataset except that only the most recent sample is taken instead of the whole sequence. The mean value from all samples upto that point are also entered in the feed forward network. The network thus has 18 input nodes. The BiLSTM network uses the dataset with last samples from each sample window. The FNN network keeps up with the BiLSTM at times close to the blood culture test but at 24 hours upfront the difference is already remarkable.

6 Conclusion

This research explored whether models that are able to capture temporal effects can bring added value to predicting blood culture test outcomes. Those temporal models are clearly better in predicting the outcome of a blood culture test than non-temporal models. Temporal models have a slight advantage in predicting the outcome close to the time the blood sample test was taken but are noticeably better than other models in predicting this test many hours upfront. It’s clear that temporal physiologic data contains
a lot of useful information to early detect positive blood cultures. Using models that are able to capture this temporal data obviously brings an added value to the blood culture prediction problem.

REFERENCES


# Contents

1 Introduction

2 Literature study
   2.1 Blood Culture
   2.2 Sepsis
   2.3 Relevant Research
      2.3.1 Prediction of Sepsis
      2.3.2 Time Series Classification
   2.4 Conclusion

3 Machine Learning
   3.1 Neural Networks
      3.1.1 Forward Propagation
      3.1.2 Back Propagation
      3.1.3 Overfitting
   3.2 Recurrent Neural Networks
      3.2.1 Forward Propagation
      3.2.2 Back Propagation Trough Time
      3.2.3 Bidirectional RNNs
      3.2.4 Vanishing Gradient
   3.3 LSTM
      3.3.1 Why RNN over FNN?
      3.3.2 IMDB Sentiment Analysis

4 Data
4.2 Parameters ......................................................... 37
  4.2.1 Blood Pressure .................................................. 37
  4.2.2 Temperature ..................................................... 37
  4.2.3 Respiratory Rate ................................................. 38
  4.2.4 Lactate .......................................................... 38
  4.2.5 CRP-s ............................................................ 38
  4.2.6 Heart Rate ....................................................... 38
  4.2.7 Thrombocytes .................................................... 39
  4.2.8 WBC ............................................................. 39
  4.2.9 SOFA ............................................................ 39
  4.2.10 INR ............................................................ 40

4.3 Data Analysis ....................................................... 40
  4.3.1 Outliers ......................................................... 40
  4.3.2 Sequence Lengths ............................................... 41
  4.3.3 Histograms ....................................................... 43
  4.3.4 Positive vs Negative .......................................... 43

4.4 Data Processing ..................................................... 47

4.5 Limitations ........................................................ 48

5 Model ............................................................... 51
  5.1 Used Technologies ................................................ 51
    5.1.1 MySQL ......................................................... 51
    5.1.2 Python ......................................................... 52
    5.1.3 Theano ........................................................ 52
    5.1.4 Lasagne ....................................................... 52
  5.2 Architecture ....................................................... 53
    5.2.1 Unidirectional Recurrent Neural Network .................. 54
    5.2.2 Bidirectional Recurrent Neural Network ................... 56
    5.2.3 LSTM network ............................................... 56

6 Evaluation .......................................................... 59
  6.1 Prerequisites ...................................................... 59
    6.1.1 Evaluation Parameters ...................................... 59
6.1.2 Technical Information .............................................. 60
6.1.3 Data Partitioning .................................................. 60
6.1.4 Determining the best weight ratio ............................. 60
6.2 Results ........................................................................ 62
   6.2.1 Recurrent Neural Network ..................................... 62
   6.2.2 Bidirectional Recurrent Neural Network ...................... 64
   6.2.3 LSTM Network .................................................... 64
   6.2.4 Variations in Dataset ............................................ 68
   6.2.5 Early Prediction ................................................... 72
6.3 Overall Best Result .................................................... 75
6.4 Comparison with Non-Recurrent Networks .................... 76
6.5 Differentiating Parameters ........................................... 79

7 Conclusion & Future Work .............................................. 82
   7.1 Conclusion ............................................................ 82
   7.2 Future Work .......................................................... 83
Chapter 1

Introduction

Early detection of a positive blood culture with patients in the intensive care unit can be lifesaving. A positive blood culture can lead to inflammation throughout the body, a blood pressure dip or even organ failure and eventually death of the patient. The mortality rate can be reduced drastically when treatment is started within 12 hours after a positive blood culture test [29]. Nowadays, blood culture is measured by performing one or more venopunctures. The blood samples are then sent to the lab for analysis [2]. Several difficulties exist with detecting a positive blood culture. First: The doctor is the one who decides when a venopuncture is performed. He makes this decision based on data like heart rate, body temperature or else. The doctor decides to take the blood sample when his sentiment says the patient’s blood might be infected. However this might be a very difficult problem because symptoms are not always clear. The causes of blood infection are not yet fully understood so there is no clear direction in where a physician needs to pay attention to. A second difficulty is that the analysis of blood samples can take some time. As treatment needs to be started as soon as possible, time is scarce and there is need for a fast detection method. A last obstacle is that the outcome of a blood culture test is not always right, there might be false positives as well as false negatives. Those can influence the possible start of a treatment or be the beginning of a faulty treatment.

Although the detection of culture positive patients is hard and symptoms are not always clear, doctors do have the presumption that a relationship between a patient’s physiological values such as temperature, heart rate, etc and the occurrence of a positive blood culture exists. Capturing this complex relationship is hard-to-reach for the human brain but might be possible for a computational model. This research explores if those computational models can assist in
detecting culture positive patients or suggest new relationships.

Literature presents several techniques to detect sepsis. Sepsis is a condition related to a positive blood culture and detection thereof could be similar to detecting blood culture. Literature does not mention the use of models that can handle time-dependent data. Those models provided an added value to numerous applications. Monitored patient data is nevertheless time dependent and the use of temporal models can mean an improvement in blood culture detection.

This dissertation presents a model for blood culture detection based on recurrent neural networks, which is a computational model that is designed to work with time-dependent data. The intensive care unit (ICU) of Ghent University Hospital provided a dataset containing temporal information from over two-thousand admissions. This research paper will explain the different aspects of a blood culture prediction model and will evaluate its performance.

The research goal for this dissertation thus consists of two parts: Exploring if computational models can assist in detecting positive blood culture patients and verifying if using temporal models on physiologic data brings added value to this detection compared to non-temporal models.
Chapter 2

Literature study

2.1 Blood Culture

People entering the Intensive Care Unit (ICU) have severe injuries or infections caused by accidents or serious illness. The patient’s situation can get even worse if the infection spreads to other organs. Bacteria or fungi that cause those infections could make their way into the blood. As this happens, infection spreads throughout the body and can do serious harm.

Patients where infection spreads throughout the body often get nauseous, develop fever, start breathing rapidly, their heart rate rises, get confused... These signs can make a doctor aware of a possible infection whereby he can advance to extensive tests. One such test is the blood culture test. Blood cultures are used to detect the presence of bacteria or fungi in the blood, to identify the type present and to guide the treatment. If a blood culture is positive for that particular bacteria or fungi, it’s likely for the person tested that he has a blood infection for that type of micro-organism. A blood infection is most likely when the immune system is weak. This can occur with infants and older adults, or from a disease like AIDS or cancer.

To test for an infection in the blood, a sample of blood is collected and placed in a cup with special substances that allow the bacteria or fungus to grow. Two or three blood samples from different veins are often taken to make sure a bacteria or fungus is not missed. If no bacteria or fungus grows, the blood culture is called negative. A blood culture test is performed done when a person has a fever because this is the time when the bacteria or fungus is most likely to have spread to the blood.
The type of bacteria or fungus that grows can be checked by identifying them under a microscope or by more advanced techniques. Semi-automated blood culture systems exist, like the radiometric Bactec system [39]. In recent years, they are enhanced with fully automatic continuously monitoring systems for the detection of microbial growth. Some bacteria grow easier than other and are therefore easier to detect. Nowadays systems can identify most bacteria and fungi between one hour and two days. Although supplementary time and additional tests may be necessary for slow-growing or metabolically inert organisms. Stevenson et al. [39] also proposed a new technique based on spectrometry which identifies bacteria within one hour.

### 2.2 Sepsis

As infectious organisms spread into the blood, human body can go into systemic response. This is called the systemic inflammatory response syndrome (SIRS) or Sepsis [31]. Sepsis occurs when chemicals released in the bloodstream to fight the infection start causing a cascade of reactions in other parts of the body which can damage multiple organs.

Different stages of sepsis exist. Differentiating normal sepsis, severe sepsis and septic shock. A person diagnosed with sepsis has at least two of the following symptoms (SIRS indicators):

- Body temperature above 38.3 degrees Celsius or below 36 degrees Celsius.
- Heart rate greater than 90 beats per minute.
- A respiratory rate higher than 20 breaths per minute.
- White Blood cell Count (WBC) > 12,000/mm$^3$ or < 4,000/mm$^3$

Severe sepsis is defined as the presence of sepsis and one or more organ dysfunctions. Septic shock is defined as the presence of sepsis and a drop in blood pressure. The hospital mortality rate for patients with severe sepsis or septic shock was 28.6%, as rated by Bryant Nguyen et al. [31]. Sepsis is especially dangerous for people with compromised immune systems or very young or old people.

In a prospective study conducted by Rangel-Frausto et al. Positive blood cultures were found in 17% of patients with sepsis, in 25% with severe sepsis, and in 69% with septic shock whereas
20% to 30% of patients will have no microbial cause identified from any source [34]. After the onset of sepsis, the effectiveness of intervention with antibiotics or other therapeutics rapidly diminishes [26]. Thus making accurate identification of patients at risk for developing these conditions is crucial to improving standards of clinical care.
2.3 Relevant Research

2.3.1 Prediction of Sepsis

Several studies have been conducted in the domain of computer science engineering to elaborate the detection of sepsis with the use of machine learning techniques. From those papers the used techniques can be learned, an analysis of the observed parameters can be conducted and the size of the dataset and the effectiveness thereof can be identified.

Numerous papers describe models that try to predict the occurrence of sepsis or a septic shock. This problem can be easily transformed to the prediction of positive blood cultures due to the similarity between the input data for both problems. With both problems there is time-series dependent data that is monitored from the patient. The output is given by a label that classifies a patient with sepsis present/not present or a positive/negative blood culture.

The Multi parameter Intelligent Monitoring in Intensive Care II (MIMIC II) [36] database was used in some papers to provide input for the described systems. The MIMIC II database is a publicly available resource which provides data on > 30,000 patients in the ICUs of Bostons Beth Israel Deaconess Medical Center between 2001 and 2007. The clinical records include charted physiological measures, medication records, fluid input and output records, laboratory test results, procedure orders, and free-form text notes produced for each of the > 30,000 stays recorded in the database.

Ho et al. [15] used the MIMIC II database to construct a predictive model for sepsis. The model used two feature sets for patients, one that consisted of the patient’s clinical history and the other contained non-invasive measurements of four physiological variables (Heart Rate (HR), Respiratory Rate (RR), Blood Pressure (BP) and Pulse Oxymeter (SpO2)). Variables included in the first set were demographic information (age, gender), medical history and basic health information (weight and physician calculated SOFA and SAPSI scores). Only variables measured within 6 hours after admission to the ICU were considered. Imputation methods were used to cover the missing data points. In this way a relation between sepsis and the initial state of the patient when entering the ICU could be derived. Logistic regression was used to make the prediction. This technique achieved an AUC (Area Under Curve under ROC or Receiver Operation Characteristic) score of 0.823. What may be a more interesting and more applicable
### 2.3 Relevant Research

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mean/recent value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Systolic Blood Pressure (SBP)</td>
<td>recent &amp; mean</td>
</tr>
<tr>
<td>Heart Rate (HR)</td>
<td>recent &amp; mean</td>
</tr>
<tr>
<td>Pulse Pressure (PP)</td>
<td>recent &amp; mean</td>
</tr>
<tr>
<td>Respiratory Rate (RR)</td>
<td>recent &amp; mean</td>
</tr>
<tr>
<td>Pulse Oxymeter (Sp0₂)</td>
<td>recent &amp; mean</td>
</tr>
<tr>
<td>Temperature (TEMP)</td>
<td>recent &amp; mean</td>
</tr>
<tr>
<td>White Bloodcell Count (WBC)</td>
<td>recent</td>
</tr>
<tr>
<td>Arterial pH</td>
<td>recent</td>
</tr>
<tr>
<td>Shock Index = HR / SBP</td>
<td>recent</td>
</tr>
</tbody>
</table>

Table 2.1: Physiologic and laboratory features for predicting septic shock used by Ho et al. [15].

Component in this paper is the prediction of septic shock. This problem tries to predict the occurrence of a septic shock by taking mean and most recent values as input. Beside the variables used in sepsis prediction as mentioned above, extra parameters are taken into account. An overview can be found in table 2.1. Observations are made 120, 60 and 30 minutes before onset of septic shock. Patients with no septic shock are given an onset time halfway between their first and last available BP measurement.

Three different models are used for Septic shock prediction: Multivariate logistic regression, a linear kernel support vector machine (SVM) and a regression tree with respective AUC scores of 0.847, 0.882 and 0.867 for the prediction 60 minutes before shock onset. Taking a look at different times before onset, it’s clear that prediction becomes better the closer to the shock onset. 30 minutes upfront, an AUC with logistic regression of 0.853 is achieved. 60 minutes upfront gives 0.847 and for 120 minutes the AUC is 0.829. This model is more representative for the research goal in this dissertation because it makes predictions for different timestamps. However, the predictions are made on just a single snapshot in time. Correlations between different timestamps are not taken into account although it is mentioned in future work.

A somewhat similar approach was taken by Mani [27] where late-onset (+72h) neonatal sepsis was predicted using a dataset of 299 infants. A blood culture was taken to further subdivide in
2.3 Relevant Research

positive and negative culture sepsis. 60 temporal values were gathered each 6h hours ranging from 48 hours before to 12h after the first blood test. Those values together with 30 non-temporal variables (demographics, birth weight, gestational age, Apgar scores, mode of delivery etc.) are provided to a machine learning system to classify the neonatals. Temporal values included blood counts, oxygen saturation, heart rate and respiratory rate. First some feature selection algorithms are used to select the highly predictive features from the large dataset. The best features are provided to a set of classifiers consisting of: SVM, Naive Bayes (and variants), averaged one dependence estimators (AODE), K-nearest neighbour algorithm, decision and regression trees, Logistic regression and random forests. Five fold cross-validation is used to select the best classifier. the SVM model often comes out as best performing one. Although this method has high sensitivity (recall, up to 95%), the specificity or true negative rate was rather low (47%) which means a lot of predictions are positive where the actual condition is negative.

A paper where relations between different time samples are taken into account is written by Kim et al. [21]. A dataset of 1213 sepsis patients and 26 non-sepsis post-operative patients is used to train an SVM. Since time series deal with variable lengths and missing values, they cannot be fed to the SVM directly. Therefore a feature vector is created first. The feature vector contains the mean, slope, standard deviation, range, max negative change and max negative change values and is calculated on the SIRS parameter time series (WBC, HR, temperature, RR). In addition to the four vital signs, the data included demographic information including age, height, weight and ASA (American Society of Anesthesiologists) classification for each patient. This method achieves an AUC score of 0.95, 0.92, 0.90 and 0.90 for respectively 0, 3, 6 and 12 hours in advance.

Apart from only measuring clinical values as described above, cytokine and chemokine blood values may also be defining factors. Lukaszewski [26] trained an Artificial Neural Network (ANN) on those parameters in combination with previously discussed clinical parameters. This model is able to correctly predict patient outcomes in an average of 83.09% of patient cases between 4 and 1 days before clinical diagnosis with sepsis. Aside from the standard clinical values, the levels of expression of interleukin-1 (IL-1), IL-6, IL-8, and IL-10, tumor necrosis factor, FasL, and CCL2 mRNA are measured by real-time reverse transcriptase PCR. This study uses a dataset of only 92 patients whereof 22 diagnosed positive. To gather the special blood values, blood samples
had to be taken daily, which is an intrusive operation. Relations in time dependent data are not
considered in here.

A recent paper proposes a real-time early warning score for septic shock (TREWscore) [14].
This method uses the MIMIC II dataset as input. A sample of input data for the system can
be found in figure 2.1. The TREWscore over time for the provided input is showed in figure
2.2. The score prediction contains two detection criteria, a first one is triggered when the score
exceeds a threshold value. The second one is activated when the score stays above the thresh-
old for a certain amount of time. The score is calculated by first calculating features on the
patient-specific measurement streams, which are fed to a supervised learning algorithm. Which
specific techniques they use is not mentioned but because features are calculated on measurement
streams it’s clear they did not use models for temporal data. TREWScore identifies patients
before the onset of septic shock with an area under the receiver operating characteristic (ROC
AUC) of 0.82. The TREWscore is compared to other identification methods. A first one is the
MEWS, a score to identify medical patients at high risk of catastrophic deterioration. Although
MEWS is not specifically developed for tracking sepsis, MEWS has been used to facilitate the
identification of patients at risk for severe sepsis and septic shock. A second comparison is made
between a routine screening protocol. Figure 2.3 shows an ROC curve comparing the perfor-
mance of all methods. In a test environment, 60.9% of patients with septic shock were first
discovered by the TREWscore, while only 21.8% were first discovered by routine screening. The
remaining part was discovered at the same time (9.9%), or not discovered at all (7.5%).

2.3.2 Time Series Classification

While previous papers suggest to use different clinical patient values, none of them use the time
series provided by the manual measurements and monitoring systems and provide them as a
whole to a classification system. Some papers calculate feature vectors on the input, but the
real dependencies were not exploited. None of the above discussed papers suggest to use models
that are capable of modelling temporal information. The sepsis or blood culture prediction task
might lend itself to the use of these models. Because the absence of sepsis prediction papers
where those models are used, this section will present some applications that do make use of
these models and where they provided an added value. There will not be an extensive explana-
tion on the working and mathematical background of the models here, this is covered in chapter
3 about machine learning.

Various classifiers can be used on time series, but few can handle aspects as variable lengths.
2.3 Relevant Research

Figure 2.3: ROC curve comparing the TREWscore, MEWS and routine protocol

or multivariate inputs. Papers suggest usage of SVMs or ANNs for various purposes such as vowel or ECG classification [42] [44]. To use these SVMs or ANNs for variate length time series, feature vectors have to be calculated or the series have to be patched to a same length. Those processes can produce a loss in data. Other paper suggest the use of Recurrent Neural Networks (RNN) for time series classification. Those networks make efficient use of temporal information in time series and can be used for classification or prediction [8] [11] [12] [18] [19] [20] [24]. For the remaining of this chapter the focus will be on classification and prediction examples with RNNs and RNN variants.

Character-Level Language Models

Karpathy describes in his blog how RNNs can be trained to form a text generator. He created a Shakespeare generator which generates sequences of text indistinguishable from real Shakespeare work. All the original texts were concatenated into one text file. The network trains each character at a time and thus learns dependencies between characters and not words. It is basically learning a language from scratch. Find below an example text generated by the network.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder’d at the deeds,
So drop upon your lordship’s head, and your opinion
Shall be against your honour.

This result can not be obtained by an artificial neural network or any other machine learning technique bespoken in the previous section because it would generate a repetitive sequence from the moment it receives a previous seen input. A recurrent neural network will not fall in this repetitive situation since its output relies not only on a current input but on ”all” previous inputs.

This relates to the blood culture problem as the text input can be seen as time series data. The characters can be modelled as numbers or labels. However, this is more a prediction as it is a classification problem because not one label is assigned to the sequence but a new sequence of new labels is created. There are as many inputs to the network as there are characters, where a ‘1’ is provided if the according character is entered and a ‘0’ if not (one-hot encoding). A multivariate input of binary sequences is created. In the blood culture problem the inputs will not be binary but will have real values.

Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) is an output layer for recurrent neural networks specially designed for temporal classification tasks; that is, for sequence labelling problems where the alignment between the inputs and the target labels is unknown [11] [12]. It can be applied to a speech signal for phoneme recognition. A first method is discussed where a network activated a certain phoneme class for the length of the phoneme sound. The network is trained on segments of one phoneme. Problems arise in transitions from one phoneme to another or the network predicted two sequences of the same phoneme while it should only output one longer sequence. External post-processing algorithms are needed to eliminate those flaws. CTC achieves classification by allowing the network to make label predictions at any point in the input sequence, so long as the overall sequence of labels is correct. This removes the need for presegmented data since the alignment of the labels with the input is no longer important. Moreover, CTC directly outputs the probabilities of the complete label sequences, which means that no external post-processing is required to use the network as a temporal classifier. Figure 2.4 illustrates the
difference between CTC and framewise classification.

CTC achieves this by adding a blank label to the list of predictable phonemes. A blank label means there is no different sound than the previous or next one, or no sound at all. Blank labels are left out in the final predicted phoneme sequence. In this way, different paths can lead to the same sequence label e.g. \( a - ba - - - = -a - b - a \rightarrow aba \). This "collapsing" of paths makes it possible for CTC to use unsegmented data, it allows the network to predict the labels without knowing in advance where they occur.

Classification for one phoneme is similar to the Blood Culture Classification (BCC) problem, although this network receives only one parameter input stream. But it is, as BCC, a many-to-one classification of an input stream. The spiked output may be useful to the BCC system in the fact that a patient may evolve from culture negative to culture positive in a given timespan. Temporal classification may detect a positive-like segment and output a positive spike which can be used in a monitoring system to label a patient as positive.

**EEG Classification**

The electroencephalogram (EEG) signals reflect the electrical activity of the brain. Those EEG signals provide useful information to separate epileptic patients from non-epileptic ones. Guler et al proposes a technique using RNNs to perform this classification task [13]. Signals are pre-processed to a 4 dimensional sequence of Lyapunov exponents. Those sequences are fed to a RNN which was trained to classify the sequences in three different classes (healthy, seizure free...
epileptogenic zone, epileptic seizure). This method achieves an accuracy of 96.79%. The sequences here are all of the same length, which makes it possible to also try them on a Multilayer Perceptron Neural Network (MLPNN), which is similar to an ANN. The RNN model predicts the sequences almost 5% better.

This problem is almost equal to blood culture classification, one label is given to a multivariate sequence input. Apart that this problem needs some preprocessing which might not be needed for blood cultures. The sequence lengths for each patient will also vary in the BCC case.

**Sentiment Analysis**

An other example of how RNNs can be used for classification is sentiment analysis. A Theano tutorial shows how to implement an LSTM architecture to predict the sentiment of IMDB (Internet Movie Database) reviews [6]. LSTM (Long-Short Term Memory) is an advanced version of RRN which can hold information longer and see long term dependencies. To predict the sentiment, words are first converted to a number which is readable for the system. The number is just an identifier and the numerical value has no special meaning. A sequence of numbers is created which is used to train the network and make predictions. What is special to this solution is that the many-to-one classification is done in a special manner. Each time step provides an output. This is similar to predictions resulting in a new generated output sequence. To classify this to one label mean pooling is used. All predictions come together and the label which is most common will be the resulting label. This is one way how the label in BCC could be determined.

### 2.4 Conclusion

This chapter started with explaining that a blood culture test is a test to measure if bacteria of fungi are present in the blood stream. This can cause serious harm to organs. A patient with a positive blood culture can develop (severe) sepsis or septic shock.

Detecting positive blood cultures or sepsis as soon as possible is key in ameliorating the chances for recovery. Several papers suggest early detection mechanisms for sepsis based on monitored values and general information about a patient. Since sepsis and blood cultures are related those same mechanism may be applied to predict the outcome of a blood culture test.
However, none of those methods make use of advanced machine learning techniques that are designed to classify temporal information. Some applications are presented using so-called recurrent neural networks or LSTMs to learn and predict on time series. The techniques in those applications can be employed for predicting the outcome of a blood culture test.
Chapter 3

Machine Learning

This chapter introduces different learning models capable of modelling temporal information. It starts by explaining how normal neural networks work and how they can be trained. Normal neural networks are not optimally suited to deal with sequential information but the concept behind them is similar to the ones in recurrent neural networks. Understanding neural networks is a prerequisite for advancing to recurrent neural networks and that is why they are treated first. After normal and recurrent neural networks have been addressed, a more advanced form of recurrent neural nets is introduced, long short-term memory.

3.1 Neural Networks

Artificial Neural Networks (ANN or NN) were originally developed to represent the mathematical information processing capabilities of the human brain. It’s now clear that neural networks have little resemblance to real biological neurons. But they are very popular as pattern classifiers.

A neural network is a network of small processing units, nodes joined to each other with a weighted connection. Many neural networks have been developed over the years. An important distinction between classes is the occurrence of cycles. NNs with cycles are called recurrent neural networks. Those that are acyclic, thus containing no cycles are called Feed forward Neural Networks (FNN).

Networks typically consist of three layers: the input, hidden and output layer. The input layer is the one where input patterns come in. The information flows form the input layer to the
hidden layer and then to the output layer. An activation function is applied to the input of each node in the hidden and output layer. This activation function is often a sigmoid:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

Or a logistic sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

This activation function actually maps the input value of the neuron to a value between -1 and +1 for the sigmoid or between 0 and +1 for the logistic sigmoid function. It provides a sort of normalized value to the neuron. Near the zero input, those functions are very steep, which means when their input value changes a small amount. Near the ends, when the function approaches its horizontal asymptotic values the steepness of the function is nearly zero.

### 3.1.1 Forward Propagation

A provided input ripples through the network and changes according to different calculations that are applied on it. In figure 3.1 an overview of a feed forward neural network is given. Figure 3.2 zooms in on the actual operation of one neuron. First, the neuron is provided with all the inputs. Those inputs are first multiplied by a weight value. All those weighted inputs come together in the transfer function of the neuron. This transfer function is usually just a summation. This sum of weighted inputs together with a threshold or bias $\Theta_j$ is provided to the activation function. The result of this activation function is the output of the neuron and thus the result that ripples to the next layer in the network. The input to the activation function is labelled as $net_j$, with $j$ the index of the $j$’th hidden neuron.
3.1 Neural Networks

Figure 3.2: Model of one artificial neuron

\[ \text{net}_j = \sum_{ij} w_{ij} \cdot x_i \] (3.1)

The output or activation of the neuron becomes:

\[ o_j = \varphi(\text{net}_j + \Theta_j) \] (3.2)

Or with the logistic sigmoid as activation function \( \varphi \):

\[ o_j = \frac{1}{1 + e^{-\sum_{ij} w_{ij} \cdot x_i - \Theta_j}} \] (3.3)

This process is applied to all neurons in the hidden layer. The output of the hidden layer is on its turn transferred to the output layer. This happens in the same manner as from input to hidden, the same formulas apply, although a different activation function can be used.

To have a network predict something useful, the network first needs some training. The training process boils down to providing a network with inputs for a given output and changing the inner weights of the network in such way that the network outputs the wanted result for that input. This process is called back propagation and will be explained in the next section.

3.1.2 Back Propagation

The process of back propagation is quite complex and takes some time to understand deeply. Here, the basics will be explained. A detailed and extensive explanation of the back propagation algorithm can be found in the free online book of M. Nielsen about neural networks and deep learning [32], in chapter two he gives a clear explanation of the learning process.
As been said, training a network boils down to changing its inner weights in such a way that a provided input results in a wanted output. A possible way to do this is to make small adaptations to those weights and see how the output reacts to this. If the output changes in the direction of the wanted output, the adaptations will improve the network. To get the optimal result, each combination of weight adaptations must be tried and evaluated. The combination that moves the most in the direction of the right output will be chosen and those adaptations are carried through. This method would work but it has a major drawback: It’s very computationally expensive and inefficient. Imagine a network with 5 inputs, 10 hidden neurons and 1 output which has $5 \cdot 10 + 10 = 60$ connections and thus 60 weights, excluding biases. This yields a combination of $2^{60} = 1.15 \cdot 10^{18}$ different possibilities for weights to be incremented or decremented and even many more if the weight could also be left unchanged. This would be computationally immense and it would take far too long to train a network properly, even for a network this little. Luckily, more advanced methods exist.

If $y(x)$ represents the desired output for an input $x$, the goal is to find an algorithm which finds the weights and biases so the output of the network approximates $y(x)$ for all training inputs $x$. To quantify how well this goal is achieved, a cost function is defined:

$$C(W, b) = \frac{1}{2n} \sum_x \| y(x) - a \|^2$$  \hspace{1cm} (3.4)

Here, $w$ denotes the collection of all weights in the network, $b$ all the biases, $n$ is the total number of training inputs, $a$ is the vector of outputs from the network when $x$ is input, and the sum is over all training inputs, $x$. Of course, the output $a$ depends on $x$, $w$ and $b$. $C$ is called the quadratic cost function or Mean Squared Error (MSE). Other cost functions such as Logistic Loss:

$$C(W, b) = \sum_x log(1 - e^{ay(x)})$$  \hspace{1cm} (3.5)

can be used too. Each function has its own characteristics in terms of learning speed and suitability for certain problems. MSE will be used to explain the working of the back propagation algorithm since it is the most common and simple one.

The actual goal in training a neural network is to find weights and biases which minimize this quadratic cost function $C(w, b)$. It might be easier to understand the process of minimizing
if the actual form of the cost function is forgotten and to assume it is just a function of many variables. This function has to be minimized. It could be a real valued function of variables \( v = v_1, v_2, \ldots \). To minimize \( C(v) \) it helps to visualize \( C \) as a function of just two variables. A plot is found in figure 3.3.

![Cost function visualization](image)

Figure 3.3: Cost function visualization. Presented as a function of two variables \( v_1 \) and \( v_2 \). Finding the minimum in this plot is an easy task, but for a cost function with many variables it will not be possible to eyeball the graph and find the minimum. A possible way to find the minimum is use calculus and find a minimum analytically. Since the output of the network is calculated by a sequence of mathematical functions, computing derivatives and using them to find extrema would be possible. However, having so many variables makes the calculation of extrema very complicated and thus infeasible.
3.1 Neural Networks

**Gradient Descent**

An actual solution is a combination of the above methods, calculating derivatives and making small adaptations to the weights so the output of the network moves in the direction of the desired one. The solution is called gradient descent. It’s analogue to choosing a random point in the valley plotted in fig 3.3 and letting a ball roll step by step in the direction of the lowest point until it reaches it. The direction is the calculated gradient, representing the local ‘shape’ of the valley. The size of the step is determined by a predefined parameter and is called the learning rate. Having a high learning rate will train the network faster but adds to the possibility of overshooting the optimum and oscillating around it. The gradient $\nabla C$ is the vector

$$\nabla C \equiv (\frac{\partial C}{\partial v_1}, \ldots, \frac{\partial C}{\partial v_m})^T$$

(3.6)

With partial derivatives for each of the m variables. The applied change to the variables is:

$$\Delta v = -\eta \nabla C$$

(3.7)

with $\eta$ the learning rate. This results in the following update formula:

$$v \rightarrow v' = v - \eta \nabla C$$

(3.8)

For a neural network, the idea is to use gradient descent to find the weights $w_k$ and the biases $b_l$ which minimize the cost in equation 3.4. Writing out the gradient descent update rules in terms of components gives:

$$w_k \rightarrow w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$$

(3.9)

$$b_l \rightarrow b'_l = b_l - \eta \frac{\partial C}{\partial b_l}$$

(3.10)

Repeatedly following this update rule will hopefully result in finding a minimum of the cost function.

**Stochastic Gradient Descent**

Looking back at the cost function, it can be noticed it is of the form $C = \frac{1}{n} \sum_x C_x$, that is, it’s an average over costs $C_x$ for individual training samples. In practice, to compute the gradient $\nabla C$, the gradient $\nabla C_x$ for each input sample $x$ has to be computed separately and averaged afterwards. If the training set is large this can take a long time.
Stochastic gradient descent can be used to speed up training. The idea is to estimate the gradient $\nabla C$ by computing $\nabla C_x$ for a small set of randomly chosen training inputs. This set is called a mini-batch. The network is trained on this mini-batch and updates are applied. Then a new random set of input samples is chosen to train the network with and so on, until the training set is exhausted which is called a complete epoch of training. At this point, a new training epoch is started.

### 3.1.3 Overfitting

Similar to most machine learning models, neural networks are also prone to overfitting. This happens when the weights in a network will be shaped in such a way that they give optimal prediction for the training examples but not for unseen data. The network will have memorized the training set but it has not learned to generalize to new situations. Therefore the training goal is to minimize the test error, which is the error when a network makes predictions for a set of new, unseen samples. Methods exist to prevent overfitting, but these will not be handled here.
3.2 Recurrent Neural Networks

What are Recurrent Neural Networks (RNNs)? The concept behind RNNs is using sequential information. In traditional neural networks, the output only relies on the input at that time while in RNNs the output can rely on the entire history of inputs. RNNs assume there is a correlation between the inputs and tries to capture this correlation. For example: If the next word in a sentence has to be predicted, it might be useful to know which words came prior it. RNNs can be seen as having an internal ‘memory’ of previous inputs which influence the network output.

3.2.1 Forward Propagation

An internal ‘memory’ is obtained by allowing cyclical connections in the network which transfer information from a previous state to the next one. To get proper insight in what happens, the network can be unrolled (see fig 3.4). Unrolling (or unfolding) simply means writing out the network for the complete sequence.

![Figure 3.4: A recurrent neural network and the unfolding in time of the computation involved in its forward computation [23].](image)

The formula’s that govern the computations in an RNN are as follows:

- $x_t$ represents the input at time step $t$. For example, $x_t$ could be a patient temperature value at time $t$.

- $s_t$ is the hidden state at time $t$. It is calculated based on the input of the current step and the hidden state of the previous step according to the next formula: $s_t = \varphi(Ux_t + Ws_{t-1} - \Theta_j)$. 

The function $\varphi$ is an activation function. Usually a tanh or logistic sigmoid function. Mind the relation with formula 3.2. Here an extra term for the recurrent connection is added.

- $o_t$ is the output at step $t$. In language prediction, this could be the predicted word at time $t$. $o_t = \varphi(Vs_t)$.

The RNN uses the same parameters (U, V, W) at each step. This is different from the traditional neural network, which uses different parameters at each layer. Using the same parameters each step means that the same tasks are performed at each step, this greatly reduces the amount of parameters needed to be trained.

Figure 3.4 has outputs at each time step, this might not be necessary. As for the blood culture prediction task, all inputs are mapped to only one output, only the final output of the network is important. Similarly, inputs might not be needed at each time step, the hidden state can propagate itself. An example of this is text prediction based on only one input character, like the shakespeare generator discussed in section 2.3.2.

Training an RNN uses the same principles as training a feed forward neural network. The training process also requires unfolding the network over time. Learning the RNN over time is called Back Propagation Trough Time (BPTT).

### 3.2.2 Back Propagation Trough Time

To quantify how well the model is achieving a cost or loss function is defined. Remember the Mean Square Error (MSE, 3.4) from feed forward neural networks. Here the cross entropy loss is introduced. Cross entropy loss is introduced because it leads to simple derivations and formulas, which make it easier to understand the concept of BPTT rather than being stuck with mathematical computations. Bear in mind that various loss functions can be used. Using a different cost function will not deeply affect the equations beneath, only some final steps. The cost is named $E$. $o_t$ is the predicted output at time $t$ and $y_t$ is the wanted output at that time.

$$E_t(y_t, o_t) = -y_t\log(o_t)$$  \hspace{1cm} (3.11)
With $E_t$ the error for the output at time step $t$. Typically the full sequence is trained as one training sample. The total error becomes the mean of the sum of the errors at each time step:

$$E(y,o) = \frac{1}{t} \sum_{t} E_t(y_t, o_t)$$  \hspace{1cm} (3.12)$$

$$= -\frac{1}{t} \sum_{t} y_t \log(o_t)$$  \hspace{1cm} (3.13)$$

The goal is to calculate the gradients of the error with respect to the parameters $U$, $V$ and $W$ and then learn good parameters using stochastic gradient descent. Just like the errors are summed, the gradients at each time step for one training example are also summed up:

$$\frac{\partial E}{\partial W} = \sum_{t} \frac{\partial E_t}{\partial W}$$  \hspace{1cm} (3.14)$$

To calculate these gradients, the chain rule of differentiation is used. It is the back propagation algorithm from the previous chapter applied backwards starting from the error. To guide trough the formulas, back propagation form $E_3$ is used as an example. First remember:

$$s_t = \varphi(Ux_t + Ws_{t-1} - \Theta_j)$$ \hspace{1cm} (3.15)$$

$$o_t = \varphi(Vs_t)$$ \hspace{1cm} (3.16)$$

Usually, the activation function for the hidden layer is a $\tanh$ function, and for the output a $\text{softmax}$ function. Those equations then become:

$$s_t = \tanh(Ux_t + Ws_{t-1} - \Theta_j)$$ \hspace{1cm} (3.17)$$

$$o_t = \text{softmax}(Vs_t)$$ \hspace{1cm} (3.18)$$

Figure 3.5: Recurrent neural network with errors unfolded over time.
With fig 3.4 and 3.5 in mind, following formulas are obtained:

\[
\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial o_3} \frac{\partial o_3}{\partial V} = \frac{\partial E_3}{\partial o_3} \frac{\partial z_3}{\partial o_3} \frac{\partial z_3}{\partial V} \quad (3.19)
\]

\[
\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial z_3} \frac{\partial z_3}{\partial W} \quad (3.20)
\]

Where \( z_3 = V s_3 \). In fact, those equations tell that \( \frac{\partial E_3}{\partial V} \) only depends on the values at the current time step \( o_3, y_3 \) and \( s_3 \). The story is different for \( \frac{\partial E_3}{\partial W} \) and for \( U \). Writing out the chain rule as above brings clarification:

\[
\frac{\partial E}{\partial W} = \frac{\partial E_3}{\partial o_3} \frac{\partial o_3}{\partial s_3} \frac{\partial s_3}{\partial W} \quad (3.21)
\]

\( s_3 = \tanh(U x_3 + W s_2) \) depends on \( s_2 \), which depends on \( W \) and \( s_1 \) and so on. \( s_2 \) can’t simply be seen as a constant when deriving for \( W \). The chain rule has to be applied again and again. This results in this:

\[
\frac{\partial E}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial o_3} \frac{\partial o_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W} \quad (3.22)
\]

The contributions from each time step to the gradient are summed up. A schematic overview can be seen in fig 3.6.

Figure 3.6: Contribution from each time step to \( E_3 \).

It is very similar to the standard back propagation used in feed forward neural networks. The key difference is that the gradients for \( W \) are summed up at each step. In a traditional NN, parameters are not shared across layers, so nothing has to be summed. The same update
3.2 Recurrent Neural Networks

Equations as in FNNs apply (eq. 3.9). After time step 3 this becomes:

\[ W \rightarrow W' = W - \eta \frac{\partial E_3}{\partial W} \]

(3.23)

This update can also be performed after passing the whole sequence, \( W \) can than be updated only once with the summation of all the gradients.

\[ W \rightarrow W' = W - \eta \sum_{k=0}^{5} \frac{\partial E_k}{\partial W} \]

(3.24)

This is the basic process for training a recurrent neural network. The steps taken above to explain this BPTT algorithm are based on a blog post of Denny Britz [5] who refers to [23] and [33].

3.2.3 Bidirectional RNNs

An improvement on regular recurrent neural networks is a Bidirectional Recurrent Neural Network (BRNN). A BRNN is an extension to a regular RNN in the sense that it is not only trained based on the inputs from the past time steps, but also up to a pre-set future time frame. It was described by Schuster et al. in 1997 [38]. The overall performance of this network for regression and classification task was better compared to the results gained from a unidirectional recurrent neural network.

The network consists of a forward passing hidden layer and a backwards passing hidden layer. The topology unrolled in time can be seen in fig. 3.7. The state neurons from the regular RNN are split into a part that is responsible for the positive time direction (forward states) and a part for the negative time direction (backward states). Outputs from forward states are not connected to inputs of backward states, and vice versa. Note that without the backward states, the network is just like a unidirectional RNN.

The BRNN can principally be trained with the same algorithms as a regular unidirectional RNN because there are no interactions between the two types of state neurons and, therefore, can be unfolded into a general feed forward network. However, the back propagation trough time (BPTT) algorithm is slightly more complicated than in a regular RNN. The forward and backward pass is more difficult because the updates of state and output nodes cannot be done one at a time. Training needs to be done in slices of a number (\( T \)) of time steps. The training is
split up in three parts: first a forward pass (from input to output), then a backward pass (back propagation of the gradient) and finally updating the weights:

1. **Forward Pass** Run all input data for one time slice (from 1 to T) through the BRNN and determine all predicted outputs.

2. **Backward Pass** Calculate the gradient for the time slice used in the forward pass. Propagate the gradient through the network.

3. **Update Weights**

### 3.2.4 Vanishing Gradient

Although RNNs and BRNNs can be very effective for learning patterns in sequences, they have difficulties learning long-range dependencies or interactions between values that are several steps apart. This is problematic because those dependencies might be very important. A fever peak 24 hours before a blood culture test might be a determinant for the outcome of this test, but it is unlikely that a unidirectional recurrent neural network will capture this dependency. To understand why this is happening, take a look at the gradient calculated above (eq. 3.22, or see below). The vanishing gradient problem is explained on the basis of RNNs but it is also applicable to BRNNs.

$$\frac{\partial E}{\partial W} = \sum_{k=0}^3 \frac{\partial E_3}{\partial o_3} \frac{\partial o_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W} \tag{3.25}$$
3.2 Recurrent Neural Networks

Note that $\frac{\partial s_3}{\partial s_k}$ is a chain rule in itself. For example, $\frac{\partial s_3}{\partial s_1} = \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial s_1}$. Thus, the above gradient can be rewritten:

$$\frac{\partial E}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial o_3} \frac{\partial o_3}{\partial s_3} \left( \prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}$$

(3.26)

[16] and [33] explain in detail that the influence from hidden values on the gradient shrink exponentially fast with how many steps they are away. This can be explained by looking at the tanh function and its derivative in fig 3.8.

![Figure 3.8: tanh and derivative.](image)

The tanh (or sigmoid) activation function maps to values between -1 and 1. The derivative is bounded by 0 and 1 (or 1/4 for the sigmoid). Note that the tanh derivative has 0 values at both ends, they approach a flat line. If the values in the neurons end up at one of those ends the neurons are called saturated. They have a zero gradient and drive other gradients from previous layers to zero. Thus with different gradient multiplications (for $\prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}}$ in particular) the gradients shrink exponentially fast, and eventually vanish completely after a few steps. Gradient contributions from far away become zero and the state at those steps contribute nothing to learning the parameters for $W$. The fading influence of a node on the outputs is seen in fig. 3.9.
Fortunately, there are ways to solve this vanishing gradient problem. Regularization and proper initialization of the \( W \) matrix can reduce the effect of fading gradients. A more effective and popular solution is using Long Short-Term Memory (LSTM). First proposed by Sepp Hochreiter in 1997 [17]. Since LSTMs will be used in later chapters, they are discussed in the next section.

### 3.3 LSTM

Vanishing gradients are the main motivation behind the LSTM model which introduces a new structure called a memory cell (see Figure 3.10 below). A memory cell is composed of four main elements: an input gate, a neuron with a self-recurrent connection (a connection to itself), a forget gate and an output gate. The self-recurrent connection has a weight of 1.0 and ensures that, barring any outside interference, the state of a memory cell can remain constant from one timestep to another. The gates serve to modulate the interactions between the memory cell itself and its environment. The input gate can allow an incoming signal to alter the state of the memory cell or block it. On the other hand, the output gate can allow the state of the memory cell to have an effect on other neurons or prevent it. Finally, the forget gate can modulate the memory cell’s self-recurrent connection, allowing the cell to remember or forget it’s previous state, as needed [6].

The equations below describe how a layer of memory cells is updated at every time step \( t \).
Where:

- \( x_t \) is the input to the memory cell layer at time \( t \)
- \( W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o \) and \( V_o \) are weight matrices.
- \( b_i, b_f, b_c \) and \( b_o \) are bias vectors.
- \( h_t \) is the hidden value for time \( t \).

First, the values for \( i_t \), the input gate are calculated together with \( \tilde{C}_t \), which represent the candidate values for the memory cell at time \( t \).

\[
i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3.27}
\]
\[
\tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3.28}
\]

Second, \( f_t \), the activation of the memory cell’s forget gates at time \( t \) is computed.

\[
f_t = \sigma(W_t x_t + U_f h_{t-1} + b_f) \tag{3.29}
\]

Given the value of the input gate activation \( i_t \), the forget gate activation \( f_t \) and the candidate state value \( \tilde{C}_t \), \( C_t \) can be computed, the memory cell’s new state at time \( t \):

\[
C_t = i_t \cdot \tilde{C}_t + f_t \cdot C_{t-1} \tag{3.30}
\]

With the new state of the memory cells, the value of their output gates and, subsequently, their outputs can be computed:

\[
o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o) \tag{3.31}
\]
\[
h_t = o_t \cdot \tanh(C_t) \tag{3.32}
\]
3.3 LSTM

Training an LSTM relies on the same principle as training a traditional neural network and a recurrent neural network. Although many more parameters need to be trained than in a basic RNN. Gradients for each of those parameters can be computed and weights will be updated with those gradients.

Gating is the feature that differentiates LSTMs from simple recurrent neural networks. By using gates, the network can let inputs and their respective gradients from several time steps before ripple through the network without being faded away. A nice example of gating can be seen in Fig. 3.11 where the hidden layer contains gates. A circle means that the gate is open and a stripe means a closed gate. Only the first input ripples through because the input gates are closed in the subsequent time steps.

![Figure 3.11: Gating in an LSTM](image)

3.3.1 Why RNN over FNN?

The great advantage about using RNNs is that it is optimally suited to be used on temporal data. A feed forward neural network will also work if this temporal data has a fixed length. However this is seldom the case in many applications that use sequential data, remember the Shakespeare generator or the sentiment analysis system from the literature study. Even though the sequences have a fixed length or can be transformed into sequences that have fixed length a recurrent neural network will be a good choice. This is because a RNN is much better and efficient in finding temporal relationships. An example: suppose an application needs classification of 20
values long sequences. An FNN would need 20 inputs for this network while a recurrent neural net only needs one. If both networks use 10 hidden nodes this means \(10 + 10 = 20\) connections for the RNN (10 for input to hidden and 10 for hidden to hidden, if no cross-over from one hidden node to an other is used) and \(10 \times 20 = 200\) for the FNN and thus 180 more weights that need to be optimized. The RNN is thus much more efficient for time dependent data. The RNN can become more complex and powerful if cross-over is applied but this comes at the expensive of learning speed.

### 3.3.2 IMDB Sentiment Analysis

Here, an example of how an LSTM network can be used to perform classification is described. The goal is to do sentiment analysis on movie reviews. A review is a sequence of words where one class label has to be assigned. In that sense it is comparable to the blood culture classification task where a single class label is assigned to one or multiple parameter sequences. The word sequence is first transformed into a numerical sequence using a word-index dictionary.

Those numerical sequences are fed into an LSTM network. The network that is used in [6] is a little different than a unidirectional LSTM network in the sense that the output does not depend on the internal memory state of the cell. Eqn. 3.31 is replaced by eqn. 3.33. This is done to make the network and it’s computations less complex.

\[
a_t = \sigma(W_ox_t + U_oh_{t-1} + b_o)
\]  
(3.33)

The model is composed of a single LSTM layer followed by a mean pooling layer which pools the outputs of the LSTM cells over time and takes the average of those outputs. The outputs coming from the cells are called \(h_0, h_1, h_2...h_n\) and form a representation sentence. The representation sentence is then averaged over all time steps resulting in a representation \(h\). Finally this representation is fed in to logistic regression layer which assigns a class label that accords with the input sequence. This process can be found schematically in fig. 3.12.
Figure 3.12: LSTM network for sentiment analysis
Chapter 4

Data

4.1 Introduction

The Intensive Care Unit (ICU) from Ghent University Hospital started collecting patient data in 2013. A database containing time dependent information from over 2000 patients was constructed and is available for research under special ethical conditions. The database holds general information about a patient such as gender, admission time, discharge time, mortality ... As well as information about more specific measurements. A special table contains every monitored value with a time stamp, patient identification, the value itself and the variable that is measured.

The different tables, their columns and the relationships between them can be seen Fig. 4.1. Four different tables can be found here: rep38_cohort, rep38_results, rep38_biolimits and rep38_medication. Rep38 is just an identifier for the database. rep38_cohort holds the general patient information. Each patient has a unique admissionID. Rep38_results contains all the monitored patient values where there is a one-to-n relationship with rep38_cohort based on the patient’s admissionID. rep38_medication is similar to rep38_results, although it does not contain monitored values but administered medication for a patient at a specific time. Finally there is a rep38_biolimits table which contains the minimum and maximum possible values for a measured variable (vartype). This can be used to detect false values or outliers.

The results table contains more than 14 million entries from 2177 patients. Data is available for 29 different variables, those can be subdivided into 10 different groups. An extensive explanation for each variable will given in the next part. Each variable has its own monitoring frequency
A special column in the cohort table is the first_sampletime column. This column contains the value of the first positive blood culture sample. If a patient actually has a positive blood culture at a given moment, a value will be present in this column. If there was never a positive blood culture test for a patient, this field will just have a NULL value. This column is important because it defines a patient’s label. This separates positives from negatives. Patients with a positive blood culture will be seen as trues, ones or positives. Patients with a negative blood culture are defined as falses, zeros or negatives.

What is also important to note is how negatives are defined. Negatives are actually patients where blood culture tests have been taken from but that only had negative test outcomes. So perfectly healthy patients are not represented in this dataset since doctors only take a blood culture test if they suspect a patient to have a positive blood culture. The test thus draws a clear line in between the doubtful patients. Negative patients can be classified as negative with certainty, or the same certainty as the how reliable the culture test is.
4.2 Parameters

Parameters can be divided into 10 different groups. Find below an explanation for each group of variables, which specific metrics are measured and how they might contribute or relate to a positive blood culture

4.2.1 Blood Pressure

Two variants of blood pressure are measured, SAP (Systemic Arterial Pressure) and NIBP (Non-Invasive Blood Pressure). They each have systolic (SAPs, NIBPs), diastolic (SAPd, NIBPd) and a mean (SAPm, NIBPm) variant. Systolic blood pressure is the arterial pressure on contraction of the heart, while diastolic blood pressure is the pressure in between contractions. They are related with the following formula: $P_m = P_d + (P_s - P_d)/3$.

Non invasive techniques such as the known wristband or band around the upper arm are the easiest to use systems and therefore, the oscillometric devices are the most common used systems in the ICU unit, they are however less accurate than more intrusive techniques such as the auscultatory technique [35], where pressure is measured inside the artery.

A blood infection is often paired with low blood pressure, as a situation develops into sepsis or severe sepsis, a serious drop in blood pressure might occur. Blood pressure is thus an important factor to take into account and might signal the occurrence of a positive blood culture.

4.2.2 Temperature

Temperature values in the database are obtained by either manual measurements or automated probe measurements. For automated temperature there are entries for esofalgal, central, rectal and bladder temperature. The way temperature is measured depends on which was the easiest at the moment.

Fever often indicates that something is wrong in the human body and can point to an infection. The opposite, a very low temperature or chills, can also mean an infection is happening. Temperature might thus also be a decisive factor in predicting blood cultures.
4.2 Parameters

4.2.3 Respiratory Rate

Respiratory rate (RR) can be derived from an electrocardiogram (ECG) signal using special algorithms [28]. This derived RR is available in the dataset together with 2 other, more intrusive methods, RRv(s) and RRv(m). RRv values are measured on a respirator machine. RRv(s) represents the value on which the machine is set while RRv(m) is the actual respiration rate. Respiratory rate is a simple parameter that captures the amount of breaths per minute. Patients with infections or fever often start to breath faster.

4.2.4 Lactate

Blood lactate is a lactic acid that appears in the blood as a result of anaerobic metabolism when oxygen delivery to the tissues is insufficient to support normal metabolic demands [30]. It is common when conducting physical exertion.

In patients with septic shock, serial determinations of blood lactate levels are good predictors of the development of multiple system organ failure and death [3]. Blood lactate can thus also be a predictor of a positive blood culture.

4.2.5 CRP-s

C-Reactive Protein serum (CRP-s) is a substance produced by the liver in response to inflammation. A high level of CRP in the blood is a sign that there may be an inflammatory process occurring in the body which might be spread into the bloodstream.

A CRP test is a blood test designed to measure the amount of CRP in the blood. A CRP test only needs a blood sample.

4.2.6 Heart Rate

The heart rate is one of the vital signs. A rapid heartbeat may come from physical exercise but a fever or infection also goes paired with a raise in heart rate. A heart rate that is too low can result in a cardiac failure. Heart rate is automatically measured with an ECG-probes on a patient’s chest.
4.2 Parameters

4.2.7 Thrombocytes

Platelets (thrombocytes) are colorless blood cells that help blood clot. Platelets stop bleeding by clumping and forming plugs in blood vessel injuries. Thrombocytopenia is a condition in which a patient has a low blood platelet count. In rare cases, the number of platelets may be so low that dangerous internal bleeding occurs. The unit in the dataset is $\times \frac{1000}{\mu l}$.

Severe bacterial infections involving the blood (bacteremia) may lead to destruction of platelets which leads to Thrombocytopenia. Taking a look at the blood platelet count might thus be a determining factor to establish a good blood culture prediction.

4.2.8 WBC

Often, a complete blood count (CBC) is ordered along with or prior to the blood culture to determine whether the person has an increased number of white blood cells, indicating a potential infection. A high white blood cell count is an increase in disease-fighting cells in the blood.

The exact threshold for a high white blood cell count (WBC) varies from one laboratory to another. In general, for adults a count of more than 11,000 white blood cells (leukocytes) in a microliter of blood is considered a high white blood cell count or leukocytosis. The unit in the dataset is $\times \frac{1000}{\mu l}$.

4.2.9 SOFA

The sepsis-related organ failure assessment (SOFA) score is created to describe quantitatively and as objectively as possible the degree of organ dysfunction/failure over time. The SOFA score is designed not to predict outcome but to describe a sequence of complications in the critically ill [41].

The total SOFA score consists of several partial scores, each with a score from 1 to 4. Partial SOFA scores are: respiration, coagulation, liver, cardiovascular, Central Nervous System (CNS) and renal. For the exact score calculation, a reference is made to the original paper by vincent et al. [41].
4.2.10 INR

The prothrombin time – along with its derived measures of Prothrombin Ratio (PR) and International Normalized Ratio (INR) are assays evaluating the extrinsic pathway of coagulation. The prothrombin time is the time it takes plasma to clot after addition of tissue factor. INR is a derived ratio where a value of 1 represents a healthy person. A higher ratio means the coagulation happens slower than normal. The international normalized ratio is available in the dataset. Sepsis always leads to deranged coagulation [37], which makes it an interesting factor to watch for blood culture prediction.

4.3 Data Analysis

A great way to cope with a machine learning problem is to profoundly analyse and study the data. In the previous part the available parameters and some very basic facts were discussed. Here, more details about the variables will be taken into account and some numbers and distributions will be covered.

A first help on how the machine learning model has to look and how the data can be processed for the system to learn is to look at a variety of parameters or distributions belonging to the data itself. The dataset contains 2177 patients with each a set of parameter sequences. Fig. 4.2 contains a list of parameters belonging to each monitored variable.

4.3.1 Outliers

An early indicator of how well the data is structured is looking at the missing or false values. Bio-limits are provided for some variables, the table (fig. 4.2) shows the values that are outside the bio-limits in red. The maximum temperature value in dataset is 37638 degrees celsius, which is probably a typing mistake. To clean the dataset, values that fall outside this bio-limit should be removed. Some values are NULL values, those also have to be removed. Analysis shows that from the 14.6 million entries in the results dataset, there are 40250 NULL values or values outside the bio-limits. This boils down to 0.276% outliers, which is rather marginal. Those false values are present in 1797 admissions.
4.3.2 Sequence Lengths

A possible parameter to determine which size is appropriate for the network is the sequence length, or how many samples there are available for a specific variable. This amount varies greatly from variable to variable. SOFA scores are only generated once a day, while heart rate is available every 2 minutes. There is on average 4 days of patient data available, this results in having an average sequence length of 3.5 for SOFA, while the average length for heart rate is 1505. Other variable sequence lengths lie between those. To make data useful to the network, sequences should be converted to all having the same frequency of presence. This will be achieved by sampling the data. Sampling will be covered in a next section.
## Data information

<table>
<thead>
<tr>
<th>Vartype</th>
<th>Min value</th>
<th>Max value</th>
<th>Average</th>
<th>Median</th>
<th>Biolimits min</th>
<th>Biolimits max</th>
<th>admissions where present</th>
<th>Avg # samples</th>
<th>Avg Period (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>blood lactate</td>
<td>3</td>
<td>155,699997</td>
<td>17,927957</td>
<td>13</td>
<td>165</td>
<td>1,7</td>
<td>0,69999998</td>
<td>40280</td>
<td></td>
</tr>
<tr>
<td>blood leukocyte count</td>
<td>0,07</td>
<td>259,399994</td>
<td>13,29689</td>
<td>11</td>
<td>2100</td>
<td>3,1</td>
<td>36937,8884</td>
<td>37,1034368</td>
<td>125</td>
</tr>
<tr>
<td>blood thrombocyte count</td>
<td>4</td>
<td>1640</td>
<td>196,24509</td>
<td>171</td>
<td>2103</td>
<td>3,1</td>
<td>36937,8884</td>
<td>37,1034368</td>
<td>125</td>
</tr>
<tr>
<td>CRP-s</td>
<td>0,69999999</td>
<td>694,400024</td>
<td>125,980911</td>
<td>92</td>
<td>2000</td>
<td>2,8</td>
<td>40280</td>
<td>35819</td>
<td></td>
</tr>
<tr>
<td>HR(ecg)</td>
<td>0</td>
<td>8894</td>
<td>86,7932936</td>
<td>30</td>
<td>2175</td>
<td>1505,5</td>
<td>1662</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>INR protrombinetijd</td>
<td>0,76999998</td>
<td>16,079999</td>
<td>1,4864366</td>
<td>13</td>
<td>2094</td>
<td>2,8</td>
<td>34790</td>
<td>35043</td>
<td></td>
</tr>
<tr>
<td>NIBPd</td>
<td>0</td>
<td>176</td>
<td>68,2488144</td>
<td>67</td>
<td>200</td>
<td>57,2</td>
<td>34790</td>
<td>1523</td>
<td></td>
</tr>
<tr>
<td>NIBPm</td>
<td>0</td>
<td>208</td>
<td>89,2456397</td>
<td>88</td>
<td>1674</td>
<td>56,8</td>
<td>34790</td>
<td>1527</td>
<td></td>
</tr>
<tr>
<td>NIBPs</td>
<td>0</td>
<td>1137286</td>
<td>132,467439</td>
<td>119</td>
<td>1692</td>
<td>60,6</td>
<td>34790</td>
<td>1443</td>
<td></td>
</tr>
<tr>
<td>RR(ecg)</td>
<td>0</td>
<td>5336</td>
<td>22,6995889</td>
<td>20</td>
<td>2167</td>
<td>994,9</td>
<td>34790</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>RRV(m)</td>
<td>0</td>
<td>730</td>
<td>21,0529057</td>
<td>20</td>
<td>1068</td>
<td>41,0</td>
<td>34790</td>
<td>2662</td>
<td></td>
</tr>
<tr>
<td>RRV(s)</td>
<td>0</td>
<td>540</td>
<td>18,7880901</td>
<td>18</td>
<td>970</td>
<td>11,5</td>
<td>34790</td>
<td>8713</td>
<td></td>
</tr>
<tr>
<td>SAPd</td>
<td>-411</td>
<td>657</td>
<td>63,095855</td>
<td>60</td>
<td>1892</td>
<td>1244,3</td>
<td>34790</td>
<td>121</td>
<td></td>
</tr>
<tr>
<td>SAPm</td>
<td>-259</td>
<td>1200</td>
<td>83,9313426</td>
<td>81</td>
<td>1892</td>
<td>1095,4</td>
<td>34790</td>
<td>137</td>
<td></td>
</tr>
<tr>
<td>SAPs</td>
<td>-131</td>
<td>184183</td>
<td>127,462661</td>
<td>124</td>
<td>1892</td>
<td>1502,3</td>
<td>34790</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>serum C-reactive protein oud</td>
<td>0,89999998</td>
<td>572,200012</td>
<td>140,620072</td>
<td>122</td>
<td>93</td>
<td>3,0</td>
<td>34790</td>
<td>35383</td>
<td></td>
</tr>
<tr>
<td>SOFA_CARDIO</td>
<td>0</td>
<td>1,38271442</td>
<td>1</td>
<td>1</td>
<td>2166</td>
<td>3,5</td>
<td>34790</td>
<td>86270</td>
<td></td>
</tr>
<tr>
<td>SOFA_CNS</td>
<td>0</td>
<td>1,21345179</td>
<td>0</td>
<td>0</td>
<td>2166</td>
<td>3,5</td>
<td>34790</td>
<td>86270</td>
<td></td>
</tr>
<tr>
<td>SOFA_COAGULATION</td>
<td>0</td>
<td>0,50640941</td>
<td>0</td>
<td>0</td>
<td>2166</td>
<td>3,5</td>
<td>34790</td>
<td>86270</td>
<td></td>
</tr>
<tr>
<td>SOFA_LIVER</td>
<td>0</td>
<td>0,2982688</td>
<td>0</td>
<td>0</td>
<td>2166</td>
<td>3,5</td>
<td>34790</td>
<td>86270</td>
<td></td>
</tr>
<tr>
<td>SOFA_RENAL</td>
<td>0</td>
<td>0,51222413</td>
<td>0</td>
<td>0</td>
<td>2166</td>
<td>3,5</td>
<td>34790</td>
<td>86270</td>
<td></td>
</tr>
<tr>
<td>SOFA_RESPIRATORY</td>
<td>0</td>
<td>1,21223735</td>
<td>1</td>
<td>1</td>
<td>2166</td>
<td>3,5</td>
<td>34790</td>
<td>86270</td>
<td></td>
</tr>
<tr>
<td>SOFA_RESPIRATORYD</td>
<td>0</td>
<td>1,44486035</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>3,5</td>
<td>34790</td>
<td>86400</td>
<td></td>
</tr>
<tr>
<td>SOFA_TOTAL</td>
<td>0</td>
<td>5,12594159</td>
<td>4</td>
<td>4</td>
<td>2166</td>
<td>3,5</td>
<td>34790</td>
<td>86270</td>
<td></td>
</tr>
<tr>
<td>Temp blais</td>
<td>0</td>
<td>37637,8884</td>
<td>37,1034368</td>
<td>37</td>
<td>1055</td>
<td>998,0</td>
<td>37,1034368</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>Temp centr.</td>
<td>0</td>
<td>88</td>
<td>37,0415436</td>
<td>37,0784988</td>
<td>29</td>
<td>197</td>
<td>63,9</td>
<td>1234</td>
<td></td>
</tr>
<tr>
<td>Temp eso</td>
<td>7</td>
<td>379</td>
<td>37,0409327</td>
<td>37,2999992</td>
<td>29</td>
<td>245</td>
<td>610,5</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>Temp rectal</td>
<td>0</td>
<td>356</td>
<td>36,7185161</td>
<td>36,9000015</td>
<td>29</td>
<td>393</td>
<td>753,6</td>
<td>114</td>
<td></td>
</tr>
<tr>
<td>Temp( man)</td>
<td>0</td>
<td>36937,1992</td>
<td>39,2872744</td>
<td>37</td>
<td>1653</td>
<td>8,3</td>
<td>39,2872744</td>
<td>13729</td>
<td></td>
</tr>
</tbody>
</table>
4.3 Data Analysis

4.3.3 Histograms

The values from most parameters are distributed normally. Table (fig. 4.2) contains the mean and median values for each variable. A doctor can easily verify if these values make sense. Figure 4.3 shows a distribution for all values for all patients for the systolic Systemic Arterial Pressure (SAPs) variable. The best fitting normal distribution is shown in red. Most distributions like the SAPs distribution exhibit small positive skewness. The only variable that is not distributed normally is the Serum C Reactive Protein (CRP-s), the distribution can be seen in Fig. 4.4.

![SAPs Histogram](image)

Figure 4.3: Distribution of SAPs values

4.3.4 Positive vs Negative

So far only general information is considered. It might be useful to watch differences between positive and negative values. The dataset contains values from 229 culture positive patients and 1948 culture negative patients.

A first thing to discuss is how long a patient is recorded in the hospital. Looking at time between the discharge and admission times. The average patient stays at the ICU for 10.21
days, while it's stated that a patient with positive blood culture resides on average 27.76 days in the ICU. Negative patients only stay for 8.14 days on average. The presence of bacteria in the blood thus has a serious impact on the stay and the situation of a patient. A positive blood culture test is usually taken after 12.23 days of admission.

From these 27 days that a blood culture positive patient resides at the ICU, the dataset only contains values from up to 4 days before a positive blood sample. This is so because data from before those 4 days will not affect the blood culture (prediction) according to the medical specialists. The patient also needs to be at least 4 days at the ICU to be included in the dataset to ensure there is enough data and it is important to have the patient in a controlled environment, to exclude external factors to be the cause of the positive blood culture test.

A last parameter which might be useful to watch is the difference in variable distributions between positive and negative patients. The blood thrombocyte count parameter for example shows a difference. In fig 4.5 can be seen that positive patients have a mean count of 237 while negative patients have a mean of only 192. A difference can also be found in the SOFA score means, positive culture patients have a higher SOFA score than negative ones. This is logical
because positive patients are mostly in a worse shape. (fig. 4.6)

Figure 4.5: Boxplots that indicate the difference in distribution between negative and positive patients for the blood thrombocyte count variable

Other variables have not so clear distinction, positive patients have a slightly higher systemic arterial pressure (SAPs) but this difference is only limited (5%, see fig. 4.7). Those variables might not be so indicative themselves, but their variation over time or in combination with other parameters they can give lead to a useful prediction.
4.3 Data Analysis

Figure 4.6: Boxplots that indicate the difference in distribution between negative and positive patients for the SOFA-total variable

Figure 4.7: Boxplots that indicate the difference in distribution between negative and positive patients for the mean Systemic Arterial Pressure (SAPm) variable
4.4 Data Processing

The database contains a bunch of unstructured data sequences, those can not be fed to a learning model directly and have to be preprocessed first. Recurrent neural networks require a sequential input with one or more variables. If the input consists of multiple parameters, each of these sequences need to have an equal length.

After removing the outliers the data gets normalized. This is done for each variable with the following formula:

$$n = \frac{x - \text{avg}}{3 \times \text{std}}$$

(4.1)

Where $x$ is the value to be normalized, $n$ is the normalized value, $\text{avg}$ and $\text{std}$ are respectively the average and standard deviation of all values for that variable. This formula maps 99.8% of the values between -1 and +1 and the mean becomes 0.

The normalized sequences are then padded to equal length and with the same frequency by sampling them at predefined sampling points. The sampling frequency is a parameter that can be chosen. A first value is one sample per hour, this is a value in between most frequencies at which variables are available. SOFA has one sample a day while heart rate has one sample per 2 minutes, other variables have frequencies that lie within those bounds. One sample/hour is only an example first guess value. The results chapter discusses the influence of changing the sampling frequency.

The end of the sampled period is chosen as the moment when a first positive sample is established. If no positive sample is encountered, the sampling period can be chosen at random because a ‘healthy’ patient stays a ‘healthy’ one. For easy of use and to guarantee enough data points, the sampling end-point is chosen as the time of last value present for a patient. The begin of the sampling period is dependent on how long the sampled period lasts or for how far back data is available for an admission. If a patient has for one parameter measurements starting only 2 hours before the sampling end point there are two options: A first is to cut off all data sequences upto that point and uphold sequences that only contain 2 samples (if the frequency is one each hour). This would mean a loss of many useful data. An other and better option is to pad the restricted sequence with means (zeros due to normalization) and leave the other
sequences unchanged. In this way the data from the other sequences is not lost. The length of a sampled period is a parameter that can be played with.

The actual sampling happens as follows for each admission:

For each sampletime in sampletimes:
   For each variable:
      value = latest value where entrytime < sampletime
      add value to samples for variable

Here, the latest available (most recent) value for a variable is used. For some variables it might be useful to take the average, minimum or maximum value for all values that lie in the most recent sample window, it is: between the current sample time and the previous one. In this case a fever peak will not be missed.

The pseudo code then looks as follows:

For each sampletime in sampletimes:
   For each variable:
      latest = values where (previous_sampletime < entrytime < sampletime)
      add min/max/avg(latest) to samples for variable

Fig. 4.8 shows an example of raw data sequences for a patient. The sequence is cut to show the part that will be sampled. The blue lines are the times where the data will be sampled (sample times). The sequences depicted are not normalized to make the values more reasonable. It will be the normalized version that serves as input to the RNN. Fig. 4.9 then shows the result of sampling where the most recent data value up to a sample point is taken. This is an input than can be fed to a recurrent neural network.

4.5 Limitations

Although the data is very extended with over 14 million values, there are some limitations. The data is not equally consistent for each patient and for each parameter. There are 29 different variables, although not even half of them are usable or will be an added value for the prediction. Some variables are related to each other such as SAPd and SAPm, they have the same evolution over time so adding both variables to the input of the network will not result in a better
Figure 4.8: Example data sequences for 3 variables, with the blue lines as sample times (Temp Blaas = Bladder temperature)

Figure 4.9: Sampled data for sequences in fig. 4.8 (Temp Blaas = Bladder temperature)
4.5 Limitations

prediction.

An other limitation is the availability for some variables in admissions, some measurements like blood lactate are only present in 165 admissions. Since an empty input can not be provided to the network this variable can also not be added to the input set.

Bladder temperature is the temperature variable with the most samples present in the dataset. It is available for 1055 admissions. However, the difference between those temperature variables is just the way it is measured. Temperature will not be very different when measured centrally than when measured in the bladder. Physicians advised that all temperature variables could be joined to one. Grouping temperature variables will result in a potential number of useful admissions of 2161, which is 1106 admissions more then when using only bladder temperature.

The measuring frequency for some variables like SOFA is very low, only one per day. Since the evolution in the last hours before a culture test might be the most influential and predictive, the SOFA score will not change in a couple of hours so a constant value will be served to the network, unless the SOFA measurement happened just in those critical hours. The great strength of recurrent neural networks is that it can learn on the evolution and change in a sequence, this strength will not be exhibited when a nearly constant SOFA score is provided as input.

Despite the problems mentioned above, the dataset is fairly extensive and will be useful for the prediction task. From a all available variables a subset can be made which captures the variables with presence in greater part of the admissions and with ambiguous variables removed. This set will contain one variable from each of the in section 4.2 discussed groups except blood lactate because there are too few values.
Chapter 5

Model

Chapter 3 discussed the basics of (recurrent) neural networks and dug deeper into advanced RNNs. The theoretic fundamentals of Bidirectional RNNs and LSTMs were treated. In this chapter those concepts will be translated to a model that can learn to predict the outcome of a blood culture test. Two different machine learning models are translated into an architecture that is able to learn on multivariate time series and provide a positive/negative output based on those time series. The concepts used to train were a unidirectional recurrent neural network, a bidirectional recurrent neural network and an Long Short-Term Mermoy (LSTM) network. First, some information about the used technologies is provided because they have a certain influence in the design choices of the network architectures. Then the global architecture from data gathering to the learning model and result creation is discussed to get a high-level overview of the prediction machine.

5.1 Used Technologies

The main technologies used are MySQL to store the data, Python to program the functionality and Python libraries suited for mathematical operations and neural network tasks.

5.1.1 MySQL

MySQL is an open-source relational database management system [43]. It comes with a free and easy-to-use tool named ‘MySQL Workbench’ which enables the user to visually manage the databases and perform SQL queries on the data.
The data provided by the ICU was delivered in a SQL dump. MySQL provides easy functionality to import this dump and translate it into a working database. The main reasons that determined the choice for MySQL are its easiness to use and the facilitations to make it interact with Python through a library called pymysql.

### 5.1.2 Python

Python [10] is the core technology used in the prediction machine. Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. It is well documented and has high presence on Q&A sites such as Stack Overflow.

Additionally, it is also very suited for mathematical applications, what neural networks are in essence. Some very useful libraries exist to speed-up, optimize and facilitate the work and development of those mathematical operations. Excellent libraries are Theano and Lasagne, which will be discussed hereafter.

An other great advantage is Python’s portability: Python can be used cross platform and comes pre-installed with most Linux-distributions. Installing extra libraries is an easy task which makes it easy to transfer code to a new machine without much effort. This is a blessing when transferring the code for the computational work from a local computer to a GPU-server.

#### 5.1.3 Theano

Theano is a Python library that allows to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. It features transparent use of a GPU, allowing to speed up data-intensive operations to 140x faster than with CPU. [4]

Theano requires to define mathematical operations upfront before actually performing them. It is also preferred to define sizes of matrices and arrays upfront so Theano can allocate memory and space.

#### 5.1.4 Lasagne

An excellent Python library which serves as an abstraction layer on top of Theano to build and train Neural Networks is Lasagne [7]. It supports easy creation of different kinds of layers (dense
layers / convolutional layers / recurrent layers ...). And allows the use of different update and training schemes. Using Lasagne makes it able to focus more on the machine learning essentials and less on the model implementation in Theano, which can become quite complex.

5.2 Architecture

This section will give insight in the overall software architecture, which components are used and which design decisions are taken to implement those components. A detailed explanation will be given on what the neural networks look like and how different nodes and layers in them work together.

Figure 5.1: High level architecture of prediction machine

Figure 5.1 shows a high level architecture of the prediction machine. The main components of the architecture are:

**MySQL Database** Which contains all the patient data stored in a database. The interior of this database is discussed in chapter 4 and will not be further expounded here.

**Data Loader** The data loader serves as middleware layer to read the data and prepare it for processing in Python. The connection between MySQL and Python is provided through the pymysql library. This layer delivers the raw monitored values to the sampling component.

**Sampler** The raw monitored data doesn’t have the right format to serve as input for the network. To solve this, the data is sampled to make sure each variable has the same sequence length. The sampling happens according to the methods discussed in 4.4. The sampled data is stored in CSV files. This is because it’s much faster to load the (stripped!) data from CSV files than from the database. In this way, several models can be tested on the same data without having to gather the data from the database.
Data Preparation  This part reads the sampled data from the CSV files and processes them for the machine learning component. The data is also split in a training set, validation set and test set. This component gives the data to the machine learning model.

Recurrent Neural Network  Here the neural network is implemented. Each epoch the network is trained on the train set and validated on the validation set. After training, the test data is run through the network to evaluate the performance. Based on this test set, results are gathered.

Results  Based on the test set that is used on the learning model, some results are derived. This might be in the form of a ROC curve or some statistics in a CSV file (accuracy, sensitivity, auc)... Results will be discussed in the results chapter (ch. 6).

In the following part different kinds of learning models will be discussed, ranging from simple ones such as an RNN to more complex LSTMs. The theoretic background can be found in the machine learning chapter 3.

5.2.1 Unidirectional Recurrent Neural Network

The most basic network used for training is a vanilla RNN. Vanilla in the meaning of traditional or conventional, thus a simple recurrent neural network. This network consists of 1 input layer, 1 hidden layer and 1 output layer. The topology of an unfolded network can be seen in fig 5.2. This figure shows a network with 2 input nodes and 2 hidden nodes. This network can thus train on 2 input features (e.g. blood pressure & temperature). The number of hidden and input nodes can easily be adapted according to the needs of the learning task. $X_t[k]$ represents the $k$th input feature at time step $t$. Cross-over recurrent connections from 1 hidden layer to another are present, they add more learning capabilities to the network.

From the hidden layer only the last output is transferred to the output layer, the results from all hidden layers then come together in the output stage, where the final prediction is made.

Lasagne and Theano allow to define this network quite easily, however they require to have upfront defined lengths of the input matrices. For the blood culture prediction task the input would be a 3 dimensional matrix with size: \#admissions \times \#samples \times \#features. For example if trained on a batch containing 20 admissions, with 24 samples (one each hour) and 9 features
5.2 Architecture

Figure 5.2: Topology of an unfolded vanilla RNN with 2 input features and 2 hidden nodes

(1 from each parameter group in section 4.2, minus blood lactate) the size would be $20 \times 24 \times 9$. A problem arises when an admission doesn’t have enough samples to span 24 hours which lead to a shorter sequence length. To solve this, the shorter sequence can be padded with zeros and the hidden state is only updated if there is a valid input at a certain time step. Implementation-wise lasagne has an option to add a mask input to the network. This mask will represent the valid values in the admission sequence with a one. The padded values will be designated with a zero in the mask sequence. Adding this mask sequence is an option in lasagne, when the mask input is zero, the network will hold its current state and will not advance to a next time step. In this way a network can be trained on variable sequence lengths and still perform optimal trough defining fixed size inputs.

The data comes from the sample reader in a train, validation and test set. The data flows trough the network as follows: The training set is first subdivided in training batches, which are used to train the network using the Adam update scheme [22]. There are several training epochs, each training epoch the network gets trained on the whole training set but this set gets subdivided in new random set of training batches. Each epoch also contains a validation set to validate the network on unseen data. In this, the validation set is run through the network (without training) and the predicted results are compared to to the ground truth. The validation set is a small set which can be used to rapidly check the performance of the network. The test set is much larger and gives a more accurate view on the performance of the network. When all training epochs are finished, the test set is pushed trough the network to get a final result and performance of the network. It’s possible to add functionality for early stopping if the validation score starts
to rise when overfitting occurs. By default the network trains for a predefined amount of epochs.

To measure to what extent a predicted result differs from the wanted result, a cost function is defined. A mean square error (MSE) variant is used. Using a normal MSE will not result in good predictions because the true/false ratio is not 1. There are only 229 positive patients while there are 1958 negative. If a normal MSE would be used on this warped ratio, a network might always output zeroes and will be right for ±90% of the time. To correct this a higher error weight is ascribed to a prediction that should be positive. This is done using the following formula:

\[
C = w_{\text{true}} \cdot \sum_{x=X_{\text{true}}} ||y(x) - a||^2 + w_{\text{false}} \cdot \sum_{x=X_{\text{false}}} ||y(x) - a||^2
\]

The parameters \(w_{\text{true}}\) and \(w_{\text{false}}\) represent respectively the true and false weights, \(X_{\text{true}}\) represents the inputs that come from a positive blood culture and \(X_{\text{false}}\) represents the inputs from negative culture patients. The determination of the best \(w_{\text{true}}/w_{\text{false}}\) ratio will be discussed in the results chapter (6).

### 5.2.2 Bidirectional Recurrent Neural Network

An improved form learns dependencies in both the forward and backward direction using a Bidirectional RNN. This requires only small changes to the code because most of the functionality and topology is the same as with the previous network: The vanilla RNN. Only an extra hidden layer for the backward passing is added. A concat layer concatenates the outputs from both hidden layers into one output ‘array’ but this is rather a formality for implementing it in lasagne. How the network looks like can be seen in figure 5.3. The learning scheme and cost functions are the same as the vanilla RNN.

### 5.2.3 LSTM network

A more complex version uses LSTM cells instead of hidden nodes. The LSTM cell has the advantage that it can memorise dependencies over more time steps compared to the vanilla RNN. In this network, which can be seen in figure 5.4, the input cells are connected to the memory cell input of the LSTM cell. Only the output from the last time step gets delivered to the output layer.
5.2 Architecture

Figure 5.3: Topology of an unfolded Bidirectional RNN with 2 input features and 4 hidden nodes (2 forward, 3 backward)

Just as a backward layer can be an enhancement for the vanilla RNN, an extra backward layer of LSTM cells can be added to the ‘vanilla’ LSTM network. A Bidirectional Long Short-Term Memory network is born. Whether this extra layer actually improves the network is discussed in the next chapter. This type of networks is previously used by Ales Graves for handwriting recognition [25] or music processing [9]. It has also an application in protein localization, written down by Thireou et al. [40].
This BiLSTM can be upgraded by adding an extra hidden layer. This layer is a feed forward layer that collects all recurrent states at each time step. Adding this layer can give more influence to states from early time steps. In this way, the evolution of the data will be used better in the output and peaks in time might have more influence because they don’t have to ripple through the whole LSTM chain. Figure 5.5 shows how adding such a layer to a BiLSTM network looks like. Adding this extra layer makes the network conceptually similar as the network used in the IMDB sentiment analysis application except that the sentiment analysis model uses a pooling and regression layer instead of a feed forward neural network. The sentiment analysis network is seen in figure 3.12.

Figure 5.5: Topology of an unfolded BiLSTM network with 2 input features and 2 LSTM cells and an extra hidden feedforward layer.
Chapter 6

Evaluation

6.1 Prerequisites

6.1.1 Evaluation Parameters

The two validation parameters to quantify how well a network predicts are the Receiver Operating Characteristic (ROC) curve and Precision Recall (PR) curve. The ROC curve, is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The true-positive rate is also known as sensitivity, or recall in machine learning. The false-positive rate is also known as the fall-out and can be calculated as (1 - specificity). The precision-recall curve plots the precision against the recall.

\[
\text{Precision} = \frac{TP}{TP + FP} \\
\text{Recall} = \frac{TP}{TP + FN} \\
\text{Specificity} = \frac{TN}{TN + FP} \\
\text{Fall-out} = 1 - \text{Specificity} = \frac{FP}{TN + FP}
\]

Precision represents how useful the positive predictions are and recall represents how complete the positive predictions are. For medical applications, completeness might be a more important aspect than precision, it’s sometimes better to find all sick patients and classify some healthy patients as sick than to be right in saying patients are healthy but missing a couple of sick ones. Capturing the actual precision/recall rate boils down to choosing an appropriate threshold value (as is: choosing a split point which dictates whether the output of the network is classified as either positive or negative). However, this choice is out of the scope of this thesis and is a choice.
to be made by physicians when actually implementing a prediction/classification system. The Area Under Curve (AUC) value of a curve is as its name says, the surface under the curve. The auc lies between 0 and 1, 1 represents a perfect prediction while a 0 means the model predicts everything wrong. A random predicting model would have an auc value of 0.5.

### 6.1.2 Technical Information

Model training and evaluation is conducted using an NVIDIA Tesla K40c server GPU with a computing power of 1,43 Tflops. For reference, training a vanilla RNN network with 300 hidden nodes as seen in figure 5.2 using a train set of 1110 sequences with 24 samples per sequence takes on average 0.745 seconds per training epoch (one epoch is running the whole training set through the network). Training the same network only using a 2.8 GHz Intel Core i7 CPU takes on average 3.6 seconds. Up-sizing the network to 1000 hidden nodes takes the CPU already more than 46 seconds while the GPU trains that same network in 2.1 seconds.

### 6.1.3 Data Partitioning

The results in this section are obtained using a set of 9 features. This set is chosen in such a way that each variable group is delegated by one variable except for the blood lactate group. The blood lactate group is not represented because there are too few admissions in which values for this variable are present. The variables in table 6.1 are included in the set (please refer to section 4.2 for a detailed explanation).

Because all variables need to have values present to make an admission usable, only a subset of the original amount of admission can be used. Those 9 variables are present in 1761 admissions. This set is split into a train, test and validation set with a split ratio of respectively 0.63, 0.3 and 0.07. The train set size then is 1110, the validation set size is 123 and the test set consists of 528 admissions.

### 6.1.4 Determining the best weight ratio

Intuition would say that the ratio true weight / false weight should be inverse proportional to the amount of true admissions / the amount of false admissions. This results in a ratio around \( 9 \approx \frac{1958}{229} \). However when testing an RNN with this ratio the prediction model tends to give
### 6.1 Prerequisites

**SAPm**
- Systemic Arterial Pressure is chosen because it is the most used in the ICU and also the most accurate method. The mean value is chosen because it is a combination of systolic and diastolic pressure and the evolution of the mean relies on both.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blood Thrombocyte Count</td>
<td></td>
</tr>
<tr>
<td>CRP-s</td>
<td></td>
</tr>
<tr>
<td>SOFA total</td>
<td>SOFA Total because it combines all SOFA scores.</td>
</tr>
<tr>
<td>Heart Rate (ecg)</td>
<td></td>
</tr>
<tr>
<td>Respiratory Rate (ecg)</td>
<td></td>
</tr>
<tr>
<td>INR Protrombine Time</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>This variable combines all different temperature values (bladder, eso ...) to one sequence length, it’s possible to take all temperature variables together because the way of measuring doesn’t have a great impact on the values.</td>
</tr>
</tbody>
</table>

Table 6.1: Variables included in the set

Testing all ratio’s in a range of 1 to 10 gives an indication which might be an appropriate choice. Fig. 6.1 shows the Receiver Operator Characteristic (ROC) Area Under Curve (AUC) and the auc of the Precision Recall (PR) curve values for all weight ratios tested on a feedforward RNN. The network is trained for a predefined amount of 100 epochs.

Those curves do not have the expected shape, the curve shows some random small peaks and valleys while a concave mountain curve would be more intuitive. The cause might be the randomized choice of batches which have an impact on the learning process or the differences in network parameter initialisation. Each network learns in a slightly different way. However, some peak values can be distinguished around a weight ratio of 2-3 or near 7-8 in both curves. For the proceedings of this chapter, tests are conducted using a weight ratio of 8.

As in vanilla RNNs (and also in BiRNNs) the weight ratio has an impact on the performance of the network. Conducting the same test on an LSTM network learns that this ratio doesn’t have an impact. The network trains equally well no matter which weight ratio is chosen. See figure 6.2, there is a small dip in precision recall auc around a weight ratio of 7, but this might
6.2 Results

6.2.1 Recurrent Neural Network

Training a vanilla recurrent neural network with 1000 hidden nodes and 24 samples (one each hour) results in a test ROC auc of 0.797 and a precision recall (PR) auc of 0.493 (see fig.6.3).
6.2 Results

(a) ROC curve  
(b) Precision Recall curve

Figure 6.3: Results of training a vanilla RNN on the dataset for 200 training epochs

The network was trained for 200 epochs and in each epoch the whole training set is run through the network in batches of size 10. Figure 6.4 shows a learning curve for this network. After 200 epochs the network seems saturated. There might even be some overfitting given the rise in the validation loss. At 60 epochs the learning should have been stopped but the ROC auc (which is also calculated on the validation set) curve doesn’t show any signs of overfitting.

Figure 6.4: Learning curve of training a vanilla RNN on the dataset for 200 epochs
6.2 Results

6.2.2 Bidirectional Recurrent Neural Network

Training a more advanced version of the vanilla RNN, the bidirectional recurrent neural network yields a slightly improved prediction. The ROC auc rose to 0.818 and the PR auc rose to 0.529. The curves can be found in figure 6.5. The improved prediction can be explained because a BiRNN actually has a vanilla network in it. It has a surplus backward layer which can provide the extra 0.02 auc.

(a) ROC curve  
(b) Precision Recall curve

Figure 6.5: Results of training a Bidirectional RNN on the dataset for 200 training epochs

Figure 6.6 shows the learning curve for the bidirectional recurrent neural network. There is again a slight rise in the validation loss near the last epochs but the ROC auc doesn’t diminish. The PR auc however shows a light fall.

6.2.3 LSTM Network

The Long Short-Term Memory (LSTM) network has the most advanced building blocks and is expected to yield better results than a more simple RNN. Fig 6.7 shows the result from a trained LSTM network. The network performs slightly better than the BiRNN. The long term dependencies that an LSTM can capture might thus have influence in the prediction. The ROC auc from the LSTM is higher but the precision recall is not. The PR curve from the BiRNN shows higher precision for low recall values, the surface under those recall values is a great part of the total surface or area under curve and that is why the BiRNN PR auc is so much higher
6.2 Results

Figure 6.6: Learning curve of training a bidirectional RNN on the dataset for 200 epochs than for the LSTM network. The precision for higher recall on the other side is higher with the LSTM network. In other words: the BiRNN network is better in differentiating the positives from the negative patients while the LSTM network is more complete in finding all positives.

Figure 6.7: Results of training an LSTM network on the dataset

Training the LSTM network happens in a different manner than training the simple recurrent neural networks. The LSTM network did not produce the best results when trained with an
equal low learning rate as the RNNs. It trained well but only produced similar ROC auc as the BiRNN. However, when the LSTM network was trained with a higher rate (100 x higher) the results improved. Fig 6.8 shows the different learning curves for both learning rates. When choosing a higher learning rate the network also needs less training because the nodes are saturated faster. The loss curves are not as smooth as with a lower learning rate. The ROC auc stays around the same level. The results from above are obtained using the higher learning rate.

This network serves as the reference model. Every test with new variations in the dataset or network is compared to this model. This is because this model has quite good performance and is feasible to train within a reasonable amount of time. The best performing model is discussed in a later section.

**BiLSTM**

Just as adding a backward layer can improve the results of the vanilla RNN. A backward layer can be added to the LSTM network. A Bidirectional LSTM (BiLSTM) network is born which will also capture the long-term dependencies in the backward layer. The network performs better than the normal LSTM network, the results can be found in figure 6.9.

The better result could be caused by adding extra hidden nodes to the network which enlarges
6.2 Results

(a) ROC curve
(b) Precision Recall curve

Figure 6.9: Results of training a Bidirectional LSTM network

The learning capabilities of the network. Adding an extra layer means doubling the amount of nodes. However, a better result should then also be obtained when doubling the nodes to the normal LSTM network. Doing the test when training a normal LSTM network with double the amount of hidden cells (2000 instead of 1000) doesn’t give an equally good result as the BiLSTM network. It doesn’t even give a better result than training the normal LSTM with 1000 nodes. The backward layer thus has a real impact on the learning performance. It’s also clear that the network reached its maximum learning capabilities for the dataset and this type of networks since adding extra nodes will not result in better predictions.

BiLSTM network with extra hidden layer

As discussed in 5.2.3 an extra layer can be added which lets each time step contribute directly to the output node. Adding this extra layer means a big improvement in prediction result. The ROC auc rises to 0.891 and the PR auc rises to 0.629. Letting earlier time steps contribute to the output in a more direct way thus has a great impact on the prediction capabilities of the model. The resulting ROC and PR curves are shown in figure 6.10. The learning curve is shown in figure 6.11. This models learns best with a low learning rate in a large amount of epochs (300).
6.2 Results

(a) ROC curve  

(b) Precision Recall curve

Figure 6.10: Results of training an LSTM network with an extra hidden layer

Figure 6.11: Learning curve for a bidirectional LSTM network with extra hidden layer. Trained for 300 epochs with learning rate 0.000001.

6.2.4 Variations in Dataset

All SOFA scores

Until now, only the total SOFA score was taken into account. However, an infection will begin in one limb or part of the body and will spread throughout the body from there. This might be reflected in the SOFA scores, a partial SOFA score might have a high value first and the total
score will then evolve over time as the infection spreads throughout the body. To see if this reasoning makes sense, all partial scores are taken up in the dataset and a network is trained on this. The results are shown in figure 6.12. There is no improvement in ROC auc score but there is an improvement in PR auc.

![ROC curve](a), ![Precision-Recall curve](b)

Figure 6.12: Results of training an LSTM network on a dataset with all SOFA partial scores

### Altering Sampling Frequency

When varying the sampling frequency, a higher frequency is expected to give more accurate results because the sequence contains more values and the values gathered with lower frequency are also present in higher frequency sampled sequences. However, if the sampling frequency grows too large, it becomes more difficult for the network to capture long term dependencies which might be important for prediction. This pattern can be found in figure 6.13. The ROC auc is highest between 1 to 6 samples an hour and the PR auc is best when taking 3 samples per hour. Outside these regions the results are worse.

### Altering Sampled Period

All sequences in previous tests covered 24 hours of sampling. The influence of sampling for a longer or shorter period will be discussed here. Figure 6.14 shows the prediction scores for various sampling times. Overall there is no much variation. There is a small peak in precision recall on 24-30 hours of sampling. The PR auc at 24 hours sampling is here around 0.485 and
6.2 Results

Figure 6.13: Influence of sampling frequency on prediction

the ROC auc is 0.820. This is the same experiment as in figure 6.7. Training the same network can result in slightly different results each time because a different batch order is used each time and some initialisation values may be different. Keeping these training variations in mind it’s clear that altering the training period does not have a great influence on the results. The reason for this might be that the latest samples on one or zero hours before a positive sample are the most influential. Those are available in each of the tested periods which lead to the fact that each period yields a similar result.

Data Imputation Methods

All previous test were conducted using a sampled dataset where the most recent value in a time frame was used as sample value. However, some time frames might contain peaks that will be lost when using this type of sampling. To make full use of the available data it might be better to take the mean for all samples in a specified time frame. The results of a network trained on those mean samples is seen in figure 6.15. The network is the reference LSTM network with 1000 hidden nodes discussed in section 6.2.3. Using those mean samples improves the prediction of the network from an ROC auc of 0.837 to 0.847.
6.2 Results

Figure 6.14: Influence of sampled period on prediction

Using the means in a time frame improves the prediction result a bit. However, in some cases extreme values might be important such as a fever peak or drop or rise in white blood cells. Using average values would level those peaks out, which can be a loss of high value information. In concert with physicians a table is drafted which contains for which parameters which extreme values are be important. This information is stated in table 6.2. Training a network where samples consist of the appropriate extrema in their time frame results in a serious rise in prediction quality, the obtained ROC auc was 0.875 and the PR auc amounted 0.567 which is a
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Extremum</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMP</td>
<td>MAX</td>
</tr>
<tr>
<td>blood thrombocyte count</td>
<td>MIN</td>
</tr>
<tr>
<td>blood leukocyte count</td>
<td>MIN &amp; MAX</td>
</tr>
<tr>
<td>CRP-s</td>
<td>MAX</td>
</tr>
<tr>
<td>SOFA_TOTAL</td>
<td>MAX</td>
</tr>
<tr>
<td>HR(ecg)</td>
<td>MAX</td>
</tr>
<tr>
<td>RR(ecg)</td>
<td>/</td>
</tr>
<tr>
<td>INR protrombinetijd</td>
<td>MAX</td>
</tr>
<tr>
<td>SAPm</td>
<td>/</td>
</tr>
</tbody>
</table>

Table 6.2: Table containing which extrema are important for which parameter

reasonable amount more than training the reference network with the last samples in each time window. The curves are available in figure 6.16.

![ROC curve](Figure 6.16: Results of training an LSTM network on a dataset with extremum samples)

6.2.5 Early Prediction

The goal of this research is to develop a system that helps doctors recognize patients with a positive blood culture. The faster a patient is classified as a culture positive patient, the faster a treatment can be started and the higher the chances of survival are or the less likely it is the patient will evolve to a worse condition or even develop sepsis.
6.2 Results

So far all predictions were made on a patient sequence that covered samples up to the point when a first positive blood culture was taken or until the last available sequence in case the patient was culture negative. To check how well the model behaves for early prediction, data sequences with data up to 1, 2, 3 ... hours before the first positive samples can be entered in the network. Early prediction can be evaluated in two ways.

A first way is to train the network on sequences that contain data up to the point of the first sample (or last sample if culture negative). The model can then be evaluated for early prediction by consecutively testing that network first on sequences up to the positive sample point, next on sequences up to one hour, then up to two hours etc. Only one model has to be trained that can predict ‘if a patient will have or develop a positive blood culture’.

An other way to evaluate the early predicting is to train different networks on different training sequences of x hours in advance. Training in this manner, a network would be specialized in predicting ‘if a patient would have a positive test in x hours’. In practice, a system that uses this type of training would contain different networks and give a rating or prediction for different hours in advance. Although it should be noted that the blood culture sample time in the dataset is only a momentary shot. It is possible that a patient is already culture positive for a longer period so training a network for predicting x hours in advance is only a relative matter that is influenced by the dataset.

The results for the first method are shown in figure 6.17. Left is a model trained on data from 24 hours while right contains data spanning a 48 hour time frame. The results look very similar. Training on 24 hours yields a slightly better result, perhaps because the 48 hours variant learned some long-term dependencies which aren’t available in the test set which only contains a sampling time frame of 24 hours.

The results from the second method are available in figure 6.18. The results are again very similar to the one model approach. This curve is somewhat more bumpy, probably related to the influence of varying initialisation parameters and random training batches plays a role.

For both early prediction mechanisms the conclusion is that the closer to the positive test result;
6.2 Results

(a) Trained on 24 hours  
(b) Trained on 48 hours

Figure 6.17: Early prediction results when testing on one trained network. The left figure shows the results when the network was trained on a train set that contained samples from 24 hours sampling time, the right figure shows results from an LSTM network trained on samples spanning 48 hours of sampled time.

Figure 6.18: Early prediction results when separate networks are trained on training data that also contains data upto x hours before first positive culture sample.

the better the prediction becomes. For more than 10 hours before this test the ROC auc dives below 0.7. At this point the prediction becomes only poor according to [1]. An auc of 0.7 to 0.8 would be ‘fair’ and more than 0.8 is already good.
6.3 Overall Best Result

The previous sections each debated some new working angles or techniques used and compared them to the reference network 6.2.3. Some techniques made the predictions clearly better. A quick recap of the best models: First, the network type made a difference. The BiLSTM network with extra hidden layer came out as best performing model. The sampling frequency was also of interest, 3 samples an hour was best. A last point of improvement was using the prescribed min or max values for certain parameters. Combining those three methods should in theory give the best predictions. However, when doing tests with those 3 combined methods a better method than the ROC auc of 0.891 from the data imputation section was never achieved.

Lowering the sampling rate back to one per hour, using the minimum and maximum samples and using a BiLSTM with extra hidden layer on the other hand did result in the best prediction. A ROC and PR auc of 0.901 and 0.669 was attained. See the curves in figure 6.19. Why the model did not reach its optimum level when sampling at 3 times per hour is not clear. A supposition is that the sequences grew too long and it was more difficult for the model to train on them. An other reason might be that extrema values are less expressive because they are extrema for less samples. If the sample windows are small, the extremum is more likely to be the same as the last value. The difference between sampling with the last value becomes smaller.

![ROC curve](image_a)

![Precision Recall curve](image_b)

Figure 6.19: Results of training a BiLSTM network with extra hidden layer on a dataset with extremum samples
6.4 Comparison with Non-Recurrent Networks

How well the model predicts for several hours upfront is shown in figure 6.20. The model predicts best the closest to the blood culture test. The model still achieves an ROC auc of 0.8 at 8 hours upfront and decreases more to a ROC auc of ±0.725 at 24 hours upfront.

Figure 6.20: Early prediction results for the best achieving network trained on extremum values for the last 24 hours.

It is important to note that the early prediction capabilities of the BiLSTM network with extra hidden layer trained on the last samples in a time frame are even better. This can be caused by differences in initialisation parameters. Other tests with the extremum sample dataset resulted in a similar early prediction graph as the network trained on the last samples. Assume the network has similar capabilities with both datasets. Figure 6.21 shows a graph with ROC aucs from different networks for early prediction. It’s noticeable that training on the dataset with the most recent samples gives a less volatile curve than training on the dataset with extremum samples.

6.4 Comparison with Non-Recurrent Networks

As discussed in the literature study, several papers propose methods to predict sepsis or septic shock. Those papers don’t mention the use of recurrent neural networks for predicting, they only rely on methods not making use of temporal information such as decision trees, SVMs or
6.4 Comparison with Non-Recurrent Networks

Figure 6.21: Early prediction results for different networks and datasets. It shows an LSTM network and BiLSTM network with hidden layer (hl) trained on the dataset with most recent samples (last) and a BiLSTM network with and without extra hidden layer trained on the extremum (minmax) dataset.

A part of this research paper is to compare methods which do not rely on temporal information to the recurrent neural networks used here. A test set-up is made with an artificial neural network with 100 hidden nodes. This network is trained and tested on the same dataset as used for training RNNs but only uses values at the most recent sample time. When testing the early prediction capabilities, the most recent sample up to a specific time upfront was used.

This ANN network achieved a surprisingly good result with an ROC auc of 0.808 and a PR auc of 0.479. The curves for those results can be seen in figure 6.22. Which is not much lower than the standard LSTM network from figure 6.7 which achieved an ROC auc of 0.83. The PR auc for this ANN network is even higher than that of the LSTM network. In the literature they often add the mean values to the dataset, which is the mean over all previous samples for a variable per patient. A test was conducted with this mean value added to the dataset but the network did not produce a better result.
6.4 Comparison with Non-Recurrent Networks

![ROC curve](image1.png) ![Precision Recall curve](image2.png)

Figure 6.22: Results of training an ANN network on a dataset containing all last available samples

**Extremum samples**

Training the network on the same minimum or maximum samples as defined in table 6.2 gives a surprisingly high area under the ROC curve. The feed forward neural network approaches the result obtained by the LSTM on the extremum samples with an ROC auc of 0.869 and performs better in terms of precision recall with a score of 0.625. The used samples come from the same sampled sequence as used in the best achieving model. Except here only the most recent extremum values used as input for the model or the most recent ones upto x hours before the culture test when evaluating early prediction.

**Early Prediction**

Predicting the blood culture is most useful when it can be done upfront. It is thus interesting to see if the feed forward neural network is good at early predicting a blood culture test outcome. Again, two methods can be used: Several networks can be trained for several time steps upfront or one network can be trained for a certain time step and data from other time steps can be fed to this network. Both networks have been tested and give comparable results.

When training on a dataset with extremum samples, the early prediction results are similar to the ones of the BiLSTM, starting near a ROC auc of 0.869 and then lowering to 0.65 at
24 hours upfront. The early prediction capabilities are thus also a great deal lower than the ones from the BiLSTM network with extra hidden layer. Figure 6.23 shows a comparative graph where the early prediction curves for both the FNN and the BiLSTM network are shown, trained on the minimum-maximum samples.

![ROC PR evolution comparison](image)

Figure 6.23: Early prediction results for a FNN network compared to the best performing BiLSTM network with extra hidden layer. Both trained on extremum samples.

Although the ANN network predicts the blood outcome test reasonably well and behaves similar to some temporal models, it falls short against the BiLSTM network with extra hidden layer. Especially for very early predictions the difference becomes notable. The early prediction capabilities of the BiLSTM with extra hidden layer trained on the last samples present are even better.

### 6.5 Differentiating Parameters

LSTM networks are sort of black boxes with tuning buttons in them which update themselves to come to a reasonable prediction. It might be useful however for physicians to understand what is going on in them and why certain sequences result in a positive prediction and others in a negative one. Investigating how the LSTM network works inside is infeasible. It consists of a complex clew of mathematical operations and it is impossible to track values for a parameter
or value in a sequence let alone find combinations of sequences manually.

An option is to train a vanilla RNN with one hidden node and watch the different weights in this network. A one node vanilla RNN has light complexity and makes it possible to look what happens with input features in a transparent way. It therefore loses training ability but it’s a first step in discovering important parameters. Table 6.3 shows the weights for the trained RNN with one hidden node. This network was trained for 300 epochs and reached a surprisingly ROC auc of 0.790 and a PR auc of 0.406. This is pretty high for such a small network. From

| W in to hidden for 'TEMP'          | -0.41138036 |
| W in to hidden for 'blood thrombocyte count' | 0.02532065 |
| W in to hidden for 'blood leukocyte count'   | -0.13433482 |
| W in to hidden for 'CRP-s'           | -0.00854347 |
| W in to hidden for 'SOFA_TOTAL'      | 0.19286644  |
| W in to hidden for 'HR(ecg)'         | -0.20112183 |
| W in to hidden for 'RR(ecg)'         | -0.21087809 |
| W in to hidden for 'INR protrombinetijd' | 0.42040536 |
| W in to hidden for 'SAPm'            | 0.0471721   |
| bias in to hidden                   | -0.14081225 |
| W hidden to hidden                  | 1.13912083  |
| W out                               | -0.7964825  |

Table 6.3: Table representing weights from an RNN with one hidden node

the input to hidden weights can be derived which parameters have a potent influence on the node value, and thus indirectly to the network output. Because all input data is normalized, the parameters with the highest weight are the most influential. The most important are temperature, heart ratio and INR. Running this test a couple of times mostly yields higher weight values for those parameters. The sign in front of the weight determines in which sense the value has influence. Don’t forget to take the output weight into account, a negative output weight inverts all values. The temperature weight is negative but with a negative output weight this becomes positive (it is multiplied). In this network a higher temperature results in a higher output and thus more chance to have a positive prediction. For INR time the effect is inverse, this is strange because a patient in bad condition is expected to have a higher INR ratio. The
hidden to hidden weight is also quite high which means the temporal effects plays a great role. This network gives a nice overview which inputs are important but it can’t capture complex dependencies.
Chapter 7

Conclusion & Future Work

7.1 Conclusion

The presence of bacteria or fungi in the blood stream can emerge into a life threatening situation. Early detection is key in elevating the survival chances of blood culture positive patients. This master dissertation proposes advanced computational models to help physicians recognize and detect those patients whose blood stream is infected.

Many papers discussed methods on predicting sepsis. Sepsis is a medical condition which is highly correlated to having a positive blood culture. Those papers suggest several learning models to detect patients with sepsis but none of them use models that are able to capture temporal data which is often the type of data generated and monitored in intensive care units. Using models that handle temporal data, recurrent neural networks in particular meant a great added value to different applications such as phoneme recognition or sentiment analysis.

The intensive care unit of the Ghent University Hospital provided a dataset containing temporal information of 2177 admissions of which 229 patients were marked as blood culture positive and 1948 have only had negative blood culture tests. This dataset contained measured information from 10 different parameter groups ranging from heart rate and blood pressure to white blood cell count and blood coagulation ratios. To make the data from this dataset readable for the learning network, the data needs to be sampled. A sampling rate of 1 sample per hour made it able for the network to compare different admissions and make the best distinction between culture positive and culture negative patients.
A model is developed that is able to classify admissions based on 9 sequences of patient state defining parameters out of 29 parameters originally included in the dataset. The heart of this model contains an advanced form of a recurrent neural network, a bidirectional long short-term memory network or BiLSTM network with extra hidden layer. The model achieves a ROC auc of 0.901 when predicting sequences upto the time when a first blood culture test is performed. The early prediction capability of the network shrinks to a ROC auc of \pm 0.8 at 10 hours before a positive blood test and 0.76 at 24 hours before the positive test.

This model is compared to traditional non-temporal models used in sepsis detection literature and turns out to be more accurate in predicting at the moment of the blood culture test as well as in early prediction.

\section*{7.2 Future Work}

This dissertation explored interesting paths in the domain of blood culture prediction on the basis of computational models. However, this is not an unfinished business. Further expanding the dataset and making it more consistent will open the path for more reliable and possibly more accurate results.

New network architectures can be investigated and developed to improve the predictions even more. There is also a part of not-used data in the dataset containing medication information. Adding this to the prediction model could also mean an improvement. Doing more research on what happens inside the black box of neural networks can give deeper insight in how blood cultures develop and what the causes are. This will also give doctors something more tangible.

Improving the model is one operating point, the actual goal is to develop a real-time system that helps doctors to determine if a patient has a positive blood culture or not. This research paper only presented a computational model that could fit into one of the several building blocks that such a system would consist of. Gathering continuous data, thinking about architectural quality attributes such as availability, usability, reliability, etc. are important steps before an actual system can be developed.
Bibliography


[29] Matthew Morrell, Victoria J Fraser, and Marin H Kollef. Delaying the empiric treatment of candida bloodstream infection until positive blood culture results are obtained: a potential


