PREDICTING DETERIORATION OF PATIENTS IN THE INTENSIVE CARE UNIT

Hatem Bouabana

University of Tampere
Faculty of Natural Sciences
M.Sc. Thesis
Supervisor: Jaakko Peltonen

ABSTRACT

University of Tampere
Faculty of Natural Sciences
Master Degree Programme in Computational Big Data Analytics
HATEM BOUABANA: Predicting Deterioration of Patients in the Intensive Care Unit.
M.Sc. Thesis, 64 pages
April 2018

The massive influx of data in healthcare encouraged the building of data-driven machine learning models in order to improve medical diagnosis and assess patients health status. Thus, gaining knowledge and actionable insights from this complex, heterogeneous, poorly annotated and generally unstructured data remains a key challenge in transforming healthcare.

In this thesis, we focus on consecutive events in patients Electronic Health Records (EHR) data to predict organ failure in the ICU. We first investigate the ability of both logistic regression and random forest classifiers to achieve this task. Then, we compare these methods along with convolutional neural networks (CNN) for prediction of multiple organ failures: cardiovascular and pulmonary. Our predictions are done using a restricted set of parameters to enable "real-time" usage in the ICU. Predictions are made three hours in the future to support clinically actionable planning while having a 90 minutes window of parameters history. Our results suggest that the CNN has substantial additional predictive value for cardiovascular and pulmonary organ failures among critically ill patients.

Keywords: healthcare, machine learning, intensive care unit, logistic organ dysfunction system, convolutional neural networks.

PREFACE

This thesis has been carried out on the premises of GE Healthcare Finland Oy. First of all, I would like to thank my Engineering Manager Réne Coffeng for this awesome master's thesis opportunity. Moreover, I would like to extend my gratitude to my supervisors, Mika Särkelä for his enthusiasm and for the valuable feedback and advises along the accomplishment of my thesis and Jaakko Peltonen, for his insights and relevant comments regarding the thesis. In addition, I want to thank Hanna Viertiö-Oja for her endless passion in this project. I would like to emphasise my gratitude to Ville Pettilä for his involvement in this project, giving beneficial understanding regarding the medical perspective of this thesis.

Lastly, conducting this thesis work would not have been possible without the full support of my beloved family, relatives and my closest friends.

Helsinki, 18.04.2018

Hatem Bouabana

CONTENTS

1. INTRODUCTION

Artificial intelligence for clinical decision support has its origin in the knowledge-based Expert Systems (ES) that were introduced in the early 1970s. Early models such as the DENDRAL system [17] and MYCIN [51] turned out to be useful at that time and were commercialised at a large scale. However, this kind of learning based on experts has many limitations. In fact, the IF-THEN-ELSE knowledge in an ES is highly dependent on the human experts sharing knowledge in a way easily understandable by a knowledge-based system. In addition, the lack of human common sense needed in some decision making situations made the ES hard to apply in many applications [4].

Statistical machine learning methods such as logistic regression (LR), random forests (RF) and artificial neural networks (ANNs) have started to be used during the 90s as an alternative to ES to develop computer-aided systems in healthcare [4]. Then, deep learning has been used as a novel machine learning method that outperforms conventional methods[1] at the expense of reduced high-level interpretability [11, 36]. These techniques have been used to improve the objectivity and thereby the reliability of medical diagnosis.

Nowadays, treatments in the intensive care units (ICUs) require interpretation of a set of the patients vital signs and appraisal of many treatment alternatives. Therefore, even experienced physicians are prone to face difficulties in recognising the important and relevant deterioration patterns from patients vital signs data and reacting in a time-constrained, critical situation. In fact, the huge amount of information to be understood limits the quality and the efficiency of the intensive care. In addition, human factors such as inattentiveness, lack of expertise, and poor judgement make this task even harder. That is, integrating decision support systems in medical domains is a necessity. Furthermore, patients in the intensive care are often unconscious or incapable of alerting medical staff if their condition deteriorates. If a patient enters a critical state while being monitored, an alarm is raised to alert the medical staff. Such alarms, however, only allow doctors to react when the situation is already worse instead of helping them prevent these dangerous conditions in the

[1] LR, RF and classical ANNs

first place.

Analysing Electronic Health Records (EHR) data is a important step towards enhancing patients' diagnosis [12]. Clinical decision-support systems are usually based on machine learning models and can be an accurate, reliable and unbiased method to help clinicians with their everyday decision-making. Data-driven models can provide the advantage of being less biased than a physician when trained on a large data-set. For example, Pranav et al.[47] trained a convolutional neural network model that outperforms an average doctor on detecting cardiac arrhythmia.

Throughout this thesis, we focus on clinical decision-making. In this vein, we use Electronic Health Records (EHR) data that has been collected in Finnish ICUs to forecast, in real-time, patients' cardiovascular and pulmonary dysfunction during their ICU stay. The need for this kind of predictor has been raised by Prof. Ville Pettilä who has been the clinical domain expert in this project. This is important because misdiagnosed patients can deteriorate quite rapidly and an unnecessary intervention can be both harmful and expensive [16].

ICU scoring systems usually evaluate mortality risk as an outcome [5, 21]. Some of these scores such as the Simplified Acute Physiology Score (SAPS) use data collected during the first day of the ICU stay: They do not take into consideration the evolving clinical status of the patient [2, 41]. Other scores such as the Sequential Organ Failure Assessment (SOFA) and the Logistic Organ Dysfunction System (LODS) tend to evaluate the patient's organ deterioration, which is closely related to mortality. Calculating organ deterioration scores at regular intervals can give an approximation of the morbidity of the patient's organs.

We aim to learn feature representation from consecutive events in the patients' health records with the end of goal of predicting cardiovascular and pulmonary failures. We train a machine learning model that is able to capture complex trends and deterioration patterns in physiological signals over time and thus would be able to accurately predict deterioration onset. Continuous assessment of organ failure is particularly needed in the ICU, where the patient's health is quickly evolving and clinical intervention such as vasopressors administration can often happen.

This kind of forecasting is challenging because robust representations of human vital signs are complicated, and are most likely able to contain complex relationships and dependencies. Moreover, the evolving state of the patient requires the use of time-series data that is often varying-length, irregularly sampled and has missing values. In addition, clinical intervention on patients will often change the patient's outcome. These interventions can happen during the prediction time and cause

incorrect predictions.[54]

Getting access to such tools can help clinicians focus on deteriorating patients, especially when these tools can provide specific and accurate deterioration output. This will most likely improve the patient's outcome and reduce hospital costs.

We assess organ failure with the LODS score [34] and focus on cardiovascular and pulmonary dysfunctions for two reasons:

1. Reason of ICU admission: Suspicion of cardiovascular or pulmonary failures are the major causes of ICU admissions [43]
2. Deterioration type: Unlike cardiovascular and pulmonary organs, renal, haematologic, and hepatic dysfunctions require lab tests which are not quickly available after the patient's admission in the ICU.

Thus, we first train a logistic regression and random forest models for organ-failure prediction. These two statistical learning methods have shown to be successful in many healthcare-related applications [56, 62]. Then, we take advantage of modern machine learning advances by using Convolutional Neural Networks (CNNs) to capture deterioration trends as well as complex relationships in the longitudinal patients data. We focus on a restricted set of vital signs (input parameters) that are most likely to be available in any ICU from the patient's admission time onward, in order to start the prediction task as early as possible. CNNs have proven to be successful in classifying complex time-series data [57]. They have also achieved great results in many applications such as image recognition, speech recognition, and natural language processing. We compare the performance of our CNN with the previously cited machine learning methods in forecasting onset and continuity of organ failure.

The rest of this thesis is structured as follows:

- Chapter 2 is the background chapter, it describes the hospital and ICU environments together with the most prominent ICU scoring systems. It also explains the machine learning basics and the techniques that have been used in this thesis.
- Chapter 3 is an overview of the previous work that is related to this topic.
- Chapter 4 concerns the research methodology and materials used in this work.
- Chapter 5 compares the results that have been obtained using the previously

cited machine learning techniques and provides a further analysis of the CNN model.

- Chapter 6 provides a discussion about the findings in this work.
- Chapter 7 provides a conclusion of this thesis and discusses potential directions of future work.

2. BACKGROUND

2.1 The intensive care unit and patients' deterioration

This section will introduce concepts related to the hospital intensive care environment and patients deterioration.

2.1.1 The intensive care unit and its challenges

The intensive care unit is a special unit in the hospital that contains patients with serious injuries or illnesses. These patients require a high attention from the medical staff because they are often unconscious or unable to alert the medical staff if their condition deteriorates. For these reasons, these patients are monitored by sensors that are continuously measuring multiple parameters. ICUs usually contain a higher number of staff compared to other hospital units such as the pediatric general care unit and neonatal unit and require more medical resources. ICU patients are most commonly treated for the following complications:

- Adult Respiratory Distress Syndrome (ARDS).
- Trauma.
- Multiple Organ Failure (MOF).
- Sepsis.

ICU units contain patients assumed to be recoverable and in need of supervision or specialised techniques by highly skilled personnel. These units include many specialised equipment, among them:

- Real-time monitoring of cardiovascular parameters.
- Real-time monitoring of the respiratory parameters.

- Monitoring of the renal parameters.

This equipment is not likely to be available in any other hospital unit except the operating room.

Matalkala et al.[37] specified the need to construct large ICU units. In fact, the growing size of the population and the longer life expectancy among patients made small ICUs overpopulated. The authors have also shown that highly skilled staff shortage in the ICU constitutes, nowadays, a real issue. This problem can be tackled with clinical support systems which are designed to help clinicians and nurses focus on deteriorating patients.

2.1.2 Patients' deterioration

Patients' deterioration is a real issue in today's hospital care units. According to prof. Ville Pettilä, most of the patients in the Intensive Care Unit will experience a severe health deterioration at least once during their ICU stay. If this deterioration is detected early on, patients will most likely recover and early intervention will prevent any eventual sequelae. However, delayed intervention constitutes a huge risk to the patient. Delayed intervention is mostly caused by unexpected clinical deterioration and is related to high mortality rate [55].

Multiple Organ Dysfunction (MOD) is considered as being the most common pattern of symptoms before death in hospitals (43% of the dead patients experienced a severe MOD) [59]. It is the process of sequential failure of organs (cardiovascular, neurologic, renal, pulmonary, hematologic and hepatic). It alters the function of the affected organs from a light, reversible degree of dysfunction to an irreversible organ failure.

Taenzer et al.[55] specified that early signs and symptoms of deterioration start to manifest at least 8 hours before the cardiopulmonary arrest. According to the United States Joint Commission's 2009 National Patient Safety Goals, an increasing number of warning signs precede any critical event (6 to 8 hours before) [49]. It has been also reported that nearly 70% of the patients who encounter a circulatory arrest have respiratory problems at least 8 hours prior to the episode [50]. However, detecting early deterioration by nurses and physicians is quite challenging due to the number of ICU patients and the shortage of in-hospital nurses/clinicians.

In order to prevent delayed interventions and to assess patients health, statistical scores have been established. These scores can be grouped into two categories:

mortality scores and multiple organ dysfunction scores.

2.1.3 Mortality scores

Multiple scoring systems have been designed to predict mortality. For example, the Acute Physiology And Chronic Health Evaluation (APACHE) and the Simplified Acute Physiology Score (SAPS) are severity ranking scores that estimate risk of death in the 24 hours that follow the admission time.

2.1.4 Multiple organ dysfunction scores

The Logistic Organ Dysfunction System (LODS) and the Sequential Organ Failure Assessment (SOFA) scores have been primarily elaborated to assess the gravity of Multiple Organ Dysfunction (MOD) and not to predict mortality even though these two are highly related. According to a study conducted by V. Pettilä, these scores have also demonstrated a rather good hospital mortality prediction compared to APACHE. [46]

Logistic Organ Dysfunction System score

The Logistic Organ Dysfunction System (LODS) score is a system that measures the patient's vital status during the ICU stay. This score is calculated for each of the following organ systems: neurologic, cardiovascular, renal, pulmonary, haematologic and hepatic. Scores from one to five are assigned to the levels of severity of each organ system and the overall LODS is the sum of these system subscores and ranges.[34]

	Severity level			
Organ system	0	1	2	3
	LODS points			
Neurologic	0	1	3	5
Cardiovascular	0	1	3	5
Renal	0	1	3	5
Pulmonary	0	1	3	X
Hematologic	0	1	3	X
Hepatic	0	1	X	X

Table 2.1 *The severity of each organ according to the LODS scoring system (from Le Gall et al.[34])*

According to table 2.1, each organ has a specific weight in the LODS scoring system. For example, the neurologic, cardiovascular and renal organ systems have a bigger impact on the overall LODS score than the other organs [34]. In fact, at a high severity, these organs represent a higher risk on the patient's health. The sum of these points gives the overall LODS score.

Organ System Measures		LOD Points						
		Increasing severity/ Decreasing values			Organ dysfunction free	Increasing severity/ Increasing values		
		5	**3**	**1**	**0**	**1**	**3**	**5**
Neurologic	Glasgow Coma Score	3-5	6-8	9-13	14-15	...	...	...
Cardiovascular	Heart rate, beats/min	<30	...	...	30-139	>139	...	...
		or			and	or		
	Systolic blood pressure, mm Hg	<40	40-69	70-89	90-239	240-269	>269	...
Renal	Serum urea, mmol/L (g/L)	...	...	...	<6 (<0.36)	6-9.9 (0.36-0.59)	10-19.9 (0.60-1.19)	>=20 (>=1.20)
	or							
	Serum urea nitrogen, mmol/ L (mg/dL)	...	...	...	<6 (<17)	6-9.9 (17-<28)	10-19.9 (28-<56)	>=20 (>=56)
					and	or	or	
	Creatinine, μmol/L (ŋg/dL)	...	...	...	<106 (<1.20)	106-140 (1.20-1.59)	>=141 (>=1.60)	...
					and		or	
	Urine output, L/d	<0.5	0.5-0.74	...	0.75-9.99	...	>=10	...
Pulmonary	PaO_2(mm Hg)/FiO_2 on MV or CPAP	...	<150	>=150	No ventilation,	...	...	...
					No CPAP,			
	(PaO_2 [kPa] / FiO_2)	...	(<19.9)	(>=19.9)	No IPAP	...	...	...
Hematologic	White blood cell count, $\times 10^9$/L	...	<1.0	1.0-2.4	2.5-49.9	>=50.0	...	...
				or	and			
	Platelets, $\times 10^9$/L	...	...	<50	>=50	...	...	...
Hepatic	Bilirubin, μmol/L (mg/dL)	...	...	...	<34.2 (<2.0)	>=34.2 (>=2.0)	...	...
					and	or		
	Prothrombin time, (% of standard)	...	...	(<25%)	<=3 (>=25%)	>3	...	...

Table 2.2 *Scoring of the Logistic Organ Dysfunction System. CPAP: Continuous Positive Airway Pressure, FiO_2: Fraction of inspired Oxygen, IPAP: Intermittent Positive Airway Pressure, MV: Mechanical Ventilation, PaO_2: Partial Pressure of Oxygen (from Le Gall et al.[34]).*

Table 2.2 shows the scoring criteria for each organ. To calculate the LODS score, each organ is associated with its highest score related to a certain variable. For example, in the cardiovascular system, if the systolic blood pressure is <40 (5 LODS points) but the heart rate is 70 (1 LODS points) the cardiovascular LODS score will be the highest (5 LODS points). Then, to calculate the overall LODS score, a sum of the organ LODS sub-components is calculated.

The LODS score is a system that was elaborated with a total of 14 745 patients from 137 ICUs from 12 different countries [34]. The discrete severity levels used in the LODS score, and the way they are computed from discretized (thresholded) levels of different parameters, were developed as part of a logistic regression prediction of whether a patient would die or not. A brief description of how this was done is provided below for reference.

Le Gall et al.[34] first identified thresholds of variable ranges related to mortality. In order to do that, they plotted the continuous measurements of the input variables shown in Table 2.2 against the vital status of the patient at hospital discharge. The LOcally WEighted Scatter-plot Smoothing (LOWESS) function was used to suggest these cut points. For example, if the smoothed plot showed an increasing association between low heart rate and mortality when the heart rate is lower than 50, then 50 could be chosen as a cut point for a range of severity. Then, dummy variables were created for each category (newly created range) outside normal category (organ dysfunction free). These dummy variables would take the value of 1 if the value of the parameter is within this category or 0 if not. The parameters were then passed as input to a multiple logistic regression model. This created new coefficients (β) that represented the weight of each input parameter. The β were then grouped into levels of increased severity. These levels were based on the range of the β coefficients.

In order to define the severity levels, the authors grouped comparable levels of severity for each group of parameters defining an organ dysfunction (for example, heart rate and systolic blood pressure). Then, as they did with the continuous measurements of parameters, they defined dummy variables for each level of organ severity. For each organ, they defined a patient as being in the dysfunction level only if the values were within the limits for each parameter defining the organ dysfunction. These dummy variables now represent the severity of organ dysfunction. The β associated with each parameter for each dummy variable were then multiplied by a factor of 10 and rounded to obtain a number. Then, the LODS score was computed as the sum of the sub-scores of the system's organs. This was used as a variable in logistic regression to assess the performance of the model on patients who died and those who did not die.[34]

Sequential Organ Failure Assessment score

Sequential Organ Failure Assessment score (SOFA) is a scoring system used to assess patients health during their ICU stay. The score is calculated organ-wise then, a sum is used to measure the overall score.[58]

SOFA score is based on six physiological systems:

- Respiratory system.
- Cardiovascular system.
- Hepatic system.
- Coagulation system.
- Renal system.
- Neurological system.

The cardiovascular SOFA score has been used this thesis both to discuss related work ad as an input variable of the developed system.

It is important to specify that the cutoff of each parameter in calculating the score has been done by Vincent et al.[58] using their own medical knowledge of deterioration. The highest score in each organ is taken into account to form the final SOFA which is the sum of organ-based SOFA.

The SOFA score has been designed to describe a sequence of complications and not to predict the outcome. SOFA helps healthcare specialists to assess deterioration and evaluate morbidity of the patients.[58]

2.2 Machine learning

Samuel [48] described machine learning as the capability of machines to learn without specifically programming them. This part of Artificial Intelligence (AI) studies the design, analysis, development and implementation of techniques. Thus, enabling the machine to evolve through a systematic and iterative process to perform tasks that are difficult or impossible to accomplish by conventional algorithmic means. The use of machine learning can cover various distinct tasks such as stock market forecasting, pattern discovery or image recognition.

This chapter will introduce the basic machine learning concepts and methods used in this thesis.

2.2.1 Basic concepts

As defined by Mitchell [39], "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E".

Thus, it is possible to imagine a variety of experiments E, tasks T and performance measures P. These entities will be described in the following subsection.

The tasks

According to Goodfellow et al.[24], machine learning tasks are usually described as the task of data processing by a learning algorithm. There are many different types of machine learning tasks, here we focus specifically on classification tasks. In a classification task, the program has to label or classify each data sample. The algorithm is generally represented as a mapping function $f : R^n \rightarrow 1, .., k$ where k is a finite number of potential classes. When $y = f(x)$, the model classifies the input data x (input) into a class represented by an integer y (output).

The performance measure

Assessing the performance of a machine learning model requires using a performance measure P. This measure mainly depends on the task T and might differ from one task to another. Accuracy is the most widely used measure in classification tasks. It is represented by the proportion of correctly classified samples.

To estimate the effectiveness of a machine learning model, this model has to be tested on data that it has not seen before. These performance measures have to be used on an independent test set.

In binary classification, an error matrix known as a confusion matrix can be derived from the model's evaluation. The confusion matrix is a count of four different outcomes depending on whether the predicted value was positive or negative, and depending on whether the true value was positive or negative, as shown in Table 2.3.

	Prediction: Negatives	Prediction: Positives
True class: Negatives (N)	**True Negatives (TN)**	**False Positives (FP)**
True class: Positives (P)	**False Negatives (FN)**	**True Positives (TP)**

Table 2.3 *Confusion matrix*

Then, the following performance measures can be used:

Performance measure	**Definition**
True Positive Rate (TPR, sensitivity)	$\frac{TP}{P}$
True Negative Rate (TNR, specificity)	$\frac{TN}{N}$
False Positives Rate (FPR)	$\frac{FP}{FP+TN}$
False Negative Rate (FPR)	$\frac{FN}{FN+TP}$
Positive Predictive Value (PPV)	$\frac{TP}{(TP+FP)}$
Area Under ROC curve (AUROC)	$\int_{\infty}^{-\infty} TPR(T)FPR'(T)\,dT$

Table 2.4 *Performance measures in a binary classification*

Here, T is the detection threshold (probability threshold). A sample x is classified positive if $Pr(x) > T$, and negative otherwise.

Loss functions can also be used to measure the performance of machine learning models, among them:

Loss function	Definition
Mean Squared Error (MSE)	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
Cross-entropy	$-\sum_x p(x)\,\log q(x)$

Table 2.5 *Loss functions*

Where

- *p(x)* is the true class probability distribution.
- *q(x)* is the predictions' probability distribution.

According to the type of application, methods such as class weighting can be applied to the loss function in order to penalise in a different way the class-wise errors. For example, in in situations where proportions of different classes are unbalanced in the training data, class weighting known as "balanced weighting" is usually used. It means assigning each sample of the training data an importance weight in the cost function of the learning model, based on the class of the sample, so that samples coming from small classes are given increased importance when optimising the learning model. The weight is computed as follows:

$$\text{Class weight} = \frac{count(samples)}{count(classes) \times bincount(y)} \tag{2.1}$$

Where

- count(samples) represents the number of samples in the dataset.
- count(classes) represents the number of classes in the dataset.
- bincount(y) is the binning count (number of occurrences of each class in the dataset).

The balanced weighting heuristic was inspired by the work of King et al.[31] and has been used in this thesis.

Experience, *E*

According to Goodfellow et al.[24], the experience is defined by the scenario or the steps that the program has to follow up with the training. This scenario is most

commonly known as the training algorithm. Training algorithms can be divided into three categories: Unsupervised, supervised and semi-supervised. This work focuses on the supervised learning as it is the training method that is used in this thesis.

Supervised learning generates a model from a dataset containing features where each sample is associated with a label. For example, the "Iris" dataset contains measured characteristics of different individual Iris plants from several Iris species [18]. A supervised machine learning program can analyse this database and learn to classify Iris plants into different species given. Thus, the algorithm aims to learn the association function $y = f(x)$ between vectors x and labels y in a way that yields successful classification according to some measure such as minimal classification error.

According to Bengio[3], machine learning methods are usually trained using a single division of data into a training, a validation and a test set instead of a K-fold Cross-Validation (CV). K-fold CV becomes useful when in possession of a tiny dataset (hundreds of examples) which is not large enough to tune a complex model. The training set is the sample of data (for example around 70%) used to fit the model. The validation set is the sample of data (for example around 15%) that evaluates the model's fit on the training dataset during the iterative training process. However, this evaluation becomes biased over time as the user tune the model's hyper-parameters on this set. Therefore, the test set (for example around 15%) provides an unbiased evaluation of the model. In some cases the overall data set is small (hundreds of examples) and K-fold Cross-Validation (CV) becomes useful in these situations. K-fold CV divides the data into K equal partitions. Then, every single partition is taken as a validation data to test the performance of the model, after training it on the remaining data partitions. In K-Fold CV, this task is reiterated K times where each development partition has to be used only once. The K estimations can be then combined into an overall estimation of the model. Using this method provides more training examples than the single division method while using at each fold, exactly one validation set.

Under-fitting and over-fitting

During the learning phase, the algorithm might experience some issues. The most important ones are:

- Under-fitting: happens when the model cannot minimise the training error enough. In other words, the algorithm is not able to induce a model that fits the training data. This is mostly due to a poor dataset or a model that is not complex enough to accurately represent the underlying dependency between inputs and outputs in the data set, e.g. a nonlinear relationship would not be accurately represented by a linear predictor.
- Over-fitting: over-fitting occurs when the model is too complex compared to the data or the task's complexity. Therefore, the model fits the data in too much detail of individual data samples, and noise or random fluctuations in those samples are learnt as concepts by the model. This causes a huge difference between the training error and the test error because of the examples that the model has over-learnt and that are, most likely, not available in the test set. Therefore, the learned parameters (weights) did not allow a good generalisation.

The following sections will describe the machine learning methods that have been used in this work.

2.2.2 Logistic regression

Logistic regression is a classifier that outputs a probability that a given input belongs to a certain class out of two possibilities, here denoted as -1 and 1. It uses the following formula:

$$p(c = -1|\mathbf{x}) = \frac{1}{1 + exp[-(\mathbf{w}^T\mathbf{x} + b)]} \tag{2.2}$$

Thus,

$$p(c = 1|\mathbf{x}) = 1 - p(c = -1|\mathbf{x})) \tag{2.3}$$

Where

- $\mathbf{x} \in \mathbb{R}^n$ is the input vector.
- $\mathbf{w} \in \mathbb{R}^n$ are the learned weights.
- b is the bias.

The model maps the projection $\mathbf{w}^T\mathbf{x} + b$ through the *sigmoid(x)* function. This limits the output to [0, 1].

The logistic regression classifier is often trained by maximum likelihood. The likelihood of samples $\mathbf{x} = [x_0, x_1, ..., x_{n-1}]$ to achieve class labels $\mathbf{y} = [y_0, y_1, ..., y_{n-1}]$ according to a logistic regression model with parameters theta can be written as:

$$L(\theta \mid \mathbf{x}) = p(\mathbf{y} \mid \mathbf{x}; \theta) = \prod_{i=0}^{n-1} p(y_n \mid x_n; \theta) \tag{2.4}$$

Taking the log-likelihood for simplicity:

$$ln(p(\mathbf{x} \mid \theta)) = \sum_{i=0}^{n-1} ln(p(y_n \mid x_n; \theta)) \tag{2.5}$$

Thus, minimising the following logistic loss function is equivalent to maximising the log likelihood.

$$\textit{Log-loss} = \sum_{i=0}^{n-1} ln(1 + exp(-y_n \mathbf{w}^T \mathbf{x}_n)). \tag{2.6}$$

A penalty term is usually added in favour of small coefficients in $\mathbf{w}$.

$$\textit{Penalised log-loss} = \textit{Log-loss} + \textit{Penalty} \tag{2.7}$$

The penalty contains a coefficient λ as well as a regularisation term L_q norm (for q $>$ 0) defined as:

$$Penalty \ = \lambda \ ||\mathbf{w}||_q \tag{2.8}$$

With

$$||\mathbf{w}||_q = (\sum_{i=0}^{p-1} |w_i|^q)^{-q} \tag{2.9}$$

L_1 regularisation is also known as LASSO[1]. This regularisation promotes sparse weight vectors that discard features (feature selection). Whereas L_2 regularisation (Tikhonov regularisation) adds the "absolute value of magnitude" of coefficient and promotes smaller weights when λ is large [20].

[1]Least Absolute Shrinkage and Selection Operator

Therefore, the final equation is:

$$\textit{Penalised log-loss} \ = \sum_{i=0}^{n-1} ln(1 + exp(-y_n \mathbf{w}^T \mathbf{x}_n)) + \lambda \ ||\mathbf{w}||_q \tag{2.10}$$

2.2.3 Random forest classifier

Random forest (RF) classifier is an ensemble machine learning method. It consist of a set of decision trees trained in parallel with a different initialisation. A decision tree is a diagram that combines individual attributes into a decision reducing the Gini impurity or another cost function during the training process.[61]

The Gini impurity it is computed by the following formula:

$$I_G(p) = \sum_{i=1}^{J} p_i(1 - p_i) = \sum_{i=1}^{J} (p_i - {p_i}^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} {p_i}^2 = 1 - \sum_{i=1}^{J} {p_i}^2 \tag{2.11}$$

Where J is the number of classes in the dataset, p_i is the fraction of items labelled with class i in the dataset and $1 - p_i$ is the probability of an erroneous prediction of an item.[61]

Random forest classifier benefit from the robustness and interpretability of the decision trees and tackles their disadvantages such as over-fitting by training a multitude of decision trees.

2.2.4 Artificial neural networks

Artificial Neural Networks (ANNs) is a well-known machine learning technique that can be used in both supervised and unsupervised learning. Neural networks are represented by computational graphs where the basic unit is a *neuron*. In a network that has n layers, the first layer is called input layer, the n^{th} layer is the output layer and the $n - 2$ intermediate layers are called hidden layers.

Neural networks are typically trained (optimised) by Gradient Descent (GD). In this process (GD), the weights of the network are updated in order to minimise the loss function and to find the local optimum by calculating the gradient of the error and then updating the network's weights through the back-propagation algorithm.

Feed-forward back-propagation is the algorithm that computes the gradient of the error in the networks and adjusts the weights according to it. The term feed-forward

specifies that the data will flow through the network from the input layer to the output layer whereas back-propagation is the process that propagates the error in the network from the output layer. Every neuron will compute the gradient of the error according to its weights and adjust them locally then propagate the gradient and error backward to the previous levels. Each unit on the i^{th} layer will adjust its connection weights with the neurons of layer i-1 according to their propagated contributions to the overall error.

Feed-forward step

In the feed forward step, each neuron in the network computes an inner product between its input vector P and a weights vector W and adds a bias b (equation 2.12). The result of this last operation passes through the non-linear activation function φ (such as sigmoid(x) or tanh(x)).

$$f(x) = \varphi(b + \sum_{i=1}^{n} p_i w_i) \tag{2.12}$$

Here, the input vector P can represent the feature value vector X of an input sample if the neuron belongs to the input layer, or the outputs from predecessor hidden layer if the neuron belongs to a hidden layer.

In a one hidden layer network, the output function f of this model would be:

$$f(x) = \varphi(b_2 + W_2^T \times (\varphi(b_1 + W_1^T \times X))) \tag{2.13}$$

Where W_1, W_2 represents the weights' matrices, X the input vector and b_1, b_2 represents the biases of the hidden and output layers respectively.

Back-propagation of the error

In order to get as close as possible to the targeted function, the neural network needs to adjust its learning parameters (W_1, W_2) at each iteration using the back-propagation of the error algorithm [25]. Thus, a cost function is used to quantify the performance of the network by calculating the error using the targets and the model's corresponding outputs.

For example, let's take the neural network represented in Figure 2.1.

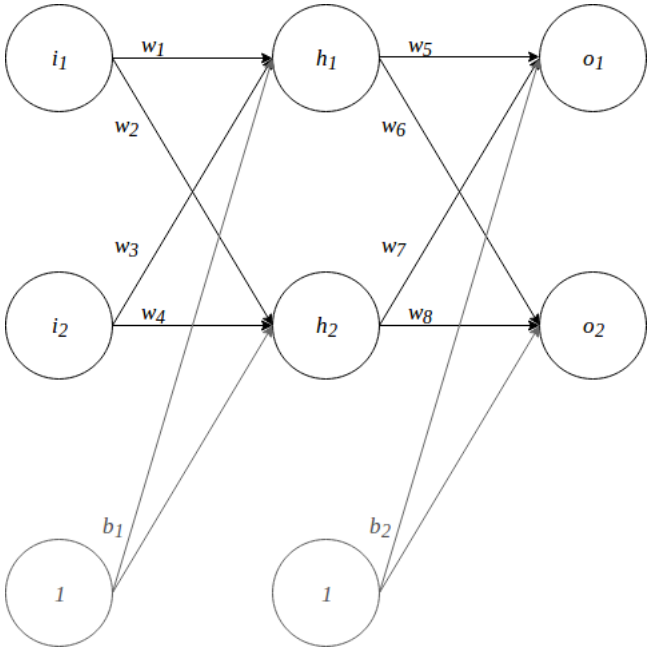

Figure 2.1 *Artificial neural network*

i_1 and i_2 represents the input neurons (input layer), h_1 and h_2 represents the hidden neurons and o the output neurons. The cost function is defined as the mean squared error function (Table 2.5):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(target_i - output_i)^2 \tag{2.14}$$

During the forward pass, the output at the hidden layer (h_1 and h_2) is calculated first then comes the output layer (o_1 and o_2). As explained earlier, all of these units use the same function to calculate their output (equation 2.12).

Once the output calculated, the error is estimated using the cost function.

Then comes the backward pass, where the weights of the ANN are updated.

Output layer

For instance, the estimate of the error with respect to w_5 is:

$$\Delta_{w_5} = \frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \cdot \frac{\partial out_{o1}}{\partial net_{o1}} \cdot \frac{\partial net_{o1}}{\partial w_5} \tag{2.15}$$

Where

$$E_{total} = \frac{1}{2}((target_{o1} - out_{o1})^2 + (target_{o2} - out_{o2})^2) \tag{2.16}$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 \cdot \frac{1}{2}(target_{o1} - out_{o1})^{2-1} - 1 + 0 \tag{2.17}$$

and

$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}} \tag{2.18}$$

$$net_{o1} = w_5 \cdot out_{h1} + w_6 \cdot out_{h2} + b_2 \tag{2.19}$$

and

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) \tag{2.20}$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 \cdot out_{h1} \cdot w_5^{(1-1)} + 0 + 0 \tag{2.21}$$

Then, w_5 is updated:

$$w_5 = w_5 - \eta \cdot \Delta_{w_5} \tag{2.22}$$

Where η is the learning rate. The learning rate can be a constant (relatively small) or a function of time or iterations. It regulates the learning process and prevents the learning algorithm from making huge (or too small) changes to the weights after one iteration.

Hidden layer

For the weights of the hidden layer (w_1 to w_4 of Figure 2.1) the process is slightly different. The output of each neuron in the hidden layer contributes to the neurons on the output layer. In other words, out_{h1} affects both out_{o1} and out_{o2}. Thus, $\frac{\partial E_{total}}{\partial out_{h1}}$

needs to take into consideration both outputs (out_{o1} and out_{o2}).

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} \tag{2.23}$$

Then, the gradient of the error with respect to w_1 is calculated using the following equation:

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} \cdot \frac{\partial out_{h1}}{\partial net_{h1}} \cdot \frac{\partial net_{h1}}{\partial w_1} \tag{2.24}$$

Where the first term on the right-hand side is computed by 2.23 and the second and third terms are computed as in 2.20 and 2.21. Then, w_1 is updated as in 2.22.

2.2.5 Deep learning and convolutional neural networks

Deep learning is the task of using a multilayer feed forward neural network with a set of learning techniques that allows producing a model with a high abstraction level. These techniques have led to significant progress in the areas of signal analysis such as voice or face recognition as well as natural language processing.

Since 2006, this progress has attracted significant investment from a diverse community (private, academic and public). As recently as 2015, the program AlphaGo developed by the team "Google DeepMind" beat 5 to 0 the European champion Fan Hui at the game of "Go". In March 2016, the same program beat world champion Lee Sedol 4 games to 1 [13].

Due to low computational power, the concept of deep learning only came to fruition towards the end of the first decade of this century thanks to the exponential evolution of machine processing power and the use of graphical processing units (GPU) [40].

One of the major drawbacks with conventional machine learning methods other than deep learning is the time spent on engineering the features. In fact, machine learning experts often have to study the problem thoroughly to determine the relevant characteristics. A simple example would be to acknowledge that blood sugar is a relevant feature in the detection of diabetes. In fact, when using machine learning, researchers would like the machine to be "intelligent" enough to capture by itself relevant features and characteristics of the data because human comprehension is different from machine understanding.

The biggest contribution of deep learning methods is to spend less time on engineering the features. The deep architecture will extract relevant features from the data by itself. It is even suggested that the importance of some features discovered

by the network cannot be explained with a domain expert's logic.[24] Some of the features extracted by the deep neural network be can be used in other tasks as done by Collobert and Westone in 2008 [10]. The authors developed a unified architecture for performing multiple natural language processing tasks such as named entity recognition and morpho-syntactical labelling. Therefore, it is assumed that these tasks require a set of similar characteristics.

Another advantage of deep architectures is that they tend to produce a better generalisation on unseen data than non-deep-networks thanks to their high level of non-linearity. In fact, using non-linear functions such as the Rectified Linear Unit (ReLU) between layers ensure that the model is highly expressive. In *Learning deep architectures for AI*, Bengio [24] wrote:

> *" functions that can be represented by a depth 'k' architecture might require an exponential number of computational elements to be represented by a depth $k-1$ architecture. Since the number of computational elements one can afford depends on the number of training examples available to tune or select them, the consequences are not just computational but also statistical: poor generalisation may be expected when using an insufficiently deep architecture for representing some functions "*

The reader might be led to believe that deep learning is the solution to all problems related to machine learning. This is not yet the case. Indeed, deep architectures require much more complex parameterisation than conventional neural networks. There is still no guide on how to parameterize a deep neural network. Each set of parameters and architectures are suited for different tasks. Therefore, the use of deep architecture requires the tuning of a large number of empirical parameters and a huge database. [53]

Convolutional Neural Netowrks (CNN)

The idea behind CNN was inspired by the work of Hubel and Weisel on the monkey's visual cortex [27]. A convolutional neural network also called ConvNet or CNN aims to simulate a biological neural network. One of the most commonly used ConvNet architecture was introduced in *LeNet* [35]. It stacks multiple layers of convolution and sub-sampling layers followed by a fully connected multilayer perceptron network.

The following techniques are usually used in CNNs but can be applied to multiple deep learning architectures.

Convolution

The convolution is an operation applied to two functions to output a third function. In the context of CNNs, a convolution is usually applied as a filter on a signal.

$$S(t) = (I * h)(t) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(t+m, n)h(m, n) \tag{2.25}$$

Where S is the new signal, I is the source signal, h is the convolutional filter, M is the size of the time axis and N is the number of the signal sources.

The following figure is an example of a one-dimensional convolution.

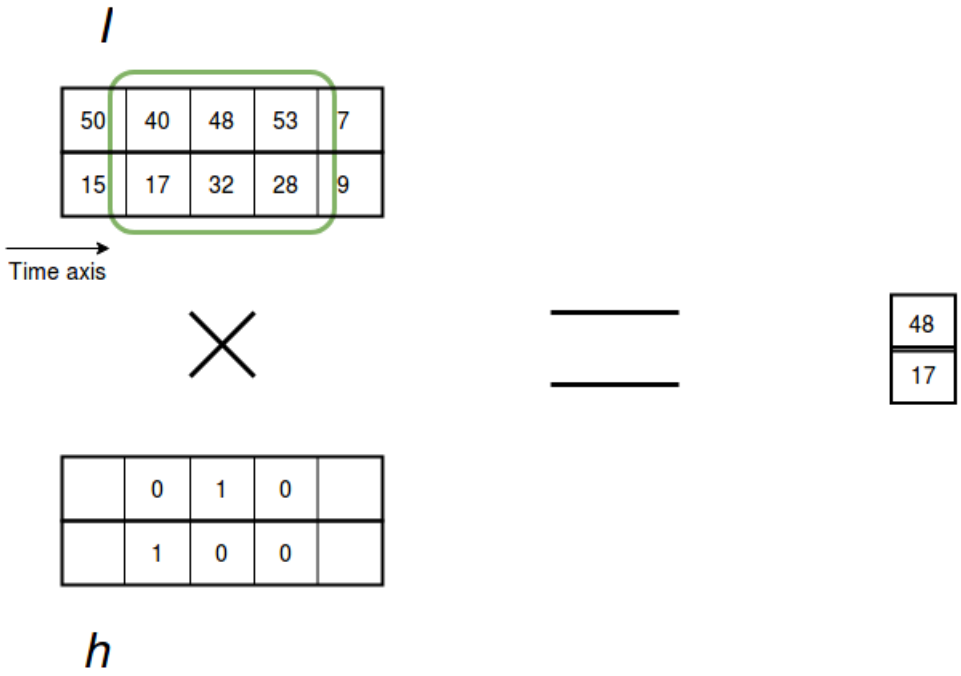

Figure 2.2 *An example of a convolution operation.*

In this example, the size of the convolutional filter has been set to 3, The stacked convolutional layers in a neural network simply apply this same kind of convolution while the convolutional filter h contains the weights of the layer.

Padding

Applying a convolution on a set of data (ex. time-series) will reduce its dimensionality. In fact, the components situated in the borders of the time-series cannot be convolved which causes a sensitive information loss. Padding is usually used to overcome this issue. One of the most common padding techniques is the zero-padding. It consists of adding an outer component to the image filled with zero before applying the convolution.

Sub-sampling

The sub-sampling is a technique that aims to reduce the dimensionality of a signal. It has been used in many applications such as image and audio compression.

The sub-sampling method that is usually used in CNNs is better known as max-pooling. It has been introduced in LeNet by LeCun [35]. This technique simply splits the signal into small, non-overlapping fragments (the larger the fragments, the larger the reduction). Then, it outputs the maximum value in each fragment. These values will form the new signal. This allows, among other things, to steadily reduce the dimensions of the data consequently decreasing the number of input parameters and computation in the network to control over-fitting.
Figure 2.3 represents a max-pooling operation.

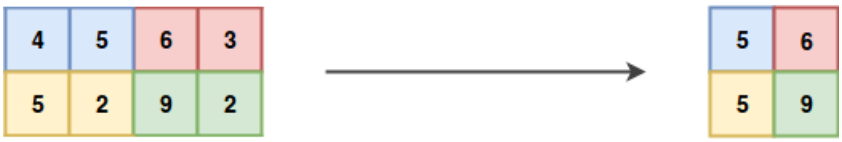

Figure 2.3 *Max-pooling with a two-sized kernel example.[30]*

Batch normalisation

Batch (set of input samples) normalisation is a commonly used technique that usually comes after a convolutional layer. It aims to reduce the shift of the hidden units (also called internal covariate shift). This shift is due to the different distributions of data between the layers of the deep neural network during the training

process. Therefore, batch normalisation increases the stability of the neural network by subtracting the batch mean and dividing by the batch standard deviation at each hidden layer. This makes the network converge better and faster and prevent over-fitting.[28]

Fully connected multi-layer perceptron

Several layers of convolution and max-pooling are followed by the Fully Connected Multi-Layer Perceptron (FC-MLP) layers. A fully connected layer connects the neurons from previous layer to its perceptrons. This aims to bring all the possible information that has flown through the network to the decision layer (output layer).

Dropout

The dropout is often applied during the training phase of a fully connected multi-layer perceptron, precisely during back-propagation. Dropout removes some neurons and their connections temporarily from the network, and gradients are then computed only for the remaining neurons and connection weights according to back-propagation rules, as if the dropped neurons and connections were not present in the network. For example, using a 30% dropout will force the back-propagation to use only 70% of the perceptrons in that layer (these neurons will be randomly selected at each back-propagation). In other words, the neurons of the same layer will not learn the same features from the samples (inputs). This allows the network to make a better generalisation and prevent over-fitting.[26]

3. LITERATURE REVIEW

This chapter presents a literature review on the prediction of patients deterioration in the intensive care unit and the use of convolutional neural networks on electronic health records.

3.1 Prediction of patients' deterioration

Many studies have focused on predicting diagnosis, ICU risk of death or cardiac arrest using EHR data. Predicting diagnosis is a task that usually involves a relatively long patient history data to output the most probable diagnosis in the upcoming days or months. The patient can be diagnosed with several illnesses that are already present during the prediction time. For example, Lipton et al.[36] conceived a set of machine learning methods that aim to predict diagnosis in the upcoming 180 days. In their work, the patients were diagnosed with a set of 14 illnesses.

The assessment of patients' deterioration is a task sensitive to minute-by-minute changes in the patient's health status during the ICU stay. Therefore, it relies on a specific target (ex. death or cardiac arrest) compared to diagnosis prediction which usually focuses on large outcome possibilities (diabetes, asthma, neoplasm, scoliosis, ...) and is a task that does not only involve ICU stays. Some of these targets (deterioration) are known to be an in-hospital outcome (predictions are made at the admission time and aim at the end of the stay) rather than an assessment of the patients' health during their ICU stay (several successive predictions during the stay). For example, Churpek et al.[8] trained a multinomial logistic regression on EHR data to predict the probabilities of a Cardiac Arrest (CA) and a patient's transfer from the ward (a non-intensive unit) to the ICU. They reported an 88% AUROC on their first target (CA) and 77% AUROC on the second target (ICU transfer). A similar study conducted later by Churpek et al.[9] used another dataset to predict the same targets (cardiac arrest and death). They trained random forests and logistic regression models. Their results showed that random forest model was the most accurate model and outperformed VitalPAC (an early warning score for cardiac arrest). The authors focused on predicting deterioration in the ICU where

the patient's EHR history window and the prediction window are much shorter than the ones used in diagnosis prediction (average ICU stay is 4 days).

Other works have focused on predicting deterioration focusing on Sepsis as a deterioration pattern in the ICU. It has been reported by Fleischmann et al.[19] that Sepsis is among the most important causes of in-hospital deaths. Sepsis has been redefined in 2016 as Sepsis-3 "A life-threatening organ dysfunction caused by a dysregulated host response to infection" [52]. It corresponds to a suspicion of infection and an increase of at least 2 points in the SOFA score. In this vein, T. Desautels developed a machine learning classification system called *InSight* that uses multiple input parameters from an EHR data to predict Sepsis-3. In their work, sepsis detection attains 88% AUROC [14]. Similarly, Mitchell et al.[38] developed a logistic regression model to detect an increase in SOFA score (sepsis risk) in the next 12-24 hours using EHR data.

The aforementioned works focus on predicting Sepsis which is based on a change in the SOFA score and a suspicion of infection. The scoring system in the SOFA has been defined by researchers based on their knowledge of deterioration in contrast to the LODS or APACHE scores which have been statistically derived and are prone to be less biased. In addition, the studied population concerns only patients suspected of infection. Therefore, to fully evaluate goodness of such a model, predicting deterioration on the other set of patients (patients that are not suspected of infection) needs to be evaluated since the model was not trained on this set.

3.2 Convolutional neural networks applied on electronic health records

Convolutional neural networks were used to model multivariate clinical time-series by transforming these time-series to a regularly sampled set of data. Zheng et al.[64] applied the convolution filter over the temporal axis of the data without excluding any parameter from the study. Thus, relevant deterioration patterns and important features were discovered by the CNN at different timescales. Most of the time, these patterns are not easily detected or driven by conventional statistical feature extraction and conventional machine learning methods. Similarly, Drewry et al.[15] demonstrated that certain patterns and changes in body temperature that occur over multiple hours have a relatively high predictive value for sepsis prediction. It is unlikely that predefined parameters and their statistical derivatives will capture the most predictive variations at important time-steps.

The patient's EHR data is known to be longitudinal because the patient's condi-

tion changes over time. Thus, the temporal information has to be included in the modelling part as done by Wang et al.[60] to extract relevant features from EHR matrices. They used convolutional matrix factorisation to detect deterioration patterns across patient EHR matrices. In this work, the size of the EHR matrix and the convolutional filter of the CNN was set to a specific length. Therefore, the detected patterns in these time-based matrices had a similar deterioration time-length. Thus, longer or shorter time-wise deterioration is not likely to be captured given that each patient has a unique matrix based deterioration patterns.

To overcome this issue, Cheng et al.[6] constructed a Temporal Matrix Based data to represent patients' EHR data as temporal matrices with one dimension representing the time and the other dimension the parameters. They set the size of the matrix as well as the convolutional kernel and the number of filters in the CNN as tunable hyper-parameters. It means that during the training of the CNN, these parameters were tested with several values and the best ones were selected at the end. They were also able to detect the top-10 relevant input features by recording the neurons whose gained the highest weights in the first layer.

This brief review's purpose was to analyse the common research trends related to patients deterioration prediction using machine learning. From this review, it appears that most of the studies aimed to predict a specific target related to ICU outcome such as CA or death or to assess patients health through detecting Sepsis (a change of 2 in the SOFA score). All these studies trained a multitude of machine learning methods and focused on a 12 to 24 hours deterioration patterns prior to prediction time.

4. RESEARCH METHODOLOGY AND MATERIALS

The previous chapters contain definitions about patients health assessment in the ICU, machine learning basics, and a brief literature review. This chapter describes the dataset that has been used in this study, as well as the preprocessing and machine learning methods applied to the data to predict patients organ deterioration in the ICU.

As explained in chapter 2, there are several scoring systems that allow assessment of patients health during an ICU stay. The LODS score was chosen as a target to predict as it is a scoring system that permits the evaluation of the organ-wise severity. In fact, the risk of death is highly related to the number of dysfunctional organs [63]. As mentioned earlier in chapter 2, SOFA is a score that has been derived by clinicians based on their own knowledge of organ dysfunction whereas LODS has been generated using a logistic regression. In addition, it has been demonstrated that LODS has a better AUROC in patients' mortality prediction compared to SOFA. [46].

4.1 Dataset exploration

This section will describe the dataset as well as the preprocessing methods and techniques used on it.

4.1.1 Description

The FINNAKI data were collected in 17 ICUs in Finland between September 1^{st}, 2011, and February 1^{st}, 2012.[44]. The total number of patients in this dataset is 2969 (only adults are considered).[44]

The data are distributed over several files including over one thousand features. Each file has been analysed carefully in order to include all the features (variables)

available in a real ICU theatre. The final data include many features such as patient demographics, hemodynamic variables, pulmonary variables, drugs, ICU severity scores (APACHE, Glasgow).

4.1.2 Features

At first, removing some measurement variables (features) from the data was necessary. Prof. Ville Pettilä helped during this phase. The goal was to define a restricted subset of meaningful variables (features) which will be most likely available during the real-world usage in the ICU. The aim was to keep in features that could be beneficial in order not to compromise the performance of the model.

The collected variables can be divided into four categories:

1. **Hemodynamics**
   - Heart rate: Heart Rate (HR) calculated in beats per minute.
   - Systemic arterial pressure: Systemic Arterial Pressure (SAP), which has a systolic (SAPs), diastolic (SAPd) and a mean (SAPm) variant. Systolic blood pressure is the arterial pressure on the contraction of the heart, while diastolic blood pressure is the pressure in between contractions.
2. **Pulmonary**
   - Respiratory rate: Respiratory Rate (RR) calculated in breaths per minute.
   - Arterial blood gas: arterial blood gas analysis evaluates the performance of the lungs. It measures the partial pressures of oxygen (PaO_2) and carbon dioxide ($PaCO_2$). A low PaO_2 indicates that the patient is not oxygenating properly. A high $PaCO_2$ implies under-ventilation, a low $PaCO_2$ hyper-ventilation.
   - Positive end-expiratory pressure: Positive End-Expiratory Pressure (PEEP) represents the lungs' pressure at the end of the expiratory phase. A value different than 0 denotes the presence of mechanical ventilation (MV).
   - Fraction of inspired Oxygen: Fraction of inspired Oxygen (FiO_2) is the fraction or percentage of oxygen in the lungs.
3. **Drugs**

   The following drugs are known as vasopressors. They have been measured by μg (dose) per kg (body) per min (time)

- Norepinephrine.
- Epinephrine.
- Dopamine.
- Dobutamine.

4. **Additional features**
   The dataset also includes the following features:

   - Age.
   - Gender.
   - Rectal temperature (TRECT).
   - SOFA score (cardiovascular).
   - LODS score (cardiovascular and pulmonary).

4.1.3 Preliminary exploratory data analysis

Data analysis is a fundamental step to accomplish before every machine learning experiment in order to understand the data. Hence, more details about the data features will be given in this subsection.

Table 4.1 contains a statistical summary of the data features described earlier.

Category	Name	Clinical range	Count	Mean	Standard deviation	Minimum	25 percentile	50 percentile	75 percentile	Maximum	Number of patients
Hemodynamic	**HR**	[0, 300]	$7.473\,906 \times 10^{6}$	8.539×10^{1}	2.751×10^{1}	0.0	7.200×10^{1}	8.400×10^{1}	9.700	2.977×10^{4}	2957
	SAPs	[0, 200]	$7.308\,440 \times 10^{6}$	1.287×10^{2}	2.647×10^{1}	-3.190×10^{2}	1.100×10^{2}	1.260×10^{2}	1.460×10^{2}	3.600×10^{2}	2954
	SAPm	[50, 300]	$7.287\,624 \times 10^{6}$	1.265×10^{12}	3.416×10^{15}	-3.190×10^{2}	7.100×10^{1}	8.100×10^{1}	9.300×10^{1}	9.223×10^{18}	2592
	SAPd	[30, 180]	$7.260\,634 \times 10^{6}$	6.114×10^{1}	2.136×10^{1}	-3.150×10^{2}	5.100×10^{1}	5.900×10^{1}	6.800×10^{1}	7.7070×10^{3}	2593
Pulmonary	**RR**	[0, 60]	$5.472\,639 \times 10^{6}$	1.781×10^{1}	5.767×10^{1}	0.0	1.300×10^{1}	1.700×10^{1}	2.200×10^{1}	6.530×10^{4}	2475
	$PaCO_2$	[0.5, 15]	6.1036×10^{4}	5.404	1.905	0.0	4.500	5.100	5.800	1.020×10^{2}	2931
	FiO_2	[0, 100]	$2.401\,166 \times 10^{6}$	4.189×10^{1}	1.500×10^{1}	0.0	3.200×10^{1}	3.960×10^{1}	4.900×10^{1}	9.070×10^{2}	2593
	PEEP	[0, 20]	$1.546\,374 \times 10^{6}$	7.016	2.784	-3.400×10^{1}	5.100	7.000	8.200	8.981×10^{2}	1786
Drugs	**Norepinephrine**	[0, 6]	$1.378\,47 \times 10^{5}$	$1.149\,717 \times 10^{-1}$	$1.848\,943 \times 10^{-1}$	0.0	3.33×10^{-2}	6.67×10^{-2}	1.333×10^{-1}	5.846200	1841
	Dobutamine	[0, 10]	1.3198×10^{4}	3.239619	2.466288	0.0	1.6026	2.5316	4.0816	$1.515\,490 \times 10^{1}$	1841
	Epinephrine	[0, 3]	4.137×10^{3}	$1.164\,017 \times 10^{-1}$	$4.892\,941 \times 10^{-1}$	0.0	1.73×10^{-2}	3.45×10^{-2}	6.94×10^{-2}	5.952400	1841
	Dopamine	[0, 10]	7.69×10^{2}	4.602720	2.805977	0.0	2.4691	4.4444	6.6667	$1.904\,760 \times 10^{1}$	1841
Additional features	**TRECT**	[32, 42]	$2.825\,584 \times 10^{6}$	$7.245\,858 \times 10^{1}$	$1.716\,803 \times 10^{4}$	-4.000×10^{-1}	3.670×10^{1}	3.730×10^{1}	3.780×10^{1}	$8.330\,779 \times 10^{6}$	1796
	Age	[18, 110]	2.737×10^{3}	61	16	18	51	64	74	97	2737
	Gender	[Female, Male]	2.737×10^{3}	-	-	-	-	-	-	-	2737

Table 4.1 *Statistical analysis of the Finnaki dataset*

Outliers

The clinically valid range for each data feature shown in Table 4.1 has been defined by prof. Ville Pettilä. The lower and upper quantiles for all these data features are within their respective clinical range.

Missing data and resampling

From table 4.1, the most frequently measured data features are the hemodynamic measurements then the pulmonary measurements[1]. Drugs are not as frequently measured as the previously cited features because the system only records the change in the administered concentration (drugs).

Table 4.1 shows that there are many missing feature values for a high amount of patients. These attributes are missing not at random. Table 4.2 displays the imputed value for these fatures.

PEEP and drugs have been imputed with 0. When these attributes are missing in a patient, it means that the patient was either not on mechanic ventilation (missing PEEP data) or that the patient was not administered drugs (missing drugs data). For TRECT, the choice has been made by prof. Ville Pettilä to fill missing data with a clinically normal value (37.3). He explained that a missing value in this attribute means that the nurses did not measure it and therefore assumed it to be normal, unlike the other features (hemodynamic, pulmonary and drugs) that are automatically measured by the system.

Parameter	Imputed value
PEEP	0
Drugs	0
TRECT	37.3

Table 4.2 *Imputed values according to the missing parameter.*

Table 4.3 represents a statistical analysis of the sample interval in minutes for a frequently measured parameter (HR) and an infrequently measured parameter ($PaCO_2$). The statistical measures were calculated from the time difference between two consecutive events for the same parameter. It is important to note that

[1] $Frequency = \frac{Count}{Patients}$

this analysis has been carried on raw data (before imputation).

Features	Count	Maximum	Minimum	Mean	Standard deviation	Median	25 percentile	97 percentile
$\mathbf{t(HR)}_n$ - $\mathbf{t(HR)}_{n-1}$	7473906	10151	0	2.21	7.88	2.0	1.0	5.0
$\mathbf{t(PaCO_2)}_n$ - $\mathbf{t(PaCO_2)}_{n-1}$	61036	3286.0	0	201.72	121.92	179.0	118.0	263.0

Table 4.3 *Temporal analysis (minutes) of HR and $PaCO_2$ where $t(feature)_n$ denotes the n^{th} sample with respect to time.*

For the most frequently measured feature (HR), the mean time-span between two consecutive events is 2.21 minutes and for the more infrequently measured feature ($PaCO_2$) the time span is larger, about 201 minutes. In order to handle this issue, re-sampling the data has been necessary.

A one-minute sampling interval has been chosen at the beginning of the study in order not to miss any important event. Forward filling of the latest available value was used to fill the gaps (newly created data points) as done by Lipton et al.[36] and Cheng et al.[6]. However, due to the huge amount of data generated with upsampling (one-minute interval), a down-sampling technique has been applied where the sample corresponding to the highest LODS score has been selected.

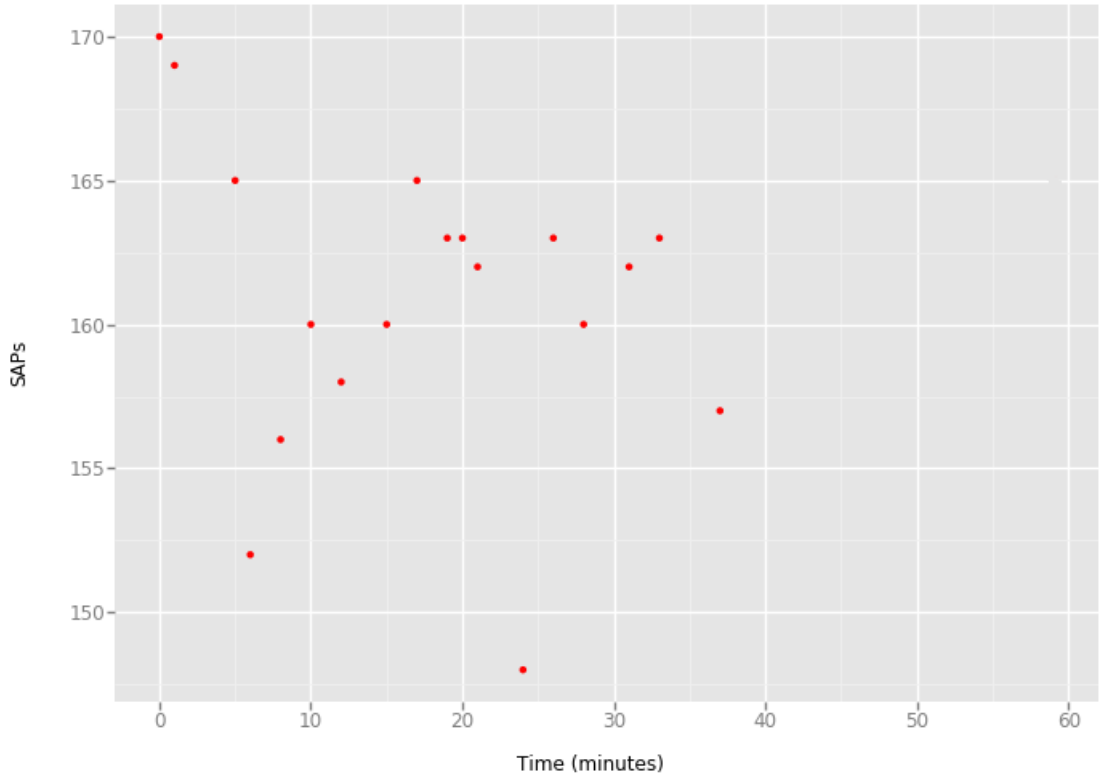

Figure 4.1 *SAPs values for a single patient (raw data).*

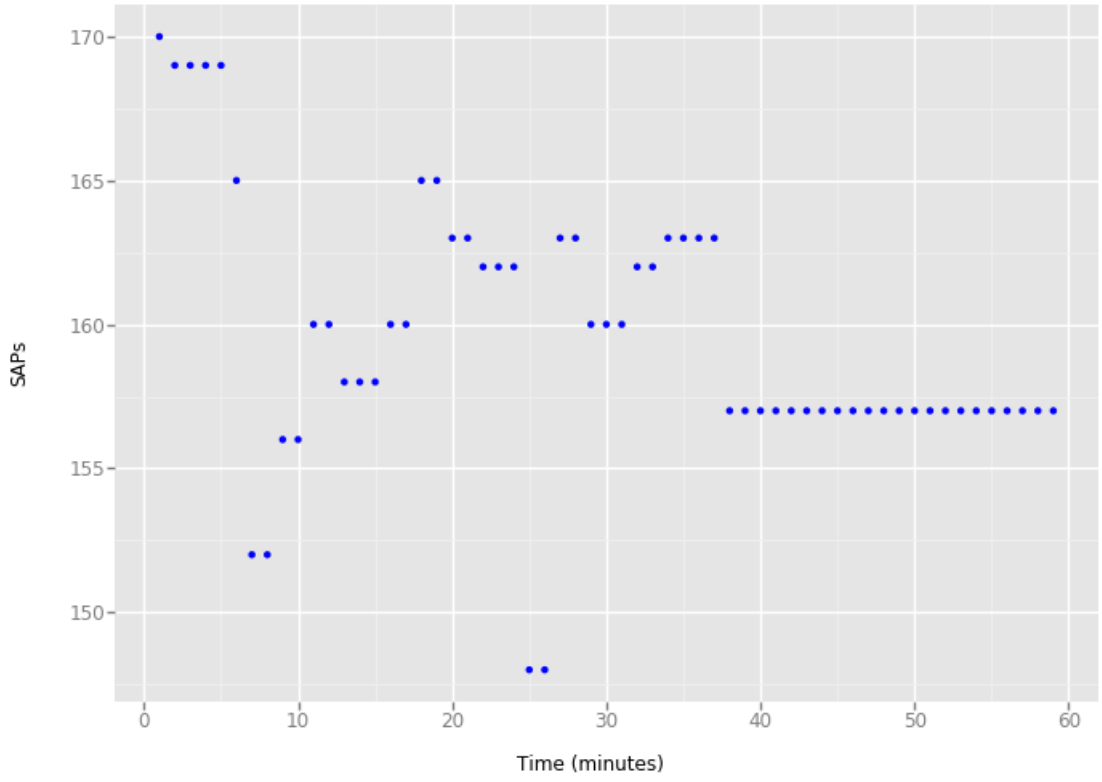

Figure 4.2 *SAPs values for a single patient after 1 minute resampling.*

Figure 4.1 and 4.2 represents SAPs values over one hour for a single patient before and after one-minute resampling.

4.2 Targets description

As described earlier, this project aims to predict organ dysfunction within the ICU. The most relevant organ failures to detect are cardiovascular and pulmonary dysfunction because of several reasons described earlier. Therefore, the targets were defined as being the worst LODS (max(cardiac LODS, pulmonary LODS)) in the next three hours from the moment of prediction.

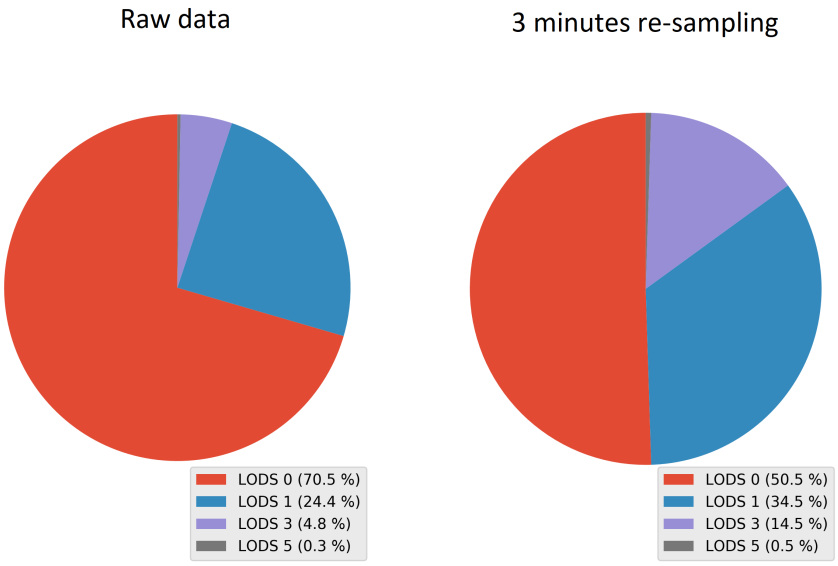

Figure 4.3 *Distribution of max LODS before and after resampling.*

Figure 4.3 represents the LODS distribution over the dataset before and after resampling. It is clear that resampling had a positive impact on the targets distribution since the percentage of critical LODS (3 and 5) have increased (from 4.8% to to 14.5% and from 0.3% to 0.5% for LODS 3 and LODS 5 respectively). Thus, the targets were redefined as the joint organ (cardiovascular and pulmonary) dysfunction which is represented by LODS 3 and 5 (15%) and potentially safe organs (cardiovascular and pulmonary) as LODS 0 and 1 (85%). This transforms the original multi-class classification task into a binary classification task represented according to the following table.

Maximum LODS (cardiovascular and pulmonary)	Class
{0, 1}	**0**
{3, 5}	**1**

Table 4.4 *Maximum cardiovascular and pulmonary LODS class representation*

4.3 The prediction task

In order to assess organ dysfunction in real time in the ICU, predictions had to occur regularly at tight intervals. The chosen re-sampling rate (three minutes) was relatively short compared to the original sampling of the dataset and was chosen as a prediction rate.

As done by Cheng et al.[6] and Lipton et al.[36] to predict medical events using time-series dataset, two windows had to be chosen, a history window from which the input features of the classification task are gathered and a prediction window from which the target class is determined. A too large history window would give more information about the patient's dynamics but would also increase drastically the dimensionality of the dataset and thus make the training hard or almost impossible. A too short history window would make the training faster but not necessarily easier as the model would probably not be able to capture deterioration if it happened before the events collected in the history window.

Concerning the prediction, a large enough window had to be chosen. According to prof. Ville Pettilä, a prediction window of 3 to 6 hours helps clinicians prevent organ failure. A 90 minutes history window and a 3 hours prediction window was chosen in this study.

4.4 The training method of the machine learning models

The following section will describe the machine learning methods that have been used in this work.

4.4.1 Conventional methods

Feature extraction

In order to train machine learning methods that do not detect patterns over the time dimension, it was necessary to generate features that are able to provide information about a patient's history in the ICU. Thus, statistical features were generated for each of the previously mentioned data features excluding age and gender which are time-independent. The statistical features included the mean, maximum, minimum, variance, mode and discrete wavelet coefficients over the 90 minutes history window.

A dense vector of 268 statistically generated features was given as input to the LR and RF models.

Logistic regression

A logistic regression has been trained using the 268 input features described above. The following parameters were optimised using a grid search with a 5-fold cross-validation.

- C, the regularisation coefficient ($C = \frac{1}{\lambda}$ from formula 2.8).
  $C \in [10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^{2}]$.
- The choice of norm for the penalty term: either l_1 or l_2.

Random forest classifier

A random forest classifier was trained using the previously described input features. The number of trees, as well as the evaluation criterion of the classifier, were empirically evaluated using grid search with a 5-fold cross-validation.

- The number of decision trees $\in [10, 20, 30, 40, 50]$.
- The choice of the evaluation criterion: either "gini" or "entropy".

As shown earlier in chapter 3, the imbalanced class distribution of the dataset led us to use a technique that uses class weights during the training. Class weights for

both LR and RF were set to "balanced". It basically penalises the models with a coefficient proportional to the distribution of the class of the erroneous output.

Scikit-learn [45] was used to scale the data and train RF and LR models.

4.4.2 Convolutional neural networks

A convolutional neural network was trained to predict pulmonary and cardiac dysfunctions. The CNN model was composed of several convolutional layers (from 3 to 6). Each one of these layers was followed by a max-pooling layer and a batch normalisation. These convolutional layers were further followed by a fully connected layer of 100 neurons, which was finally connected to a 2-neurons *softmax(x)* layer for classification.

As done by Cheng et al.[6], the input data was represented by a fixed size Temporal Matrix Based Representation such that one dimension corresponds to the time axis and the second one corresponds to the features. Convolutions were performed over the time axis to capture trends and patterns of the evolution of these longitudinal features.

Training deep multi-layered neural networks is known to be hard due to the huge number of trainable parameters (weights). In addition, the choice of the hyper-parameters highly influences the learning of the parameters and the performance of the network [33, 42]. Therefore, only the number of hidden layers, the size of the convolutional kernel as well as the batch size were tuned (Table 4.5). The other parameters were set to their default values in the machine learning environment used in this thesis (Table 4.6).

Hyper-parameter	**Value**
Number of hidden layers	**{3, 4, 5, 6}**
Size of convolutional kernel	**{3, 5, 7}**
Batch size	**{64, 128, 256}**

Table 4.5 *CNN's tunable hyper-parameters.*

As explained earlier in chapter 2, increasing the depth of the network increases

its complexity and therefore its non-linearity. Thus, the network would be able to capture complex trends and patterns across the data. However, a too complex neural network (too deep) tends to over-fit rapidly. Therefore, this parameter was set to an empirical evaluation.

The size of the convolutional kernel was also empirically evaluated as done by Cheng et al.[6]. It basically implies that the CNN will optimise this parameter according to the varying time-wise deterioration trends.

A batch size of 128 means that the weights are updated after each 128 input samples. This parameter was also set to empirical evaluation.

The fixed hyper-parameters are presented in Table 4.6

Hyper-parameter	**Value**
Activation function	**ReLU, Softmax**
Training algorithm	**ADAM**
Learning rate	**10e-3**
Weight decay	**0**
Loss function	**Binary cross-entropy**
Weights initialisation	**Glorot Uniform**
Regularisation	**Dropout (0.5)**
Epochs	**500**

Table 4.6 *CNN's fixed hyper-parameters.*

ReLU(x) was chosen as the activation function of the convolutional and the fully connected hidden layers. This choice was motivated by earlier findings introduced in chapter 2. The output layer had a *softmax(x)* activation function. *Softmax* is well suited for this classification task because the classes are dependent.

The chosen training algorithm, ADAptive Moment estimation (ADAM) is a learning algorithm based on Stochastic Gradient Descent (SGD). ADAM adjusts the learning

rate separately for each parameter (weight) during each mini-batch, unlike SGD which uses the same learning rate for all weights.[32] ADAM takes into consideration the changes that occur in the input features as different data samples are presented over iterations of the learning. In other words, ADAM allows a broad update for rare and significant events and minor updates for the most common events. This gives a higher importance to the examples where the rare features appear. The learning rate and weights decay have been initialised with their default values as in the original paper describing ADAM [32].

Binary cross-entropy was chosen as the loss function of the CNN. Cross-entropy is more adapted to classification problems and more advantageous than the quadratic error function because it suffers less frequently from the local optimum problem [23].

Concerning the model weights initialisation. A rule of thumb suggests to randomly initialise the weights. However, random initialisation can cause weights saturation and hence prevent proper network learning. In order to skirt this problem, Glorot initialisation was used.[22] This initialisation forestalls the weight's saturation problem and consequently allows to obtain a better performance from the learned model.

A 0.5 dropout was used as a regularisation hyper-parameter. As described in chapter 3, dropout prevents over-fitting by deleting a 0.5 ratio of randomly selected neurons during the back-propagation phase. Batch normalisation was also used, it was applied to the time dimension (for each input parameter separately).

During the training of the CNN, a validation is performed every 10 iterations (epochs). Moreover, a patience of 10 validations was granted to the CNN before stopping the training process. Thus, if after 10 validations, there is no improvement over the validation loss, the execution stops before achieving 500 epochs.

It is important to note that the model was trained using class weights defined previously in formula 2.1.

The model was developed under Keras[2] [7] and Tensorflow[3] [1]. The availability of an NVIDIA GPU that supports CUDA® Deep Neural Network library (cuDNN) accelerated the training process. It has been reported that the training process of neural networks is 10 to 20 times faster than with CPUs [29].

[2]Version 2.0.8
[3]Version 1.3.0

5. RESULTS

This chapter will introduce the performance evaluation of the machine learning methods that have been trained for predicting organ dysfunction.

5.1 Performance measures

Sensitivity and Positive Predictive Value (PPV) were chosen as performance metrics for evaluating the produced models. The performance of the predictions were defined as a minimum of 50% PPV with the highest possible sensitivity. PPV has been set to 50% in order not to produce too many false alarms. It is important to note that sensitivity and PPV of a model can be traded off by adjusting the classification threshold. However, these two measures were reported using the trained parameters as is.

5.2 Baseline and machine learning methods

The baseline was defined as a naive rule-based model that assumes that the state of the patient will remain the same during the next three hours. In addition, LR and RF classifiers were trained using grid search 5-fold cross-validation for optimising the classifiers' parameters (as explained earlier in chapter 5). The Cross-Validation (CV) set contained a total amount of 900 patients. The final models were tested on an independent test set of 300 patients.

The following subsection contains the confusion matrices yielded by the classifiers on the test set (300 patients) calculated per sample.

5.2.1 Naive classifier

The following table represents the confusion matrix derived from the test set.

	Predicted: 0	Predicted: 1
True class: 0	**507084**	34025
True class: 1	71360	**33855**

Table 5.1 *Confusion matrix of the naive classifier.*

The naive classifier predicts the future value to be the current value, thus in Table 5.1 the bottom right shows the amount of positives (bad outcome) that remained positives during the next three hours. These represent 33855 out of 105215 positive samples during the next three hours (32.17% sensitivity) and therefore constitutes approximately a third of the targets. In addition, the PPV is equal to 49.95% which means that using this last assumption (no change in patients status), half of the samples classified as positives were true positives.

5.2.2 Logistic regression

The following parameters yielded the best mean CV performance (sensitivity) for logistic regression.

- Regularisation coefficient: $C = 10^{-2}$.
- Penalty term: l_1.

The following table represents the confusion matrix that was derived from the test set using a logistic regression with l_1 regularisation:

	Predicted 0	Predicted 1
True class 0	**528029**	34895
True class 1	71147	**34068**

Table 5.2 *Confusion matrix of logistic regression classifier.*

From Table 5.2, the number of positive samples that have been predicted correctly is 34068 which represents a 32.37% sensitivity versus 31.17% for the naive classifier.

These results simply show that the LR model did not achieve a better performance neither in sensitivity nor in PPV (49.40% LR versus 49.95% for naive predictor).

5.2.3 Random forest

The following parameters yielded the best mean CV performance (sensitivity) for random forest.

- Number of decision trees: $N = 40$.
- Evaluation criterion: “gini”.

	Predicted 0	Predicted 1
True class 0	**502977**	39002
True class 1	66286	**38929**

Table 5.3 *Confusion matrix of random forest classifier.*

Table 5.3 represents the confusion matrix of the RF classifier. This RF model was trained using 40 decision trees and the "gini" performance criterion.

The reported confusion matrix shows that RF predicted more accurately the positive class with a sensitivity of 36.99% which represents a gain of 1.149 compared to the naive classifier without exhibiting a loss in the PPV (49.95%).

5.2.4 Convolutional neural network

The following subsection will present the tuning process of the hyper-parameters and their impact on the performance of CNN as described earlier in chapter 4.

The results shown in the following tables represent the mean of three consecutive runs for each configuration. This method was carried out due to the randomness in weights initialisation of the CNN. In fact, on a single run, the results are not reproducible (Using the Nvidia cuDNN software which do not allow yet, to save the random seed parameter).

The training set contains 800 patients, the validation set 100 patients and the test set 300 patients. These 300 patients (test set) are similar to the ones used for testing the naive, LR and RF models.

Number of layers

Increasing the number of layers of CNN increases its complexity by including more non-linear operations through the layer-to-layer cascade. Therefore, the number of layers was the first hyper-parameter tuned.

Number of hidden layers	Sensitivity (%)	PPV (%)
2	71.15	46.98
3	72.23	45.66
4	**73.33**	**48.69**
5	73.05	46.25
6	71.80	42.14

Table 5.4 *Performance measures according to the depth of the CNN.*

Table 5.4 displays the impact of the depth of the CNN on its performance on the test set. It is important to note that the size of the convolutional kernel has been set to 7 and the batch size to 512 in order to make the training faster.

The depth of the network is defined as the number of hidden layers which is represented by $n - 1$ convolutional layers and 1 fully connected layer.

From table 5.4, the sensitivity increases by nearly 1% with each additional hidden layer. For instance, three hidden layers (two convolutional and one fully connected) reported a sensitivity of 71.15% against 73.33% with four hidden layers (three convolutional and one fully connected) which was the highest sensitivity. Then, this performance measure starts decreasing when the number of hidden layers is larger than four. For example, five hidden layers yielded 73.05% and six hidden layers reported 71.80%. The highest PPV (48.69%) was also yielded by the 4-layered CNN.

Thus, 4 was chosen as the optimal number of hidden layers for this configuration[1].

Size of the convolutional kernel

The size of the kernel is another important hyper-parameter to tune since it allows to specify the length of the time-windows to process by the CNN inside the 90 minutes history window.

Kernel size	Sensitivity (%)	PPV (%)
3	**76.15**	50.15
5	74.59	**50.23**
7	73.33	48.69

Table 5.5 *Performance measures with respect to the size of the convolutional kernel.*

Table 5.5 shows the impact of the width of the convolutional kernel on the CNN's performance. It appears that reducing the kernel size consistently improves the sensitivity of the CNN. In fact, a kernel size of three yielded the highest sensitivity of 76.15% compared to a kernel size of five (74.59%) and seven (73.33%). Reducing the kernel size also improved the PPV from 48.69% (kernel size of seven) to 50.23 (kernel size of five)%. This last slightly dropped (by 0.08%) with a kernel size of three. Thus, 3 was chosen as the optimal kernel size as it substantially improved the sensitivity for an insignificant drop in the PPV.

Batch size

The size of training batch represents the number of input samples to back-propagate on during the training phase.

[1]Size of convolutional kernel equal to 7 and a batch size of 512 samples.

Batch size	Sensitivity (%)	PPV (%)
64	77.23	51.15
128	**77.90**	51.02
256	75.90	**52.80**
512	76.15	50.15

Table 5.6 *Performance measures of the CNN with respect to the size of the batch.*

Table 5.6 shows the effect of the size of the batch on the performance of the CNN. The best performance in terms of sensitivity (77.90%) was obtained with a batch size of 128 samples. The PPV corresponding to this last batch size was not the highest among the ones yielded per different batch sizes. That is, the PPV loss is minor compared to the gain in sensitivity.

5.3 In-depth analysis of the convolutional neural network

In the following section presents a further analysis of the performance of the best machine learning model (77.90% sensitivity, 51.02% PPV).

5.3.1 Patient-wise analysis

The results presented in the previous section contain performance measures calculated as an average for all the test set patients (300 patients). This subsection present the sensitivity and the PPV of the classifier with a patient-wise estimation, where estimates are calculated for each patient separately, and means are given as a final result.

As specified earlier, the test set contains 300 patients that were randomly selected at the beginning of the study. Thus, these patients have the same overall distribution of targets as in the training data set.

	Positives (%)	Sensitivity (%)	PPV (%)
Test set	17.26	61.32	42.72

Table 5.7 *Ratio of positives, sensitivity and PPV over the test set.*

Table 5.7 represents the positives' ratio in the test set and the performance of the CNN in terms of sensitivity and PPV. The sensitivity and PPV were computed for each test patient, and then averaged over patients.

Performance measures by the length of ICU stay

The following table presents the ratio of positives, sensitivity as well as the PPV for patients with respect to the length of the stay.

Length of the stay t (days)	Positives (%)	Sensitivity (%)	PPV (%)
$t < 2$	14.03	55.90	35.38
$2 \leq t < 4$	17.93	61.70	45.49
$4 \leq t < 6$	19.67	70.01	48.68
$t \geq 6$	17.42	57.67	41.34

Table 5.8 *Ratio of positives, sensitivity and PPV with respect to the length of the stay (days).*

Table 5.8 shows the performance of the CNN with respect to the length of the ICU stay for the test set patients calculated as a patient-wise average.

The ratio of positives, the sensitivity, and the PPV consistently increases with the length of the stay and reaches its peak with the patients that stayed from 4 to 6 days (19.67% positives' ratio, 70.01% sensitivity, and 48.68% PPV). However, these measures decrease for patients who stayed in the ICU more than six days (17.42% positives' ratio, 57.67% sensitivity, and 41.34% PPV).

Performance measures by type of deterioration

This thesis focuses on forecasting cardiovascular and pulmonary dysfunctions. These two dysfunctions are valued with different scores. The cardiovascular dysfunction is assessed with 4 severity scores {0, 1, 3, 5}, and the pulmonary dysfunction with 3 severity scores {0, 1, 3}.

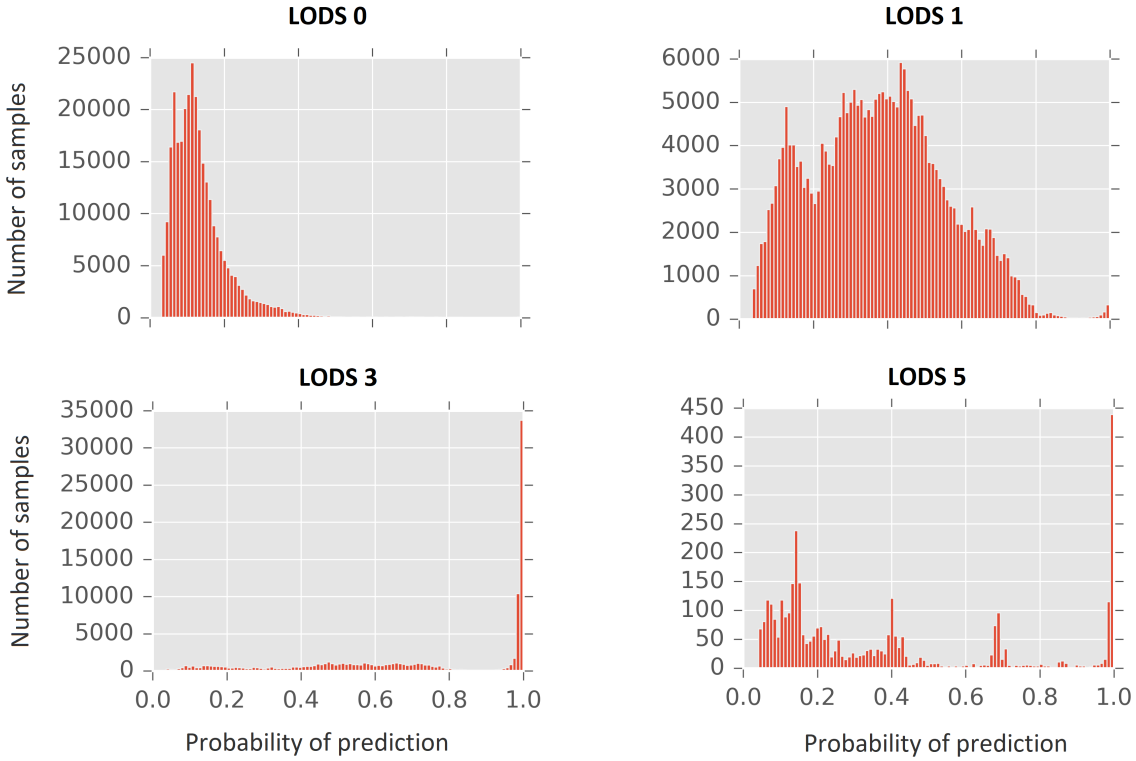

Figure 5.1 *Histograms of predicted probabilities of class 1 (LODS 3 or 5) yielded by the CNN, for samples from each of the four true LODS scores.*

Figure 5.1 represents four histograms corresponding to the maximum true LODS (cardiovascular and pulmonary) score during the prediction process. It simply answers the question: How well did the CNN predict the LODS score?

To answer this question, the samples of the test set were re-classified as per their LODS score (LODS 0, LODS 1, LODS 3, and LODS 5 respectively represented by their histograms) together with the probabilities yielded by the CNN for these samples.

For example, the histogram corresponding to LODS 0 suggests that the CNN for most samples belonging to LODS 0 a probability less than 0.4 to belong to class 1 (LODS 3 or 5). LODS 1 histograms show that the CNN did perform less well

in predicting LODS 1 compared to LODS 0. In fact, most of LODS 1 prediction's probabilities are around 0.5. However, it seems that the number of samples predicted with a probability higher than 0.5 consistently decreases till 0.8. Here, 0.5 appears to be the best threshold for predicting the correct binary class for samples from LODS 1. For LODS 3 histogram, most of the samples were detected with a probability higher than 0.4 with a peak near 0.98. However, LODS 5 probabilities of prediction were irregularly distributed in the interval [0, 1]. It is important to note that the LODS 0 was the prominent class with more than 50% of the samples belonging to LODS 0. LODS 1 was the second most frequent with 34.5% of the samples, then LODS 3 which is represented by 14.5%. LODS 5 is nearly nonexistent with 0.5%.

From the four previously shown histograms, it appears that the CNN is able to discriminate between LODS 0 (belongs to class 0), and LODS 3 (belongs to class 1) in a better way than LODS 1 and LODS 5.

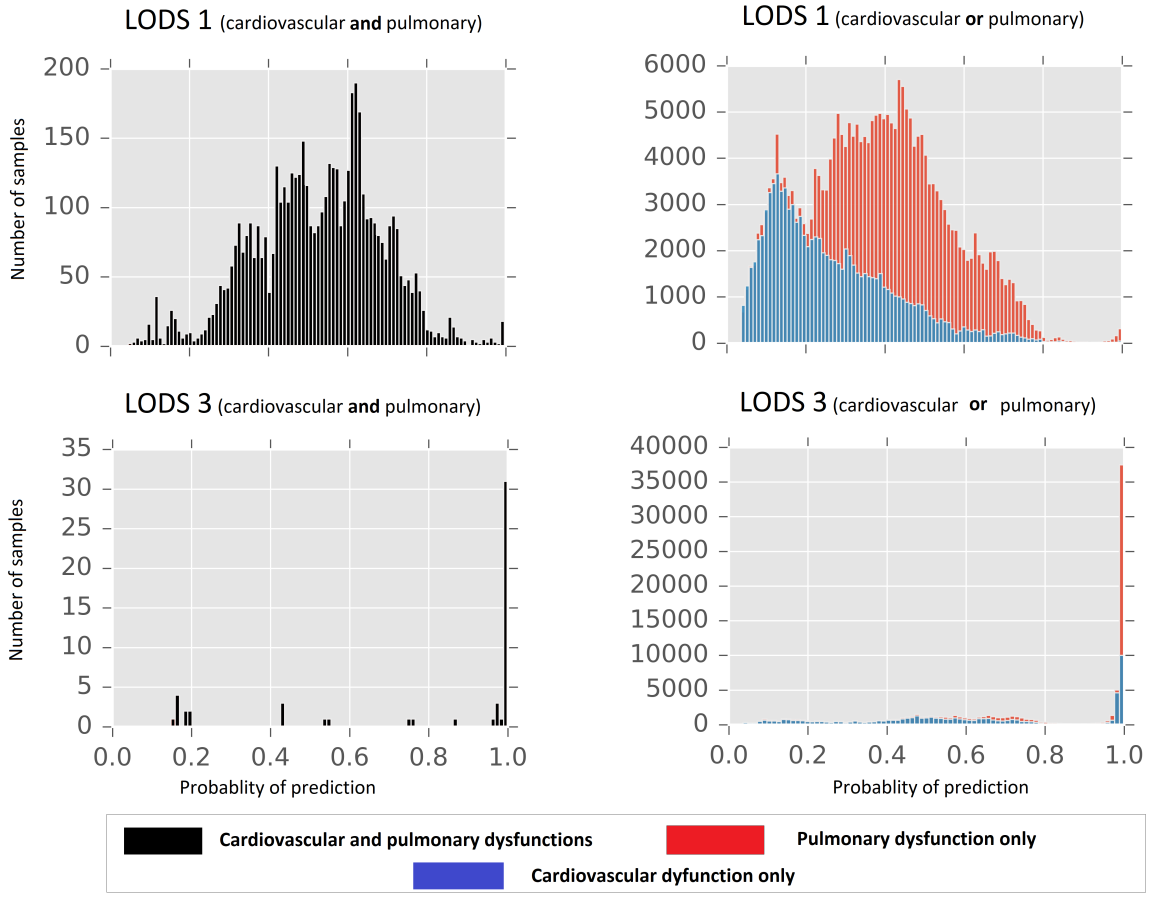

Figure 5.2 *Histograms of predicted probabilities yielded by the CNN by dysfunction type.*

Figure 5.2 displays the histograms for patients LODS 1 and LODS 3 prediction probabilities. For instance, the histogram showing the class 1 (LODS 3 and 5) prediction

probabilities for samples of LODS 1 that have both cardiovascular and pulmonary dysfunctions shows that most of them get probabilities between 0.4 and 0.6. Similarly, in the next histogram, the LODS 1 caused by a pulmonary dysfunction (free of cardiovascular dysfunction) shows the same pattern: A peak around 0.5. However, the LODS 1 caused by cardiovascular dysfunction are better classified by the CNN with with low probability to belong to class 1 (LODS 3 and 5).

The histogram showing the LODS 3 caused by both cardiovascular and pulmonary dysfunctions together with the histogram displaying the LODS 3 that is caused by only one dysfunction suggest that the CNN reacts well in all these cases as the majority of dysfunctions are predicted to the correct class 1 (LODS 3 and 5) with a threshold of 0.5, with a peak near 0.95.

5.3.2 Confidence interval

This subsection presents the 95% confidence interval (CI) (over the test set) which has been estimated based on the Jackknife error, leaving out data from one patient at a time.

	Batch evaluation	**Patient-wise evaluation**
95% CI	$[77.9080, 77.9150]$	$[57.7890, 64.9580]$

Table 5.9 *The 95% CI of the CNN over the test set.*

Table 5.9 shows that the 95% CI is particularly small in the batch evaluation (interval size is 0.007). However, it is larger in the patient-wise evaluation (interval size is 7.169).

6. DISCUSSION

The previous chapter presented the results and findings of the used machine learning methods to predict deterioration in the ICU. This chapter will present a discussion of those results and an analysis of their impact on predicting deterioration.

This work focuses on predicting cardiac and pulmonary deterioration as it is the most common reason for ICU admission. It also focuses on predicting the LODS as a deterioration score as described in Table 4.4. Thus, a maximum LODS score higher than 1 means that the organ is deteriorating.

The CNN model achieved a 77.90% sensitivity with 51.02% PPV on the test set. It is important to note that this evaluation is an average of the performance of the classifier on the samples of the test set. The patient-wise analysis returned a lower sensitivity and PPV (61.32% and 42.72% respectively). In addition, 95% CI is wider on the patient-wise analysis. This is mainly due to the length of the stay in the ICU as patients who stay longer tend to increase the average sensitivity and PPV. In fact, results presented in Table 5.8 show that the CNN yielded its best performance on patients that stayed in the ICU between 4 and 6 days. These patients represent the highest ratio of positives (19.67%) in the dataset and thus deteriorate more often than other patients.

The CNN has also shown the ability to discriminate between LODS 0 (deterioration-free) and LODS 3 (deteriorating) in Figure 5.1. This was mainly due to their high availability in their respective classes. In fact, LODS 0 represents nearly 60% of negative samples and 50% of the whole samples while LODS 3 represents 96.66% of the positive samples and 14.5% of the whole samples. In addition, the definition of the LODS score creates a distinguishable difference on the input parameters when the LODS score is 0 and when the LODS score is 3. In contrast, no such discernible difference exists between LODS 0 and 1 or 1 and 3. This explains the high overlapping between these 3 classes of LODS during the prediction and therefore, lowers the sensitivity of the classifier. The CNN accurately predicted more than 50% of the LODS 3 samples. For LODS 5, the classifier misses nearly all the test set samples during the prediction. It is important to note that LODS 5 is an under-

represented score with only 4.4% of the positives and 0.5% in the whole dataset. That is, the model was unable to learn from this nearly nonexistent class.

It is important to note that the high availability of LODS 3 in the positive class is mainly due to the LODS definition of severity. For instance, when considering the cardiovascular organ, the parameters generating the LODS (Table 2.2) shows that LODS 5 is only caused by a severely low heart rate or systolic blood pressure while an extremely high value in any of these parameters does not generate LODS 5. The increased severity is limited to 3 for all values higher than 140 for heart rate and 270 for systolic blood pressure. Concerning the pulmonary organ, the highest severity is limited to 3.

The patient's LODS (maximum between cardiovascular and pulmonary LODS) cannot be 0 unless both cardiovascular LODS and pulmonary LODS are equal to 0. Similarly, patients LODS 5 cannot be caused by a pulmonary dysfunction as the maximum scoring for it is 3. This reasoning cannot be applied to patients LODS 1 and LODS 3 as both of them can be caused by a different organ dysfunction. Therefore, it is important to check the performance of the CNN on this kind of classifications (LODS 1 and 3). Results shown in Figure 5.2 suggests that the CNN predicts the cardiovascular LODS 1 better than the pulmonary LODS 1 while the latter dysfunction is more represented than the cardiovascular. This is mainly due to the ability of the CNN to learn from the data. In fact, the input data is represented by a temporal matrix containing the input parameters on a history window of 90 minutes. Thus, the availability of the parameters highly impacts the variance of each parameter in the matrix during these 90 minutes. It has been shown in Table 4.1 that hemodynamic parameters are more available and thus prone to more changes compared to the pulmonary parameters. Thus, the input matrices were most likely able to contain redundant values from the pulmonary parameters than the hemodynamic parameters. This has possibly made the model over-learn from these redundant parameters and thus, performed worse than expected on detecting pulmonary LODS 1 that created most of the false positives. Concerning the cardiovascular and pulmonary LODS 1 (happening at the same time), the model was unable to learn to represent it even if it predicted the cardiovascular LODS 1 in a good manner. It is possible that the miss-classifications and incertitude created by the pulmonary LODS 1 confused the classifier. In contrast, the results shown for LODS 3 suggests that the model learned to classify both cardiovascular and pulmonary failures (happening both at the same time and not). This is mainly due to the ability of the classifier to learn from a majority score (LODS 3 represents more than 96% of the positives).

Compared to the RF and the LR models, the CNN performed almost two times better in terms of sensitivity while keeping nearly the same PPV. The input data representation in the RF and LR models did not allow a time-wise analysis of the parameters by these classifiers as they only allow a one-dimensional data input (their performance was comparable to the naive classifier). Therefore, two-dimensional windows had to be aggregated into a vector deleting the time information provided by these time-series. In contrast, two-dimensional data can be given as input to the CNN in order to perform dimensional convolutions on several parameters at once. The input windows were in the form of Temporal Matrix Based Representation as done by Cheng el al.[6]. Thus, complex deterioration trends and patterns were detected through several convolutional layers which increased the non-linearity and the complexity of the model. However, further development and data structure analysis has to be performed to confirm the lack of LR and RF models to successfully achieve this kind of tasks.

The performance of the CNN, suggests that this predictor is well-suited for predicting organ failure in the ICU. Importantly, the model was able to appropriately distinguish between LODS 0 and LODS 3 which are relevant scores in deterioration. In fact, knowing that the patient will have a LODS 0 score in the next 3 hours means that the practitioners will prefer to focus on other patients. On the other hand, knowing that a patient will have a LODS 3 score in the next 3 hours (with a PPV of approximately 50%) will allow enough time for medical intervention. Especially that LODS 3 often precedes LODS 5.

To the best of our knowledge, this thesis is the first to provide a prognostic model that predicts the LODS score as a deterioration criterion for cardiovascular and pulmonary organs in short-term ICU stay. Therefore, the CNN model excels in predicting deterioration because more than 90% of LODS 5 cases (predicted poorly) were previously diagnosed as LODS 3 during their ICU stay. In other words, the instantaneous change from LODS 0 to 5 happens only 10% of the time. Importantly, a good predictive performance was from only 19 input parameters available from admission time, in every intensive care unit present in the dataset.

Choosing the best hyper-parameters in the CNN was, as explained earlier, a hard task because of the impact of each of these hyper-parameters on the performance and the speed of the training of the model. Therefore, the complexity of the model and the batch size were tested. The complexity of the model was controlled by the number of hidden layers in the network and the size of the convolutional kernel. In fact, more layers induce a more complex model whereas a larger convolutional kernel reduces the complexity of the learning task by providing a higher view over the data.

It is likely that the CNN's architecture is not optimal and can be optimised by an empirical evaluation while testing more hyper-parameters.

7. CONCLUSION

Through this thesis, we analysed patients' deterioration and organ failures in the ICU and aimed to predict them using a machine learning approach. We explored the different existing solutions and added our contribution to the research community with a prognostic model for cardiovascular and pulmonary organs failures prediction.

Our results suggest that the CNN model has substantial additional predictive value for cardiovascular and pulmonary dysfunction among critically ill patients. This, allow clinicians to focus on deteriorating patients and minimise clinical intervention on potentially safe patients.

We believe that this work can be further analysed by investigating the input parameters and values that maximally activate the neurons of the CNN model. Thus, highlighting interesting and clinically-relevant trends in patients data. Another important factor to analyse is the structure of the input data and the impact of a longer input history window of parameters on the performance of the model. Thus, increasing the complexity of the model would probably enable detecting more long and short term dependencies.

BIBLIOGRAPHY

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. Software available from tensorflow.org.

[2] Y. Arabi, N. Al Shirawi, Z. Memish, S. Venkatesh, and A. Al-Shimemeri, *Assessment of six mortality prediction models in patients admitted with severe sepsis and septic shock to the intensive care unit: a prospective cohort study*, Critical care, 7 (2003), p. R116.

[3] Y. Bengio, *Is cross-validation heavily used in deep learning or is it too expensive to be used?* `"https://www.quora.com/Is-cross-validation-heavily-used-in-deep-learning-or-is-it-too-expensive-to-be-used"`, 2016. [Online; accessed 1-October-2017].

[4] J. D. Bronzino, *Biomedical engineering handbook*, vol. 2, CRC press, 1999.

[5] K. L. Caballero Barajas and R. Akella, *Dynamically modeling patient's health state from electronic medical records: A time series approach*, in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, New York, NY, USA, 2015, ACM, pp. 69–78.

[6] Y. Cheng, F. Wang, P. Zhang, and J. Hu, *Risk prediction with electronic health records: A deep learning approach*, in Proceedings of the 2016 SIAM International Conference on Data Mining, SIAM, 2016, pp. 432–440.

[7] F. Chollet et al., *Keras*. `https://github.com/keras-team/keras`, 2015.

[8] M. M. Churpek, T. C. Yuen, S. Y. Park, R. Gibbons, and D. P. Edelson, *Using electronic health record data to develop and validate a prediction model for adverse outcomes on the wards*, Critical care medicine, 42 (2014), p. 841.

[9] M. M. Churpek, T. C. Yuen, C. Winslow, D. O. Meltzer, M. W. Kattan, and D. P. Edelson, *Multicenter comparison of machine learning*

methods and conventional regression for predicting clinical deterioration on the wards, Critical care medicine, 44 (2016), p. 368.

[10] R. Collobert and J. Weston, *A unified architecture for natural language processing: Deep neural networks with multitask learning*, in Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 160–167.

[11] A. Cruz-Roa, A. Basavanhally, F. González, H. Gilmore, M. Feldman, S. Ganesan, N. Shih, J. Tomaszewski, and A. Madabhushi, *Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks*, in Medical Imaging 2014: Digital Pathology, vol. 9041, International Society for Optics and Photonics, 2014, p. 904103.

[12] M. C. Data, *Secondary Analysis of Electronic Health Records*, Springer, 2016.

[13] G. DeepMind, *The first computer program to ever beat a professional player at the game of go*. `https://www.deepmind.com/alpha-go.html`, 2016. Online; accessed 30-october-2017.

[14] T. Desautels, J. Calvert, J. Hoffman, M. Jay, Y. Kerem, L. Shieh, D. Shimabukuro, U. Chettipally, M. D. Feldman, C. Barton, et al., *Prediction of sepsis in the intensive care unit with minimal electronic health record data: a machine learning approach*, JMIR medical informatics, 4 (2016).

[15] A. M. Drewry, B. M. Fuller, T. C. Bailey, and R. S. Hotchkiss, *Body temperature patterns as a predictor of hospital-acquired sepsis in afebrile adult intensive care unit patients: a case-control study*, Critical Care, 17 (2013), p. R200.

[16] F. D'Aragon, E. P. Belley-Cote, M. O. Meade, F. Lauzier, N. K. Adhikari, M. Briel, M. Lalu, S. Kanji, P. Asfar, A. F. Turgeon, et al., *Blood pressure targets for vasopressor therapy: a systematic review*, Shock, 43 (2015), pp. 530–539.

[17] E. A. Feigenbaum, B. G. Buchanan, and J. Lederberg, *On generality and problem solving: A case study using the dendral program*, (1970).

[18] R. A. Fisher, *The use of multiple measurements in taxonomic problems*, Annals of Eugenics, 7 (1936), pp. 179–188.

[19] C. Fleischmann, A. Scherag, N. K. Adhikari, C. S. Hartog, T. Tsaganos, P. Schlattmann, D. C. Angus, and K. Reinhart, *Assessment of global incidence and mortality of hospital-treated sepsis. current*

estimates and limitations, American journal of respiratory and critical care medicine, 193 (2016), pp. 259–272.

[20] J. Friedman, T. Hastie, and R. Tibshirani, *Regularization paths for generalized linear models via coordinate descent*, Journal of statistical software, 33 (2010), p. 1.

[21] M. Ghassemi, T. Naumann, F. Doshi-Velez, N. Brimmer, R. Joshi, A. Rumshisky, and P. Szolovits, *Unfolding physiological state: Mortality modelling in intensive care units*, in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014, pp. 75–84.

[22] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.

[23] P. Golik, P. Doetsch, and H. Ney, *Cross-entropy vs. squared error training: a theoretical and experimental comparison.*, in Interspeech, vol. 13, 2013, pp. 1756–1760.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016. `http://www.deeplearningbook.org`.

[25] R. Hecht-Nielsen, *Neural networks for perception (vol. 2)*, Harcourt Brace & Co., Orlando, FL, USA, 1992, ch. Theory of the Backpropagation Neural Network, pp. 65–93.

[26] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, *Improving neural networks by preventing co-adaptation of feature detectors*, arXiv preprint arXiv:1207.0580, (2012).

[27] D. H. Hubel and T. N. Wiesel, *Receptive fields and functional architecture of monkey striate cortex*, The Journal of physiology, 195 (1968), pp. 215–243.

[28] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, CoRR, abs/1502.03167 (2015).

[29] S. Jones, *How the gpu is revolutionizing machine learning.* `"https://blogs.nvidia.com/blog/2015/12/10/machine-learning-gpu-facebook/"`, 2015. [Online; accessed 28-September-2017].

[30] A. Karpathy, *Cs231n: Convolutional neural networks for visual recognition.* `"http://cs231n.github.io/convolutional-networks/"`, 2018. [Online; accessed 13-January-2018].

[31] G. King and L. Zeng, *Logistic regression in rare events data*, Political analysis, 9 (2001), pp. 137–163.

[32] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).

[33] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, *Exploring strategies for training deep neural networks*, Journal of machine learning research, 10 (2009), pp. 1–40.

[34] J.-R. Le Gall, J. Klar, S. Lemeshow, F. Saulnier, C. Alberti, A. Artigas, and D. Teres, *The logistic organ dysfunction system: a new way to assess organ dysfunction in the intensive care unit*, Jama, 276 (1996), pp. 802–810.

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, 86 (1998), pp. 2278–2324.

[36] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell, *Learning to diagnose with lstm recurrent neural networks*, arXiv preprint arXiv:1511.03677, (2015).

[37] M. C. Matlakala, M. C. Bezuidenhout, and A. D. Botha, *Challenges encountered by critical care unit managers in the large intensive care units*, Curationis, 37 (2014), pp. 1–7.

[38] S. Mitchell, K. Schinkel, Y. Song, Y. Wang, J. Ainsworth, T. Halbert, S. Strong, J. Zhang, C. C. Moore, and L. E. Barnes, *Optimization of sepsis risk assessment for ward patients*, in Systems and Information Engineering Design Symposium (SIEDS), 2016 IEEE, IEEE, 2016, pp. 107–112.

[39] T. Mitchell, *Machine learning*, tech. rep., McGraw-Hill Higher Education, 1997.

[40] G. E. Moore, *Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp. 114 ff.*, IEEE Solid-State Circuits Society Newsletter, 20 (2006), pp. 33–35.

[41] A. P. Nassar, A. O. Mocelin, A. L. B. Nunes, F. P. Giannini, L. Brauer, F. M. Andrade, and C. A. Dias, *Caution when using prognostic models: a prospective comparison of 3 recent prognostic models*, Journal of critical care, 27 (2012), pp. 423–e1.

[42] A. Ng, *Parameters vs hyperparameters, neural networks and deep learning*. `https://www.coursera.org/learn/neural-networks-deep-learning/lecture/TBvb5/parameters-vs-hyperparameters`, 2017. Online; accessed 20-February-2018.

[43] Q. Nguyen, P. Ha, et al., *Cardiovascular and respiratory monitoring in critical nursing care: a literature review*, (2016).

[44] S. Nisula, K.-M. Kaukonen, S. T. Vaara, A.-M. Korhonen, M. Poukkanen, S. Karlsson, M. Haapio, O. Inkinen, I. Parviainen, R. Suojaranta-Ylinen, et al., *Incidence, risk factors and 90-day mortality of patients with acute kidney injury in finnish intensive care units: the finnaki study*, Intensive care medicine, 39 (2013), pp. 420–428.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.

[46] V. Pettilä, M. Pettilä, S. Sarna, P. Voutilainen, and O. Takkunen, *Comparison of multiple organ dysfunction scores in the prediction of hospital mortality in the critically ill*, Critical Care Medicine, 30 (2002), pp. 1705–1711.

[47] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng, *Cardiologist-level arrhythmia detection with convolutional neural networks*, CoRR, abs/1707.01836 (2017).

[48] A. L. Samuel, *Some studies in machine learning using the game of checkers*, IBM Journal of Research and Development, 3 (1959), pp. 210–229.

[49] N. M. Saufl, *2009 national patient safety goals*, Journal of PeriAnesthesia Nursing, 24 (2009), pp. 114–118.

[50] R. M. Schein, N. Hazday, M. Pena, B. H. Ruben, and C. L. Sprung, *Clinical antecedents to in-hospital cardiopulmonary arrest*, Chest, 98 (1990), pp. 1388–1392.

[51] E. H. Shortliffe, *A rule-based computer program for advising physicians regarding antimicrobial therapy selection*, in Proceedings of the 1974 annual ACM conference-Volume 2, ACM, 1974, pp. 739–739.

[52] M. Singer, C. S. Deutschman, C. W. Seymour, M. Shankar-Hari, D. Annane, M. Bauer, R. Bellomo, G. R. Bernard, J.-D. Chiche,

C. M. Coopersmith, et al., *The third international consensus definitions for sepsis and septic shock (sepsis-3)*, Jama, 315 (2016), pp. 801–810.

[53] J. Snoek, H. Larochelle, and R. P. Adams, *Practical bayesian optimization of machine learning algorithms*, in Advances in neural information processing systems, 2012, pp. 2951–2959.

[54] H. Suresh, *Clinical event prediction and understanding with deep neural networks*, Master's thesis, Massachusetts Institute of Technology, 2017.

[55] A. H. Taenzer, J. B. Pyke, and S. P. McGrath, *A review of current and emerging approaches to address failure-to-rescue*, The Journal of the American Society of Anesthesiologists, 115 (2011), pp. 421–431.

[56] C. L. Tsien, H. Fraser, W. J. Long, and R. L. Kennedy, *Using classification tree and logistic regression methods to diagnose myocardial infarction*, Medinfo, 98 (1998).

[57] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, *Wavenet: A generative model for raw audio*, CoRR, abs/1609.03499 (2016).

[58] J.-L. Vincent, R. Moreno, J. Takala, S. Willatts, A. De Mendonça, H. Bruining, C. Reinhart, P. Suter, and L. Thijs, *The sofa (sepsis-related organ failure assessment) score to describe organ dysfunction/failure*, Intensive care medicine, 22 (1996), pp. 707–710.

[59] J.-L. Vincent, D. R. Nelson, and M. D. Williams, *Is worsening multiple organ failure the cause of death in patients with severe sepsis?*, Critical care medicine, 39 (2011), pp. 1050–1055.

[60] F. Wang, N. Lee, J. Hu, J. Sun, and S. Ebadollahi, *Towards heterogeneous temporal clinical event pattern discovery: a convolutional approach*, in Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2012, pp. 453–461.

[61] Wikipedia, *Random forests.* `"https://en.wikipedia.org/wiki/Random_forest"`, 2017. [Online; accessed 25-January-2018].

[62] F. Yang, H.-z. Wang, H. Mi, W.-w. Cai, et al., *Using random forest for reliable classification and cost-sensitive learning for medical diagnosis*, BMC bioinformatics, 10 (2009), p. S22.

[63] S. L. Zanotti-Cavazzoni, R. P. Dellinger, and J. E. Parrillo, *Chapter 26 - severe sepsis and multiple organ dysfunction*, in Critical Care Medicine (Third Edition), J. E. Parrillo and R. P. Dellinger, eds., Mosby, Philadelphia, third edition ed., 2008, pp. 467 – 484.

[64] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, *Time series classification using multi-channels deep convolutional neural networks*, in International Conference on Web-Age Information Management, Springer, 2014, pp. 298–310.