



Degree Project in Medical Engineering

Second Cycle, 30 Credits

# Deep Learning for Continuous Time Series of Clinical Waveform Data

Development of a clinical decision support system for predicting mortality in Covid-19 patients

**CAROLIN DANKER**

# **Deep Learning for Continuous Time Series of Clinical Waveform Data**

**Development of a clinical decision support system for predicting mortality in Covid-19 patients**

CAROLIN DANKER

Master's Programme, Medical Engineering, 120 credits  
Date: June 21, 2022

Supervisors: Ricardo Vinuesa (KTH), Eric Herlenius (KI)

Examiner: Matilda Larsson

Reviewer: Jayanth Raghothama

In cooperation with Karolinska Institute

Swedish title: Djupinlärning för kontinuerlig klinisk vågformsdata

Swedish subtitle: Utveckling av ett verktyg för kliniskt beslutsfattande gällande prognoser av dödlighet bland Covid-19 patienter

Royal Institute of Technology  
Kungliga Tekniska Högskolan  
KTH STH  
SE-141 86 Flemingsberg, Sweden  
<http://www.kth.se/sth>



## **Abstract**

The coronavirus disease 19 (Covid-19) pandemic led to challenging decisions about allocating limited resources. Common methods for assessing the health status of patients were not reliable enough. Prediction of clinical events has recently become a focus of research, and several models have been developed, often requiring various inputs. This thesis deals with the question of to what extent commonly recorded high-frequency waveform data can be used for prediction. Two new Covid-19 datasets with 105 and 112 patients, respectively, were created using recorded data from Karolinska Hospital. Three different model architectures were then implemented and trained, varying hyperparameters and input features. A balanced accuracy of over 0.75 and an F1-score of over 0.60 was achieved with all architectures, however, with high standard deviations. The use of demographic information could improve the results significantly. The thesis shows that predictions are generally possible, but further experiments with a larger data set are required.

## **Keywords**

Deep Learning, Mortality Prediction, Time Series Data, Waveform Data, Covid-19

## Sammanfattning

Pandemin till följd av Coronaviruset (Covid-19) har lett till svåra beslut gällande fördelning av begränsade resurser, och allmänna metoder för bedömning av patienters hälsa har inte varit tillräckligt tillförlitliga. Prognoser av kliniska händelser har nyligen fått större fokus inom forskning och flera modeller har utvecklats, metoder som ofta kräver olika mätvärden. Således bemöter detta arbete frågan till vilken utsträckning det är möjligt att använda högfrekventa vågformer inom prognosberäkning. Två nya Covid-19 dataset med 105 respektive 112 patienter var, skapades av mätdata från Karolinska Sjukhuset. Tre olika model-arkitekturer var implementerade och tränade, med olika hyper-parametrar och inputdata. En balanserad träffsäkerhet över 0.75 och ett F1-mått över 0.60 uppmättes med alla modeller, dock med hög standardavvikelse. Användning av demografisk information skulle kunna förbättra resultatet markant. Detta arbete visar att prognoser är möjliga, men att vidare experiment med större datamängder behövs genomföras.

## Nyckelord

Djupinlärning, dödlighetsprognos, tidsseriedata, vågformsdata, Covid-19

## Acknowledgments

First, I would like to thank my supervisors Ricardo Vinuesa and Eric Herlenius for giving me the opportunity to work on this project and for guiding and supporting me throughout the entire time. I would also like to thank Antoine Honoré for supervising me and for the technical setup and support, especially when working with the database and the server. I would also like to thank Marat Murzabekov, Niklas Lidströmer, Susanne Rautiainen, and Rami Asfar from Karolinska Institute for their collaboration. Special thanks to Nuria Ortigosa Araque from the faculty at the Polytechnic University of Valencia for providing expertise in the preprocessing of physiological data and applying it to deep learning. Last, I want to thank Henrik Siren for the continuous exchange and the enriching discussions throughout our time together at CMM, and to my supervisor group led by Maksims Kornevs for the suggestions and comments.

Stockholm, June 2022

Carolin Danker

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Methodology</b>	<b>3</b>
2.1	Data Preparation . . . . .	3
2.1.1	Data Collection . . . . .	3
2.1.2	Preparation of High-Frequency Data . . . . .	4
2.1.3	Data Splitting and Normalization . . . . .	5
2.1.4	Data Augmentation . . . . .	5
2.1.5	Labeling . . . . .	6
2.2	Model Development . . . . .	6
2.3	Evaluation Metrics . . . . .	7
2.4	Experiment Device . . . . .	8
<b>3</b>	<b>Experimental Design</b>	<b>9</b>
3.1	Model Architectures . . . . .	9
3.1.1	Vanilla CNN . . . . .	9
3.1.2	CNN Inception . . . . .	10
3.1.3	CNN-LSTM . . . . .	11
3.2	Training Hyperparameters . . . . .	12
3.2.1	Learning Rate . . . . .	12
3.2.2	Training Loss . . . . .	13
3.2.3	Fixed Training Hyperparameters . . . . .	13
3.3	Experiments . . . . .	13
3.3.1	Hyperparameter Study . . . . .	13
3.3.2	Input Data Study . . . . .	15
<b>4</b>	<b>Results</b>	<b>16</b>
4.1	Study Population . . . . .	16
4.2	Hyperparameter Study . . . . .	18
4.2.1	Vanilla CNN . . . . .	19
4.2.2	CNN Inception . . . . .	21
4.2.3	CNN-LSTM . . . . .	23

4.2.4	Comparison . . . . .	26
4.3	Input Data Study . . . . .	27
4.3.1	Input Features . . . . .	27
4.3.2	Dataset . . . . .	27
<b>5</b>	<b>Discussion</b>	<b>30</b>
5.1	Data Processing and Study Population . . . . .	30
5.2	Model Development and Performance . . . . .	31
5.3	Comparison with Previous Results . . . . .	32
5.4	Explainability . . . . .	33
5.5	Other Limitations . . . . .	34
5.6	Future Work . . . . .	34
<b>6</b>	<b>Conclusion</b>	<b>35</b>
	<b>References</b>	<b>36</b>
<b>A</b>	<b>State of The Art</b>	<b>45</b>
A.1	Clinical Data . . . . .	45
A.1.1	Data Types and Sources . . . . .	45
A.1.2	Clinical Decision Support Systems . . . . .	46
A.2	Physiological Monitoring . . . . .	46
A.2.1	Vital Signs . . . . .	46
A.2.2	Continuous Physiological Monitoring . . . . .	47
A.2.3	Patient Deterioration . . . . .	49
A.3	Machine Learning for Clinical Data . . . . .	50
A.3.1	Machine Learning Approaches . . . . .	50
A.3.2	Deep Learning Approaches . . . . .	50
A.3.3	Explainability of Algorithms . . . . .	51
A.4	Prediction Models . . . . .	52
A.4.1	General Prediction Scores . . . . .	52
A.4.2	Prediction Models for Covid-19 . . . . .	52
A.4.3	Mortality Prediction Models . . . . .	54
A.4.4	Other relevant studies . . . . .	55
<b>B</b>	<b>Model Summaries</b>	<b>57</b>
<b>C</b>	<b>Additional Results</b>	<b>60</b>
C.1	Vanilla CNN . . . . .	60
C.2	CNN Inception . . . . .	63
C.3	CNN-LSTM . . . . .	66

# List of Figures

2.1	Exemplary timeline for a patient . . . . .	4
2.2	Filtering of physiological signals . . . . .	5
2.3	Data augmentation techniques . . . . .	6
2.4	Confusion matrix and evaluation metrics . . . . .	7
3.1	Vanilla CNN architecture . . . . .	10
3.2	CNN Inception architecture . . . . .	11
3.3	CNN-LSTM architecture . . . . .	12
3.4	Step scheduler for learning Rate . . . . .	12
4.1	Study population . . . . .	17
4.2	Demographics of study population . . . . .	18
4.3	Vanilla CNN Experiment 1 loss and balanced accuracies . . . . .	19
4.4	Vanilla CNN Experiment 1 loss curves . . . . .	19
4.5	Vanilla CNN Experiment 1 ensemble evaluation . . . . .	20
4.6	Vanilla CNN Experiment 2 ensemble balanced accuracies . . . . .	21
4.7	CNN Inception Experiment 1 loss and balanced accuracies . . . . .	21
4.8	CNN Inception Experiment 1 ensemble evaluation . . . . .	22
4.9	CNN Inception Experiment 2 ensemble balanced accuracies . . . . .	23
4.10	CNN-LSTM Experiment 1 loss and balanced accuracies . . . . .	24
4.11	CNN-LSTM Experiment 1 loss curves . . . . .	24
4.12	CNN-LSTM Experiment 1 ensemble evaluation . . . . .	25
4.13	CNN-LSTM Experiment 2 ensemble balanced accuracies . . . . .	25
4.14	Vanilla CNN variation of input signals . . . . .	27
4.15	Vanilla CNN comparison of datasets and augmentation techniques . . . . .	28
4.16	CNN-LSTM comparison of datasets and augmentation techniques . . . . .	29
A.1	Electrode placement and lead perspectives of ECG . . . . .	48
A.2	ECG wave morphology . . . . .	48
A.3	PPG wave morphology . . . . .	49
A.4	ANN and DNN architecture . . . . .	50
A.5	CNN architecture . . . . .	51

A.6	RNN, LSTM, and GRU cell architectures . . . . .	51
B.1	Model summary of Vanilla CNN . . . . .	58
B.2	Model summary of CNN Inception . . . . .	58
B.3	Model summary of CNN-LSTM . . . . .	59
C.1	Model hyperparameter analysis for Vanilla CNN Experiment 1 . . . . .	61
C.2	Model hyperparameter analysis for CNN Inception Experiment 1 . . . . .	64
C.3	Model hyperparameter analysis for CNN-LSTM Experiment 1 . . . . .	67

# List of Tables

3.1	Variation of model hyperparameters for Vanilla CNN Experiment 1. . . .	14
3.2	Variation of model hyperparameters for CNN Inception Experiment 1. . .	14
3.3	Variation of model hyperparameters for CNN-LSTM Experiment 1. . . .	14
4.1	Availability of high-frequency signals . . . . .	16
4.2	Overview of datasets . . . . .	18
4.3	Vanilla CNN Experiment 1 best performing configurations . . . . .	20
4.4	Vanilla CNN Experiment 2 best performing configurations . . . . .	21
4.5	CNN Inception Experiment 1 best performing configurations . . . . .	22
4.6	CNN Inception Experiment 2 best performing configurations . . . . .	23
4.7	CNN-LSTM Experiment 1 best performing configurations . . . . .	25
4.8	CNN-LSTM Experiment 2 best performing configurations . . . . .	26
4.9	Comparison of best performing configurations . . . . .	26
4.10	Evaluation of Vanilla CNN for different datasets and augmentation techniques . . . . .	28
4.11	Evaluation of CNN-LSTM for different datasets and augmentation techniques . . . . .	29
5.1	Comparison with previous results . . . . .	33
A.1	Relevant studies for deep learning in Covid-19 populations to predict mortality . . . . .	53
A.2	Relevant studies for mortality prediction using time-series data . . . . .	56
C.1	Vanilla CNN Experiment 1 results . . . . .	62
C.2	Vanilla CNN Experiment 2 results . . . . .	63
C.3	CNN Inception Experiment 1 results . . . . .	65
C.4	CNN Inception Experiment 2 results . . . . .	66
C.5	CNN-LSTM Experiment 1 results . . . . .	68
C.6	CNN-LSTM Experiment 2 results . . . . .	69

## List of acronyms and abbreviations

1D	One dimensional
2D	Two dimensional
ACC	Accuracy
ANN	Artificial Neural Network
APACHE	Acute Physiology Score
AUC	Area under the curve
BACC	Balanced Accuracy
BCE	Binary Cross Entropy
BN	Batch Normalization
BP	Blood Pressure
CDSS	Clinical Decision Support Systems
CNN	Convolutional Neural Network
Covid-19	coronavirus disease 19
CPU	Central Processing Unit
CT	Computed Tomography
DBP	Diastolic blood pressure
DL	Deep Learning
DNN	Deep Neural Network
DS	Dataset
DT	Decision Tree
ECG	Electrocardiogram
EHR	Electronic Health Records
EMR	Electronic Medical Records
GB	Gigabyte
GPU	Graphics Processing Unit
GRU	Gated recurrent unit
HMM	Hidden-Markow Models
HP	Hewlett-Packard
HR	Heart Rate

Hz	Hertz
ICD	International Statistical Classification of Diseases and Related Health Problems
ICU	Intensive Care unit
iLR	initial Learning Rate
k-NN	k-nearest neighbours
lab	laboratory
LoR	Logistic Regression
LR	Learning Rate
LSTM	Long short-term memory
MAP	Mean arterial pressure
ML	Machine Learning
MPM	Mortality Probability Model
plet	Photoplethysmogram pulse wave
PPG	Photoplethysmogram
PRE	Precision
ReLu	Rectified Linear Activation Function
RF	Random Forest
RNN	Recurrent Neural Network
RR	Respiratory Rate
SAPS	Simplified Acute Physiology Score
SARS-CoV-2	Severe acute respiratory syndrome coronavirus 2
SBP	Systolic blood pressure
SEN	Sensitivity
SHAP	Shapley additive explanation
SOFA	Sequential Organ Failure Assessment
SPE	Specificity
SpO2	Peripheral oxygen saturation
XGBoost	Extreme gradient boosting

# Chapter 1

## Introduction

Over the past two and a half years, coronavirus disease 19 (Covid-19) has been one of the greatest challenges for healthcare systems across the world. In contrast to the normal state, physicians and other health professionals had to deal with limited resources [1]. The situation was complicated by the fact that the disease was unknown, the diverse and unspecific disease course, and only limited available treatment options [2]. This led to difficult decisions about who should receive an intensive care unit (ICU) bed or mechanical ventilation incorporating both ethical and practical considerations and placed great responsibility and stress on decision makers [3, 4]. Common methods of assessing the health status of Covid -19 patients might support the physicians in the first place, but they were not always reliable enough for predicting long-term outcomes and are not accurately tailored to the disease [5]. Therefore, to better assist decision makers and to provide the right care to everyone, and effectively allocate limited resources, it is crucial to be able to predict clinical events such as deterioration, the need for ventilation, or mortality.

To support physicians in the decision-making process, scores using logistic regression exist to evaluate the patients' health state and to predict clinical events [6–10]. In addition, machine learning and deep learning methods can be applied. Generally, artificial intelligence has been shown to have great potential for solving the problems of today's world. With regard to the Sustainable Development Goals of the United Nations, it shows that in the field of medicine, the positive effects of artificial intelligence outweigh the negative ones [11].

Previously developed models aim to predict sepsis [12–17] or mortality [18–25]. The models are based on conventional time-series data from electronic health records, such as laboratory values and vital signs. Some research also focused on decision-support systems specifically for Covid-19 patients. For example, Li et al. developed a model to identify risk factors and predict ICU admission and mortality [26]. The model of Ryan et al. predicts mortality and the need for mechanical ventilation [27]. Many more models were proposed, mainly to predict mortality and to identify risk factors (see reviews [28, 29]).

In general, the prediction task can be considered as a classification of time-series data.

A variety of deep learning model architectures have been proposed over the last couple of years [30]. Common architectures are convolutional neural networks and recurrent neural networks and variations and combinations of these. But also new architectures like Transformer models show promise [31].

Although many studies yielded good results for mortality prediction, there are a few fundamental drawbacks. Most models require several input parameters, including extensive laboratory tests and manually measured vital signs. In a time-critical situation such as during the Covid-19 pandemic, this can be a potential challenge. Furthermore, the timing of the prediction and the amount of data needed are not always useful and suitable for the situation. To customize the treatment early, it would be optimal if the prediction could be made as quickly as possible after admission and with as minimal data as possible.

Motivated by this, the goal of this thesis is the development of deep learning methods to predict clinical outcomes using commonly available time-series data of vital parameters. Unlike most other models, this work utilizes continuous high-frequency monitored data to make predictions as easy as possible. The main focus is on the feasibility of the models with particular emphasis on clinical relevance. For this purpose, a new dataset containing only Covid-19 patients will be used. Based on the objective of the thesis, the primary research question is:

- **Is it possible to predict mortality based on high-frequency monitoring data?**

Several sub-questions emerge from the primary research question:

1. How can a clinically relevant data set be formed from a small sample of patients?
2. Which model architecture is suited best in terms of balanced accuracy and F1-score?
3. What is the minimum amount of data in terms of number of features and length needed to make a stable prediction?

The thesis is divided into five parts. In chapter 2, the methodology used to prepare the dataset, develop the models, and evaluate the results is described. In chapter 3, the proposed model architectures and experimental study design are explained. Following this, the results of the dataset preparation and experiments to answer the research questions are reported (chapter 4). The thesis ends with a discussion (chapter 5) and conclusion (chapter 6).

# Chapter 2

## Methodology

The methodology was based on an extensive literature research on similar tasks (Appendix A). The thesis follows the general approach of an experimental research design in machine learning as described in Kamiri and Mariga [32]: data collection and data preprocessing (section 2.1), model training and testing (section 2.2), and model evaluation (section 2.3). An overview of the used technical environment is given in section 2.4.

### 2.1 Data Preparation

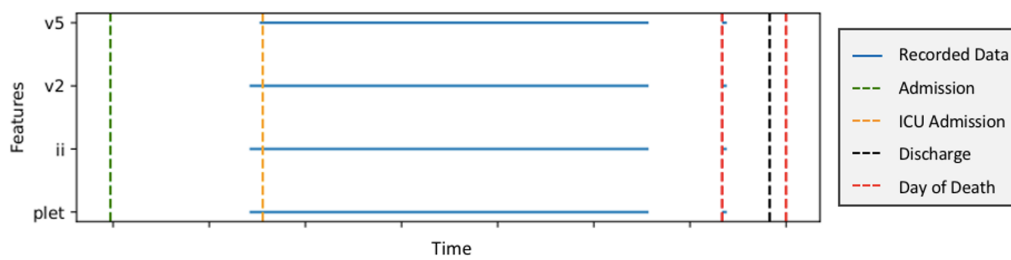
This thesis concentrates on working with data from patients with coronavirus disease 19 Covid-19. The data preparation can be divided into data collection (subsection 2.1.1), data preparation (subsection 2.1.2), and data splitting and normalization (subsection 2.1.3). Further, data augmentation techniques were developed (subsection 2.1.4) and the data was labelled (subsection 2.1.5).

#### 2.1.1 Data Collection

The data has been collected prospectively from five hospital wards receiving Covid-19 positive patients at Karolinska Hospital. The data from all patients entering the wards were automatically stored and then filtered for ICD codes U07.1 and U07.2. U07.1 is used for patients with the diagnosis Covid-19 and a confirmed infection by a laboratory test; U07.2 is used if the virus has not been identified but diagnosed clinically [33]. The data was collected between March 2020 and June 2021.

The patients were continuously monitored with mainly two modalities recording high-frequency dynamic data: electrodes providing the electrocardiogram (ECG) and a pulse oximeter providing a photoplethysmogram (PPG). Each of these signals was stored separately for each patient in a compressed format. Other available data from the electronic health records (EHR) include static demographic information, e.g. sex or age at admission, and time-varying data such as lab results or manually sampled blood pressure.

In addition to the time-varying monitoring and demographics data, a clinical timeline of patient admission and discharge was constructed. This included admission times, discharge times, the times of admission to Intensive care units (ICUs), and potentially the day of death. Figure 2.1 shows an example of a patient timeline. Based on the timestamps, the relevant part of the data was chosen for the study. In the cases when patients were admitted for several stays, only the data of one stay was considered. Data before the date of admission and data on the day of death of deceased patients was excluded. Patients who died later than three days after discharge were excluded from the study.



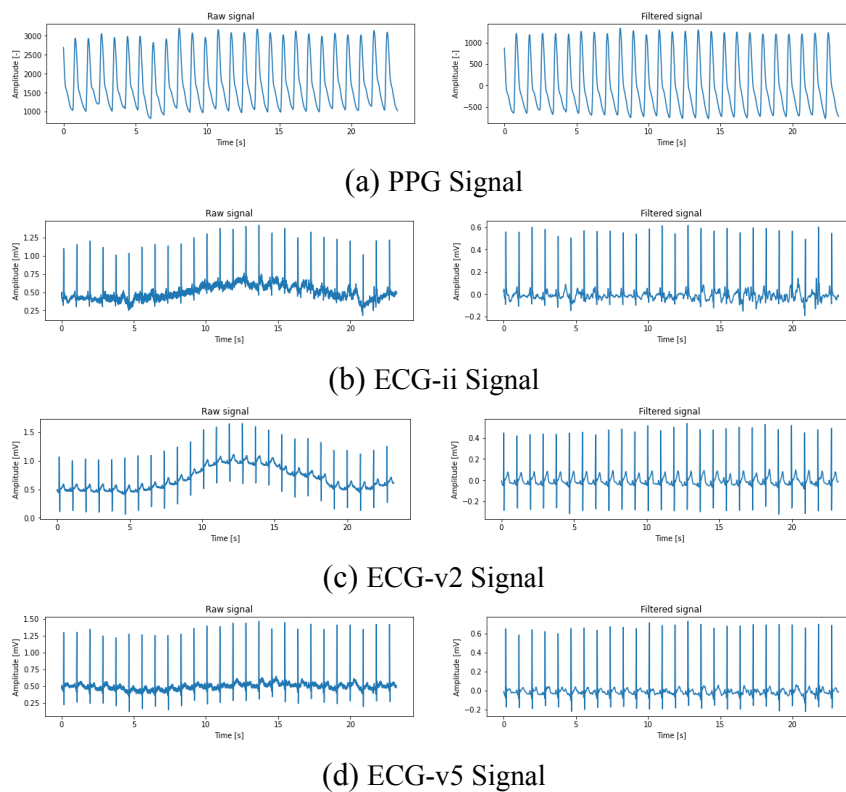
**Figure 2.1.** Exemplary timeline for a patient.

## 2.1.2 Preparation of High-Frequency Data

The diverse patient situations and hospitalization wards in the cohort lead the amount of data to vary across patients. It was identified which four signals were occurring the most, by counting for how patients the signal was stored. Patients without all four features available were excluded. The further preparation included time-series extraction, signal filtering, and resampling of the signals to the same frequency.

Time-series extraction was performed iteratively to select the earliest possible time interval in which all four physiological signals were available. Different data sets were created depending on the length of the recorded data  $n_{original}$  and the maximum allowed gap length  $g$  during which data was missing. During this step, it was also checked whether the recorded data was actual physiological data by calculating the variance of the data. If the variance was below a threshold, it was assumed that the signal was empty. If no time interval with the given length was found, the patient was excluded.

The extracted raw physiological signals were filtered to remove common noise sources. First, a high-pass filter was applied to remove baseline wanders due to motion resulting in frequencies below 0.5 Hz [34]. Second, a 50 Hz Notch filter was used to remove powerlines causing electromagnetic fields that lead to sinusoidal disturbances [34]. And third, a 50 Hz low-pass filter to remove high-frequency noise due to muscle movements and to avoid aliasing when resampling was applied [35]. The parameters of the filters were chosen to affect the wave morphology as little as possible while removing as much noise as possible. Figure 2.2 shows the four signals before and after filtering. Finally, the signal was resampled at a frequency of 125 Hz.



**Figure 2.2.** Raw (left) and filtered (rights) signals of PPG (a) and ECG (b-d).

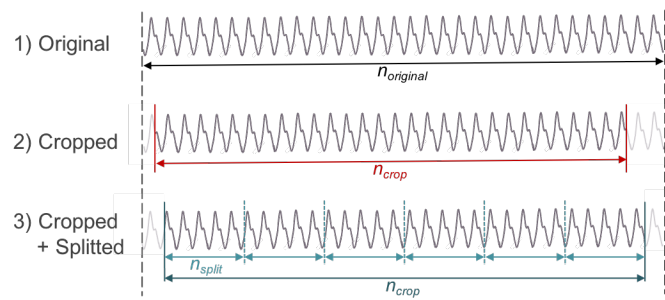
### 2.1.3 Data Splitting and Normalization

The dataset was split on a patient basis into two subsets, Training & Validation Subset and Testing Subset. 10% of the data was randomly chosen for the Testing Subset, whereby the ratio between the two classes had to approximately match the ratio in the entire data set  $\pm 10\%$ . In addition, K-fold cross-validation with eight folds was performed during training, splitting the subset into a Training Subset and a Validation Subset.

Eventually, the data was normalized. The Testing Subset was normalized using the mean and standard deviation of the entire Training & Validation Subset. The training and validation data was normalized for every fold using just the data from the Training Subset, thus, excluding the validation data.

### 2.1.4 Data Augmentation

Since only a small data set was used, data augmentation was used to train the models with slightly different time series in each epoch. Two data augmentation techniques were developed as shown in Figure 2.3. With the first technique, the signal was randomly cropped on both ends. With the second technique, the signal was additionally split into several pieces resulting in multiple sequences for each patient.



**Figure 2.3.** Schematic representation of data augmentation techniques. Top: Original signal with length  $n_{original}$ . Middle: Randomly cropping beginning and end of signal resulting in a signal length of  $n_{crop} < n_{original}$ . Bottom: Randomly cropping beginning and end of signal and split signal in  $k$  parts of length  $n_{split} = n_{crop}/k$ .

### 2.1.5 Labeling

The model aims to solve a classification task of whether the patient died from Covid-19 or not. Therefore, binary labels were used. Patients who died before or within three days of discharge are labeled as diseased (positive class), while the rest was labeled as survived (negative class). The temporal aspect is not considered, which means that only the final result of the stay is considered and not how long the period is between the prediction and the event.

## 2.2 Model Development

As mentioned above, the thesis followed an experimental research design. The model architectures were chosen based on the literature review for models solving similar tasks. In this work, only deep learning methods were considered. Deep learning methods have recently becoming very popular because of the large amounts of data when working with medical data, including time-series data from sensors and health records [36]. The used input data consists of several high-frequency time-series data and static features. Conventional machine learning methods require manual extraction of features from time series data, whereas deep learning methods automatically create the features that best fit the data and the task best [37]. This is beneficial both in terms of better use of the data and by the fact that the data can be used directly, without an additional step of pre-processing.

The model development phase was divided into two parts. First, experiments for finding a suitable model architecture and training hyperparameters were performed. In the second part, the input data was varied.

## 2.3 Evaluation Metrics

Every hyperparameter configuration was run five times to reduce arbitrary deviation. The model performances were evaluated using different metrics based on the confusion matrix (Figure 2.4):

- **Sensitivity (SEN):** Measures ability to predict mortality.
- **Specificity (SPE):** Measures how good survival is predicted.
- **Balanced Accuracy (BACC):** Average of sensitivity and specificity.
- **Precision (PRE):** Measures how accurate the mortality predictions are.
- **F1 Score:** Harmonic mean of sensitivity and precision:

$$F1 = \frac{2 \cdot SEN \cdot PRE}{SEN + PRE}$$

The metrics were calculated on the testing, validation, and testing data for each fold individually. To compare the models mean and standard deviation were calculated over all folds and runs. Additionally, an ensemble was built using all models of one run. Ensemble predictions were made using majority voting. The same metrics as above were used and averaged over all runs of one configuration. Due to the high imbalance between the two classes, the balanced accuracy is used instead of the regular accuracy and was the main measure to compare models in this work.

		Prediction		
		Deceased ("1")	Survived ("0")	
Truth	Deceased ("1")	True Positive (TP)	False Negative (FN)	<b>Sensitivity (SEN)</b> TP / (TP+FN)
	Survived ("0")	False Positive (FP)	True Negative (TN)	<b>Specificity (SPE)</b> TN / (TN+FP)
		<b>Precision</b> TP / (TP+FP)		<b>Balanced Accuracy</b> SEN+SPE / 2

**Figure 2.4.** Confusion matrix and Evaluation metrics that are used in this report.

In order to evaluate and assess the performance of a model, it is first compared with other in this work developed models to find the optimal configuration. Since the focus of the work is on the general feasibility of the method and to see if such a model trains at all, this was the priority during the evaluation. In addition, the results were compared with other work to see how similar models performed. A baseline model based on human expertise was not used due to feasibility and time constraints.

## 2.4 Experiment Device

Data preparation, model development, and evaluation were performed in Python 3.7 [38]. PyTorch was used to implement the deep learning models [39]. Most tasks were carried out on an HP laptop with a 16 GB Intel core. For model training, a Computation Server with 80 CPU cores and 4 Tesla v100-SXM2-16GB GPUs was available but had to be shared with other projects. For training and testing the models, a test environment with a Singularity container was created [40].

# Chapter 3

## Experimental Design

This chapter contains an overview of the chosen model architectures (section 3.1), training hyperparameters (section 3.2) and experiments (section 3.3).

### 3.1 Model Architectures

Three deep learning architectures were selected based on similar work for processing physiological time-series data. The architectures were initially implemented as specified in the literature and then adapted to the given dataset and task. In the following, similar parts of the model architectures are described only once. Exemplary model summaries for each architecture can be found in Appendix B.

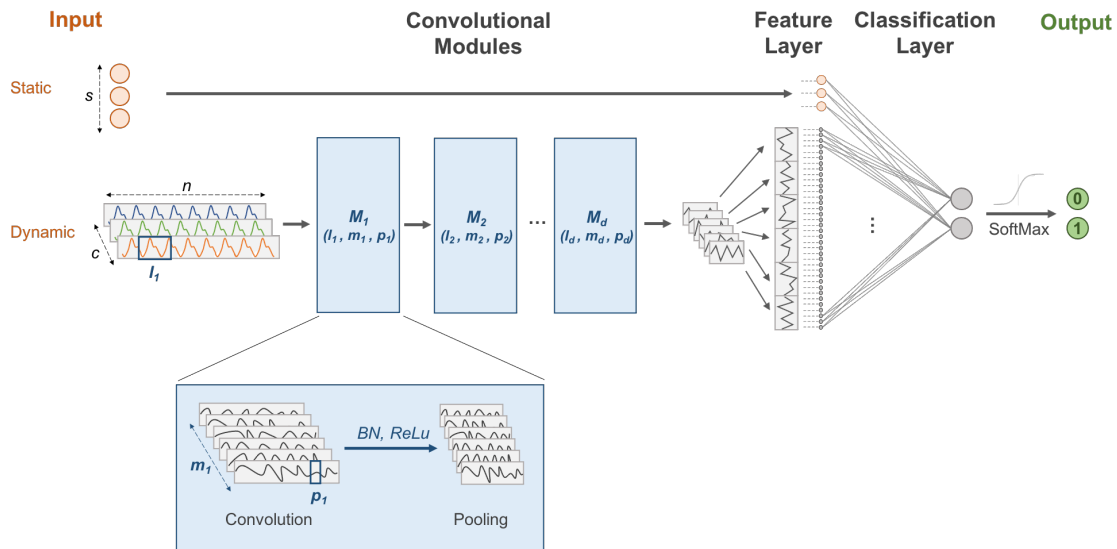
#### 3.1.1 Vanilla CNN

The first model is a Vanilla convolutional neural network (CNN). The design is based on the work of Zhao et al. who introduced a methodology for CNNs for time-series classification [41]. The model architecture is shown in Figure 3.1.

**Input:** The input consists of dynamic and static features. The dynamic features are represented as an input vector of size  $n \times c$ , with signal length  $n$  and number of channels  $c$ . The static features are represented as an input vector of size  $1 \times s$ , with number of static features  $s$ .

**Convolutional Modules:** Convolution operations are performed on the dynamic features. Every convolutional module consists of one convolutional layer followed by batch normalization (BN), a rectified linear activation function (ReLU), and a pooling layer. The convolutional layer can vary in filter numbers  $m$  and filter size  $l$ . Max-pooling is used and the pooling size  $p$  can vary. The depth of the model  $d$  determines how many convolutional modules are stacked in series.

**Feature and Classification Layer:** After the last convolutional layer, the output vector is flattened to a 1D-feature map and concatenated with static features. The final binary classification is performed using a fully connected layer followed by a soft-max layer.



**Figure 3.1.** Vanilla CNN architecture.

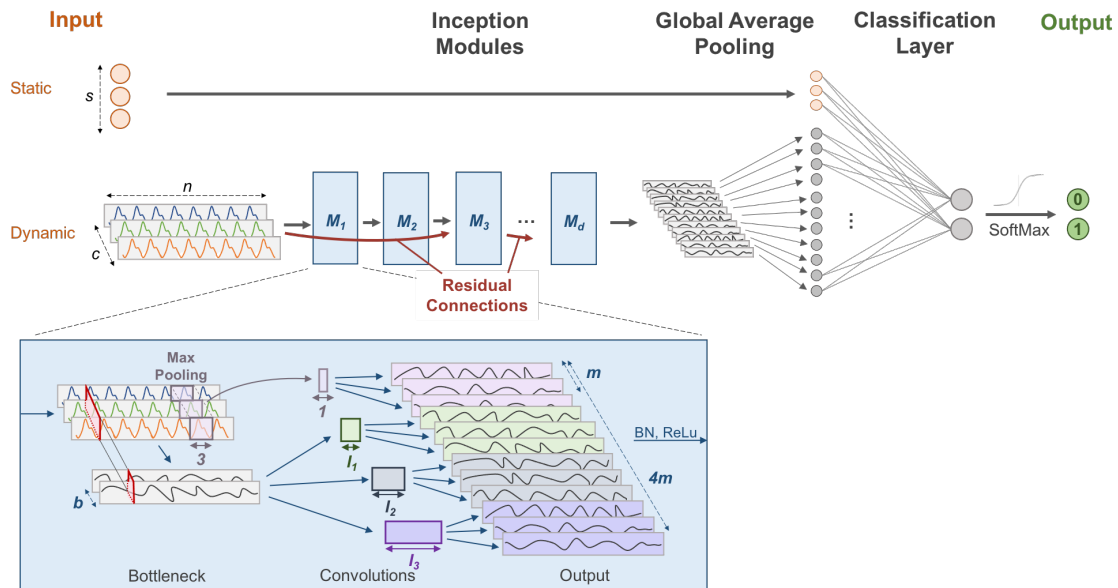
### 3.1.2 CNN Inception

The second model is based on the CNN Inception model for time series from Ismail Fawaz et al. [42]. Instead of applying several convolutions in series, they are applied in parallel and combined with a bottleneck layer. Figure 3.2 shows the architecture.

**Input:** As before, the input includes dynamic features, with signal length  $n$  and number of channels  $c_0$ , and  $s$  static features of length 1.

**Inception Modules:** Instead of convolutional modules, this architecture consists of inception modules to process the dynamic features. First, a bottleneck layer adapts the size of the input channels  $c_i$  to size  $c_b$ . Usually, it is  $c_b < c_i$  as the goal of the bottleneck layer is to reduce the number of parameters of the model. Three convolutions are performed on the output of the bottleneck layer with length  $l_1$ ,  $l_2$  and  $l_3$ , respectively. Additionally, max pooling with a pooling length of 3 is performed on the input of the module, followed by a convolutional layer with length 1. The output channels of all four convolutions are of size  $m$ , resulting in an overall output size of the module of  $4m \times n$ . Last, BN and ReLU are applied. The depth of the model  $d$  determines how many inception modules are stacked in series. Moreover, every third module is connected by a residual connection.

**Global average pooling and classification:** The binary classification is performed using a fully-connected layer. Unlike the Vanilla CNN, the output of the last inception module undergoes a global average pooling before the classification layer. This results in an input size for the fully-connected layer of  $4m \times 1$ . The fully-connected layer is followed by a soft-max layer to calculate the probability for both classes.



**Figure 3.2.** CNN Inception architecture.

### 3.1.3 CNN-LSTM

The third proposed model combines a CNN with a recurrent neural network (RNN), more specifically, a long short-term memory architecture (LSTM). The input and the convolutional layer are equivalent to the Vanilla CNN (subsection 3.1.1). After that, an LSTM and a classification layer are connected. Figure 3.3 depicts the architecture.

**LSTM Layer:** The output of the CNN is combined with the static features by extending the static features to the same length  $i$  as the CNN output. This results in  $i$  input vectors  $x_1, x_2, \dots, x_i$ . The LSTM consists of  $n$  nodes and  $h$  hidden layers. The LSTM is of the kind "many to one" and, thus, produces one output  $y$  of size  $1 \times n$ .

**Classification Layer:** The final binary classification is performed using a fully connected layer and a soft-max layer.

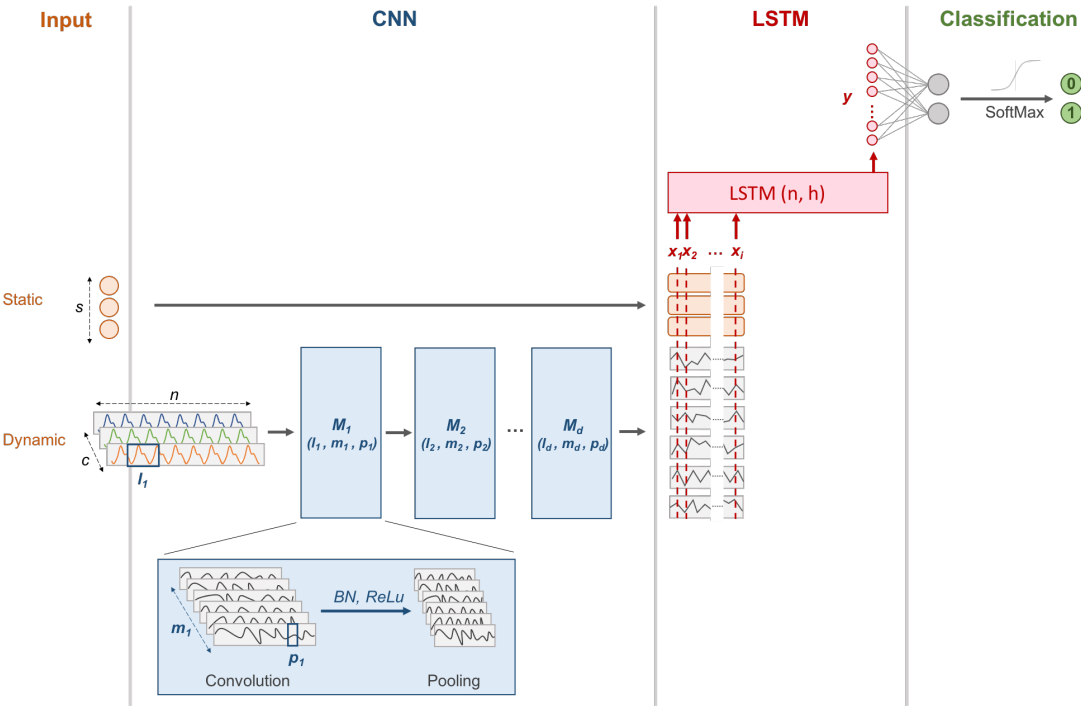


Figure 3.3. CNN-LSTM architecture.

### 3.2 Training Hyperparameters

The training hyperparameters include fixed parameters and three varying parameters: initial learning rate (iLR), step scheduler, and L1 Regularization.

#### 3.2.1 Learning Rate

All models were trained with an Adam optimizer [43]. In addition to the automatic adaption of the iLR through the Adam optimizer, the upper limit for the learning rate (LR) is given by a step scheduler (Figure 3.4). The upper limit of the LR is decreased by  $\gamma$  after a specified number of epochs  $e = 100$ .

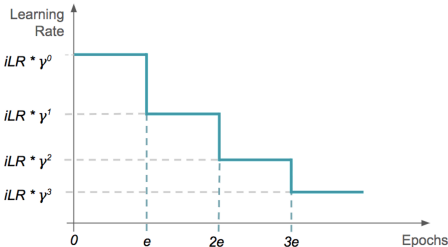


Figure 3.4. Step scheduler for learning rate. The upper limit of the learning rate is reduced every  $e$  epochs by a factor  $\gamma$ , starting from a given initial learnign rate (iLR).

### 3.2.2 Training Loss

The loss was calculated using Binary Cross Entropy (BCE) depending on labels  $y$ , predictions  $\hat{y}$  and weights  $w_y$ . Weights are used to account for unbalance between classes. Additionally, L1 regularization (Lasso Regression) was added to the loss. The L1 regularization term is calculated by summing up the absolute value of all model weights  $w_{p,i}$  with  $i = 1, 2, \dots, P$  and  $P$  is the number of model parameters, and multiplying it with  $\lambda$  to control the regularization. The loss was therefore calculated as:

$$loss = BCE(y, \hat{y}, w_y) + \lambda \sum_{i=1}^P |w_{p,i}|$$

### 3.2.3 Fixed Training Hyperparameters

Early stopping was used. The training stopped when the validation loss decreased for 15 epochs. This means that a minimum validation loss was reached and no lower loss was reached within 15 epochs; it does not mean that the loss increased 15 times in series. All models were trained for a maximum of 1000 epochs. The batch size was set to 15 patients.

## 3.3 Experiments

The experiments were divided into two parts, a hyperparameters study (subsection 3.3.1) and an input data study (subsection 3.3.2).

### 3.3.1 Hyperparameter Study

All model architectures were developed with regard to model hyperparameters and training hyperparameters. It should be noted that this work focused on feasibility and not on identifying the optimal configuration. First, a random search was performed to find a good parameter range before a grid search was used for a more detailed evaluation. All experiments were performed with all signals available for the study population.

#### Experiment 1: Model Hyperparameters

For the Vanilla CNN, the hyperparameters include depth  $d$ , filter sizes  $l_i$ , number of filter per convolutional layer  $m_i$ , pooling sizes  $p_i$  and depth, resulting in 54 configurations (see Table 3.1). All models were trained with an iLR of 0.0005, no scheduler ( $\gamma = 1$ ) and no L1 regularization ( $\lambda = 0$ ).

**Table 3.1.** Variation of model hyperparameters for Vanilla CNN Experiment 1.

Feature	Variable	Variations
Depth	$d$	3, 4
Number of Filters	$m_i, i = 1, 2, \dots, d$	2, 5, 8
Filter Size	$l_i, i = 1, 2, \dots, d$	5, 10, 15
Pooling Size	$p_i, i = 1, 2, \dots, d$	5, 7, 10

For the CNN Inception, 36 configurations were tested adapting the depth  $d$ , number of filters  $m$ , bottleneck size  $c_b$  and filter sizes  $l$  (Table 3.2). The training hyperparameters were set to an iLR of 0.0005,  $\gamma = 1$  and  $\lambda = 0$ .

**Table 3.2.** Variation of model hyperparameters for CNN Inception Experiment 1.

Feature	Variable	Variations
Depth	$d$	1, 3
Number of Filters	$m$	2, 5, 10
Bottleneck Size	$c_b$	1, 3
Filter Size	$l = [l_1, l_2, l_3]$	[100, 50, 25], [20, 10, 5], [40, 20, 10]

For the CNN-LSTM model, six model hyperparameters were varied, four for the CNN and two for the LSTM resulting in 48 different configurations (Table 3.3). For the training an iLR of 0.0002,  $\gamma = 0.85$ , and  $\lambda = 0$  was used.

**Table 3.3.** Variation of model hyperparameters for CNN-LSTM Experiment 1.

Feature	Variable	Variations
Depth	$d$	2, 3
Number of Filters	$m_i, i = 1, 2, \dots, d$	2, 5
Filter Size	$l_i, i = 1, 2, \dots, d$	5, 10, 15
Pooling Size	$p_i, i = 1, 2, \dots, d$	5, 10
LSTM Features	$[n, h]$	[5, 1], [8, 2]

## Experiment 2: Training Hyperparameters

In the second experiment three training hyperparameters were varied. The model hyperparameters were set to the best performing configuration from the first experiment.

### 3.3.2 Input Data Study

In addition to the hyperparameter testing, two experiments were performed to assess the influence of the input data on the model. First, the number of dynamic and static signals was varied instead of using all signals as in the experiments before. This experiment was performed with the Vanilla CNN. The model hyperparameters were set to  $d = 4$ ,  $m_i = 2$ ,  $l_i = 15$  and  $p_i = 10$  and training hyperparameters to  $iLR = 0.0005$ ,  $\gamma = 1$ , and  $\lambda = 0.001$ . The architecture was trained with nine combinations of dynamic input features, each once with and once without static features.

Second, the models were trained with different datasets and data augmentation techniques. This experiment was performed both with the Vanilla CNN and CNN-LSTM. The model hyperparameters were varied and the training hyperparameters were set to  $iLR = 0.0005$ ,  $\gamma = 1$ , and  $\lambda = 0.001$ .

# Chapter 4

## Results

This chapter covers the results to answer the research questions. Section 4.1 presents the results of the extraction of the study population and an analysis of the datasets. The results of the hyperparameter study and the input data study are provided in section 4.2 and section 4.3, respectively.

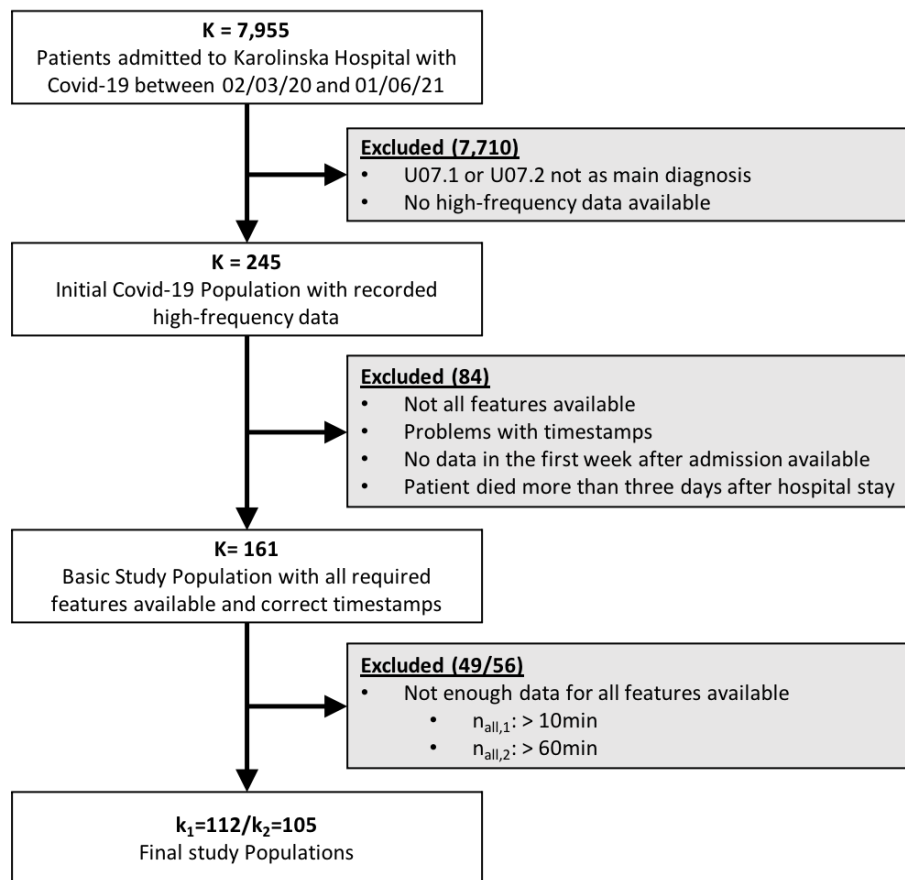
### 4.1 Study Population

Data from 7,955 patients with Covid-19 from Karolinska Hospital was available from the 2nd of March 2020 to the 1st of June 2021. From this, the study population was derived stepwise (Figure 4.1). Of those with Covid-19 as the main diagnosis (U07.1 or U07.2), high-frequency monitored data of 245 patients was recorded and stored, however, the number of features recorded varied (Table 4.1). Out of 55 signals, the most commonly recorded signals were ECG signals (II, v2, v5) as well as PPG (plet). As those four features give enough physiological information, they have been selected for the rest of the study. Static values included in the study are age at admission and sex.

**Table 4.1.** Availability of high-frequency signals that were recorded for more than 35% of all patients with available high-frequency data. The number of patients and the ratio are presented.

Signal	Main Object	All	Survived	Died
plet [-]	Blood Volume Change	241 98%	195 98%	46 98%
II [mV]	Heart Activity	225 92%	178 90%	47 100%
v2 [mV]	Heart Activity	225 92%	178 90%	47 100%
v5 [mV]	Heart Activity	225 92%	178 90%	47 100%
III [mV]	Heart Activity	196 80%	152 77%	44 93%
bv1 [mV]	Heart Activity	192 78%	151 76%	41 87%
avr [mV]	Heart Activity	116 47%	96 48%	20 42%
art [mmhg]	Blood Pressure	92 38%	68 34%	24 51%

*Note:* plet: photoplethysmogram; art: arterial blood pressure



**Figure 4.1.** Flow chart for defining study population and reasons for excluding patients.

84 subjects were removed because they either did not have all features recorded, timestamps were not aligning, no data was available in the first week after admission, or the patient died more than three days after hospital discharge. The exclusion was based only on the timestamps provided and not on the data itself.

From the basic study population of 161 subjects, two datasets were formed (Table 4.2). Dataset 1 (DS1) includes  $k_1 = 112$  patients with an extracted time-series length  $n_{\text{original}}$  of 10 minutes, 83 patients survived while 29 patients deceased. Dataset 2 (DS2) includes  $k_2 = 105$  patients with  $n_{\text{original}}$  of 60 minutes, 77 patient survived and 28 patients died. For DS1, one augmentation technique was used where the signal was randomly cropped, so that  $n_{\text{input}} = n_{\text{crop}} = 70,000$  data points. For DS2, two augmentation techniques were applied. First, the data was cropped resulting in  $n_{\text{input}} = n_{\text{crop}} = 400,000$  data points. Second, the dataset was cropped and split, so that  $n_{\text{input}} = n_{\text{split}} = 70,000$  data points resulting in 6 times more samples in one batch for training the model.

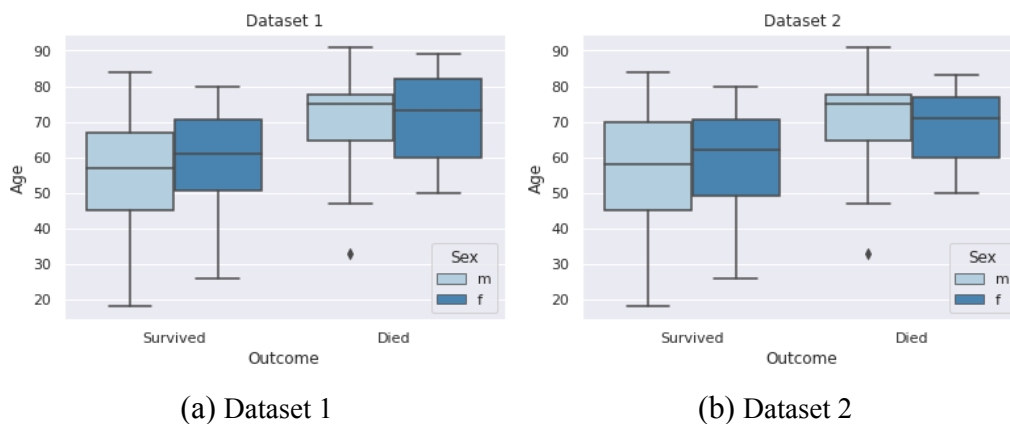
**Table 4.2.** Overview of datasets.

Dataset	Classes			Criteria		Augmentation		Subsets*		
	$k_i$	$k_{i,surv}$	$k_{i,died}$	$n_{original}$ [min]	$g$ [min]	Technique	$n_{input}$ [dp]	Strain	$S_{val}$	$S_{test}$
DS1	112	83	29	10	0.5	C	70,000	87/88	12/13	11
DS2	105	77	28	60	1	C	400,000	82/83	11/12	12
						C&S	70,000	492/498	66/72	72

Note: DS: dataset; min: minutes; dp: data points; C: cropped; S: split.

\* Sizes of subsets differ between the folds

The extra few patients in DS2 barely change the demographics of the study populations (Figure 4.2). The median age at admission of DS1 is 60 years, although the average age at admission was significantly lower for the survivors (75 years compared to 58 years). The median age at admission for DS2 is 62 years (59 years for surviving patients and 73 years for deceased patients). The percentage of women in DS1 was 33.02%, in DS2 it was 30.19%. It should be noted that for both datasets the age of females in the survivor group is higher than that of male survivors and, conversely, it is lower in the deceased female patients.



**Figure 4.2.** Age of patients depending on outcome and sex (f = female, m = male).

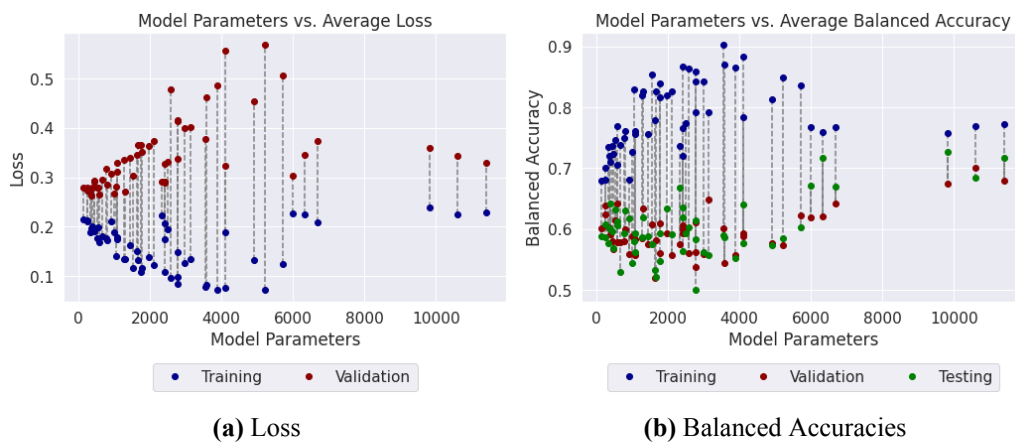
## 4.2 Hyperparameter Study

This section presents the model hyperparameter and training hyperparameter experiments for all three model architectures. For this part "DS1 Cropped" was used. A more detailed analysis of the performance of the model hyperparameters and results of all configurations can be found in Appendix C.

## 4.2.1 Vanilla CNN

### Experiment 1: Model Hyperparameters

Figure 4.3 shows the loss and BACC for all subsets for each configuration averaged over all runs and folds. The training loss increases for middle-sized models which seem to overfit. Large models on the other hand ( $> 60,000$  model parameters) are not able to train. This can be observed by the loss functions (Figure 4.4). Often the model begins to overfit and the training is stopped quite early, for other configurations the loss remains constant for all epochs. The loss functions of the smaller configuration are all constantly decreasing until they are stopped.

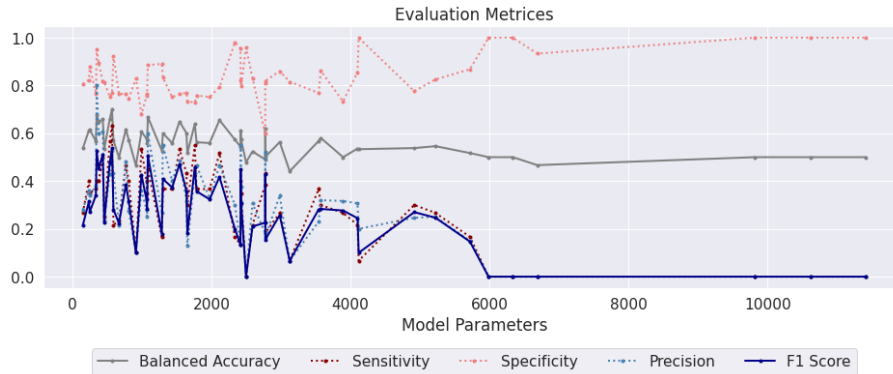


**Figure 4.3.** (a) Mean loss of Vanilla CNN Experiment 1. The loss is averaged over all runs and folds of one configuration. (b) Mean balanced accuracies of training, validation, and testing subset of Vanilla CNN Experiment 1. The results are averaged over all runs and folds of one configuration.



**Figure 4.4.** Loss curves for small and large configuration in Vanilla CNN Experiment 1. The loss for each epoch is averaged over all runs and folds of the configuration that have not been stopped. The mean and standard deviation are shown. A vertical grey line shows when the training for one fold was stopped.

Large models have a higher BACC for validation and testing subsets when comparing the average (Figure 4.3b). However, the results of the ensembles show that the more complex models have a lower BACC and a lower F1-score since just the survivor class is predicted (Figure 4.5). The best performing configuration yields a mean ensemble BACC of 0.701 and a F1-score of 0.538 (Table 4.3). Note that the spread of the results is very high and many configurations achieve a BACC below 0.5 and a very low F1-score.



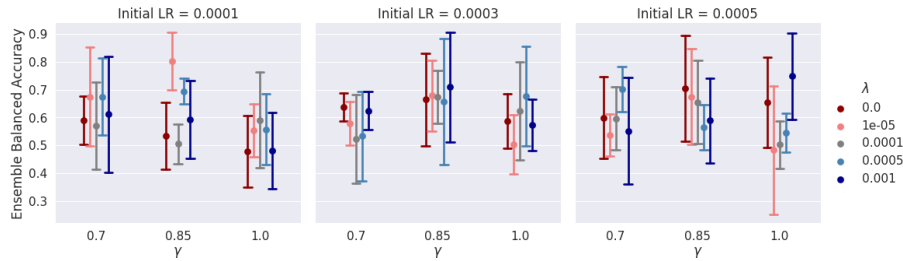
**Figure 4.5.** Vanilla CNN Experiment 1 ensemble evaluation. The graph shows the mean of each configuration over all runs.

**Table 4.3.** Vanilla CNN Experiment 1 best performing configurations based on ensemble balanced accuracy. The average and standard deviation of the six best configuration are listed.

Model Param.	Hyperparameters				Evaluation				
	d	m	l	p	Bal. Accuracy	Sensitivity	Specificity	Precision	F1-Score
350	4	2	15	10	0.675 (0.118)	0.400 (0.181)	<b>0.95 (0.068)</b>	<b>0.800 (0.274)</b>	0.527 (0.206)
438	4	2	15	7	0.658 (0.116)	0.500 (0.212)	0.817 (0.215)	0.607 (0.274)	0.510 (0.185)
538	3	2	15	10	0.659 (0.117)	0.567 (0.279)	0.752 (0.106)	0.431 (0.134)	0.477 (0.173)
570	4	2	5	5	<b>0.701 (0.177)</b>	<b>0.633 (0.375)</b>	0.769 (0.191)	0.525 (0.181)	<b>0.538 (0.237)</b>
1086	3	5	5	10	0.668 (0.115)	0.450 (0.162)	0.886 (0.079)	0.600 (0.253)	0.508 (0.194)
2116	4	5	10	5	0.655 (0.148)	0.517 (0.291)	0.793 (0.130)	0.467 (0.315)	0.417 (0.148)

### Experiment 2: Training Hyperparameters

The model hyperparameters were set to  $d = 4$ ,  $m_i = 2$ ,  $l_i = 5$  and  $p_i = 5$ . As before, all configurations exhibit high variance and no clear trend can be identified (Figure 4.6). However, some configuration are able to yield a high BACC and a high F1-score Table 4.4.



**Figure 4.6.** Vanilla CNN Experiment 2 ensemble balanced accuracies. The figure shows the mean and standard deviation over all runs for each configuration.

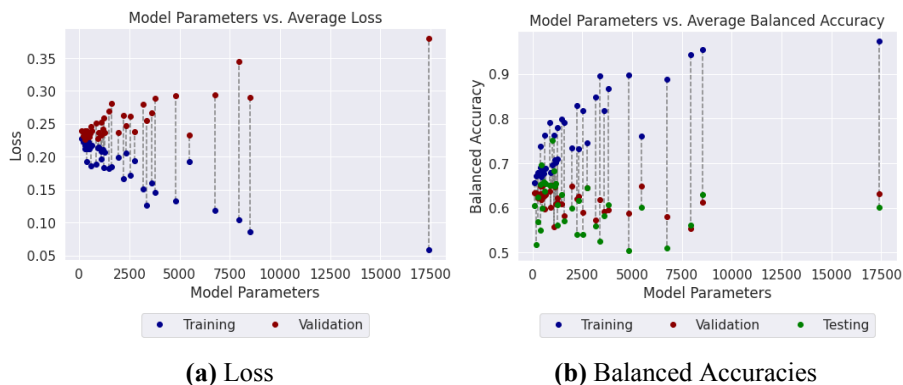
**Table 4.4.** Vanilla CNN Experiment 2 best performing configurations based on ensemble balanced accuracy. The average and standard deviation of the six best configuration are listed.

Hyperparameters			Evaluation				
LR	$\gamma$	$\lambda$	Bal. Accuracy	Sensitivity	Specificity	Precision	F1-Score
0.0001	0.85	0.00001	<b>0.801 (0.115)</b>	0.667 (0.312)	<b>0.936 (0.098)</b>	<b>0.853 (0.202)</b>	<b>0.677 (0.114)</b>
0.0001	0.85	0.0005	0.694 (0.052)	<b>0.767 (0.273)</b>	0.622 (0.205)	0.445 (0.135)	0.522 (0.120)
0.0003	0.85	0.001	0.709 (0.220)	0.650 (0.487)	0.768 (0.128)	0.363 (0.267)	0.443 (0.321)
0.0005	0.70	0.0005	0.701 (0.090)	0.583 (0.186)	0.819 (0.097)	0.547 (0.117)	0.548 (0.139)
0.0005	0.85	0.0	0.705 (0.212)	0.583 (0.373)	0.826 (0.140)	0.480 (0.398)	0.517 (0.384)
0.0005	1.00	0.001	0.748 (0.173)	0.700 (0.298)	0.796 (0.292)	0.611 (0.369)	0.599 (0.290)

## 4.2.2 CNN Inception

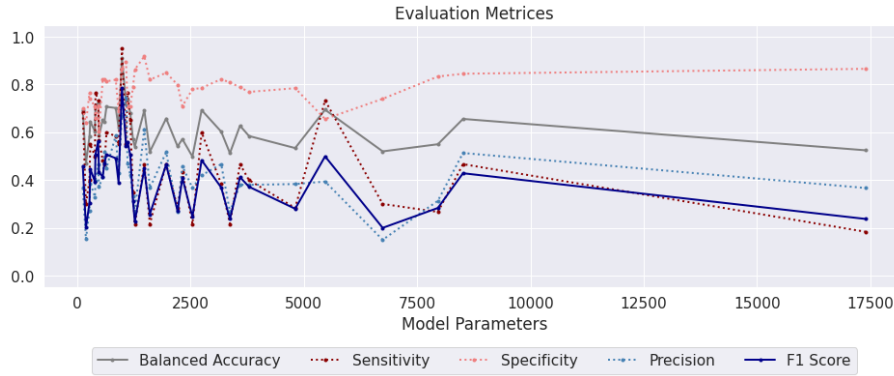
### Experiment 1: Model Hyperparameters

Similar to the Vanilla CNN, more complex models have a decreasing training loss and increasing validation loss, a sign that that the models overfit (Figure 4.7a). This is also indicated by the higher mean training BACC for more complex models (Figure 4.7b).



**Figure 4.7.** (a) Mean loss of CNN Inception Experiment 1. The loss is averaged over all runs and folds of one configuration. (b) Mean balanced accuracies of training, validation, and testing subset of CNN Inception Experiment 1. The results are averaged over all runs and folds of one configuration.

For the ensembles, the specificity is very high for most models, while the sensitivity stays below 0.6 in most cases (Figure 4.8). The precision is very low overall and never reaches more than 0.746. Several models achieve a BACC over 0.7 and an F1-score over 0.5, but even for the best models, the precision is below 0.5. The best performance is achieved by a middle sized configurations with 985 model parameters (Table 4.5).



**Figure 4.8.** CNN Inception Experiment 1 ensemble evaluation. The graph shows the mean of each configuration over all runs.

**Table 4.5.** CNN Inception Experiment 1 best performing configurations based on ensemble balanced accuracy. The average and standard deviation of the six best configuration are listed.

Model Param.	Hyperparameters				Evaluation				
	$c_b$	$m$	$l$	$d$	Bal. Accuracy	Sensitivity	Specificity	Precision	F1-Score
400	1	2	100,50,25	1	0.713 (0.037)	0.767 (0.224)	0.660 (0.173)	0.399 (0.098)	0.507 (0.079)
460	1	5	40,20,10	1	0.722 (0.162)	0.733 (0.279)	0.711 (0.127)	0.466 (0.183)	0.563 (0.205)
643	3	5	20,10,5	1	0.707 (0.159)	0.600 (0.253)	0.813 (0.083)	0.450 (0.201)	0.507 (0.214)
985	1	5	100,50,25	1	<b>0.909 (0.089)</b>	<b>0.950 (0.112)</b>	0.868 (0.066)	0.670 (0.053)	<b>0.785 (0.071)</b>
1075	1	5	20,10,5	3	0.721 (0.091)	0.550 (0.298)	<b>0.893 (0.123)</b>	<b>0.747 (0.256)</b>	0.542 (0.099)
1108	3	2	100,50,25	1	0.738 (0.143)	0.767 (0.325)	0.709 (0.184)	0.469 (0.150)	0.560 (0.192)

When studying the different hyperparameters, it can be observed that especially the depth affects the results, with the configurations with  $d = 1$  performing considerably better. Also, a lower number of filters  $m$  leads to better results. For the filter size  $l$  and the bottleneck size  $c_b$ , no clear tendency can be detected; instead, the results depend more on the combination of the hyperparameters.

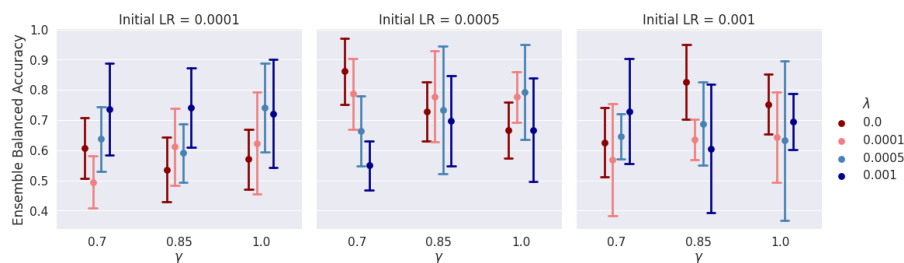
### Experiment 2: Training Hyperparameters

The model hyperparameters were set to  $m = 5$ ,  $c_b = 1$ ,  $l = [100, 50, 25]$  and  $d = 1$ . First of all, it must be emphasized that the results of the previous experiment with the same configuration are not reproduced. Nevertheless, similar configurations of training parameters yield good results (Table 4.6). A closer look at the hyperparameters shows that especially the iLR influences the result, neither a lower nor a higher iLR leads to an

improvement (Figure 4.9). For the other two hyperparameters, no clear tendency can be seen.

**Table 4.6.** CNN Inception Experiment 2 best performing configurations based on ensemble balanced accuracy. The average and standard deviation of the six best configuration are listed.

Hyperparameters			Evaluation				
LR	$\gamma$	$\lambda$	Bal. Accuracy	Sensitivity	Specificity	Precision	F1-Score
0.0005	0.70	0.0	<b>0.861 (0.123)</b>	<b>0.950 (0.112)</b>	0.773 (0.171)	0.650 (0.253)	<b>0.751 (0.201)</b>
0.0005	0.70	0.0001	0.786 (0.130)	0.800 (0.209)	0.773 (0.097)	0.537 (0.153)	0.638 (0.164)
0.0005	0.85	0.0001	0.778 (0.168)	0.767 (0.325)	0.789 (0.083)	0.467 (0.139)	0.573 (0.198)
0.0005	1.00	0.0001	0.776 (0.093)	0.767 (0.224)	0.786 (0.063)	0.513 (0.096)	0.590 (0.043)
0.0005	1.00	0.0005	0.792 (0.176)	0.850 (0.224)	0.734 (0.164)	0.560 (0.288)	0.660 (0.259)
0.0010	0.85	0.0	0.826 (0.138)	0.833 (0.236)	<b>0.819 (0.167)</b>	<b>0.669 (0.225)</b>	0.720 (0.197)



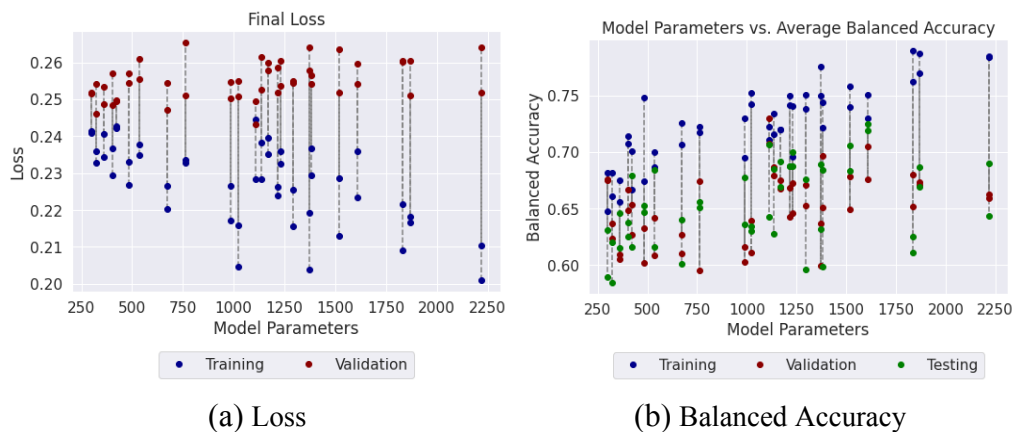
**Figure 4.9.** CNN Inception Experiment 2 ensemble balanced accuracies. The figure shows the mean and standard deviation over all runs for each configuration

## 4.2.3 CNN-LSTM

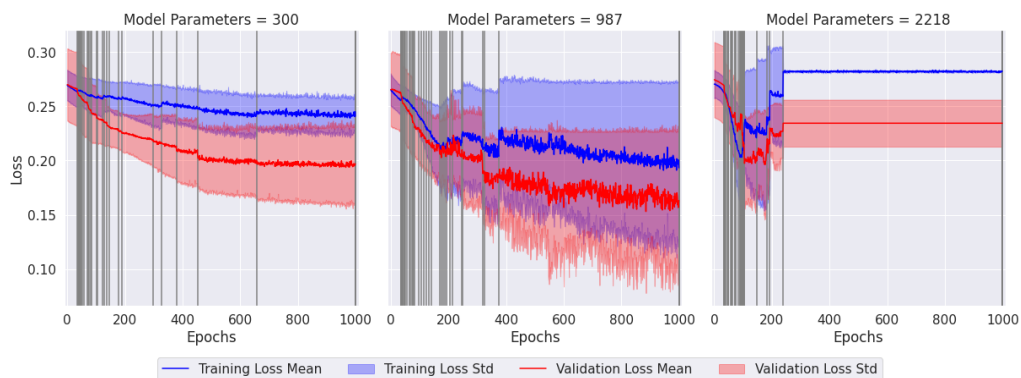
### Experiment 1: Model Hyperparameters

The CNN-LSTM was trained on 48 different configurations with less than 2200 model parameters, thus with less complex configurations than the two architectures before. The training loss decreases for more complex models, and the validation loss on the other hand is constant (Figure 4.10a).

For all configurations, it can be noted that sometimes no training was possible and the loss stays constant for all epochs. For more complex models there is a trend to overfit so that the training is already stopped after a few epochs (Figure 4.11). This is also reflected in the higher average BACC on the training subset (Figure 4.10b). For the smallest configuration on the other hand the loss decreases just slightly over time. Looking at the loss functions of the best-performing configuration, it is noticeable that it is quite possible to reduce the loss, although the mean is still high because the training is often stopped early.

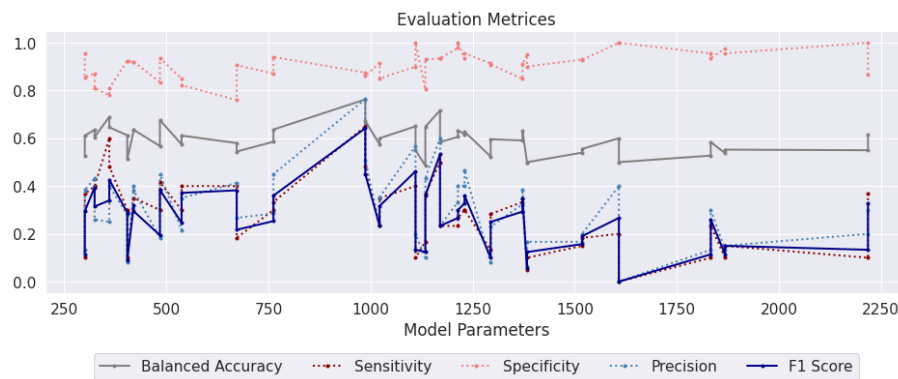


**Figure 4.10.** (a) Mean loss of CNN-LSTM Experiment 1. The loss is averaged over all runs and folds of one configuration. (b) Mean balanced accuracies of training, validation, and testing subset of CNN-LSTM Experiment 1. The results are averaged over all runs and folds of one configuration.



**Figure 4.11.** Loss curves for small (left), best (middle) and large (right) configuration in CNN-LSTM Experiment 1. The loss for each epoch is averaged over all runs and folds of the configuration that have not been stopped. The mean and standard deviation are shown. A vertical grey line shows when the training for one fold was stopped. The model configurations for  $p_i = 10$  are shown.

These effects balance each other out when examining the ensemble evaluation. Most configurations yield an average ensemble BACC over 0.5, however, the sensitivity and precision is very low (Figure 4.12). This results in a very low F1-Score for most configurations. Overall, the best results are yield with a smaller depth  $d = 2$ , higher pooling size  $p_i = 10$  and higher filter sizes  $l_i = 10/15$ . When comparing the two LSTM options, the smaller configuration with  $n = 5$  and  $h = 1$  performs better. The best-performing configuration achieves a BACC of 0.762 and a F1-Score of 0.642 (Table 4.7)



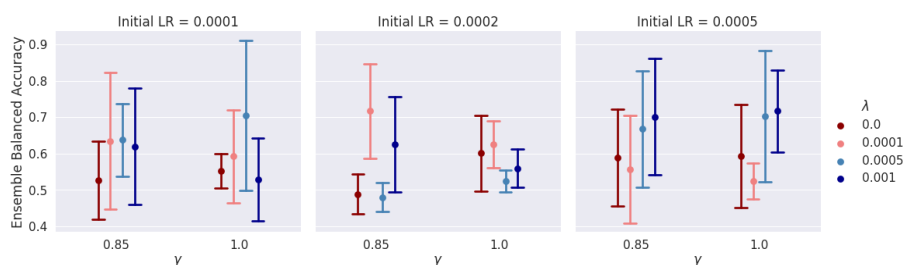
**Figure 4.12.** CNN-LSTM Experiment 1 ensemble evaluation. The graph shows the mean of each configuration over all runs.

**Table 4.7.** CNN-LSTM Experiment 1 best performing configurations based on ensemble balanced accuracy. The average and standard deviation of the six best configuration are listed.

		<i>Hyperparameters</i>					<i>Evaluation</i>				
<b>Model Param.</b>	<b>d</b>	<b><math>m_i</math></b>	<b><math>l_i</math></b>	<b><math>p_i</math></b>	<b>[n, h]</b>	<b>Bal. Accuracy</b>	<b>Sensitivity</b>	<b>Specificity</b>	<b>Precision</b>	<b>F1-Score</b>	
360	2	2	10	10	5,1	0.690 (0.195)	0.600 (0.548)	0.780 (0.268)	0.250 (0.276)	0.340 (0.344)	
484	3	2	15	5	5,1	0.676 (0.181)	0.417 (0.449)	<b>0.935 (0.093)</b>	0.450 (0.447)	0.383 (0.361)	
987	2	5	15	10	5,1	<b>0.762 (0.173)</b>	<b>0.650 (0.379)</b>	0.875 (0.153)	<b>0.764 (0.234)</b>	<b>0.642 (0.274)</b>	
987	2	5	15	5	5,1	0.672 (0.155)	0.483 (0.393)	0.861 (0.127)	0.507 (0.370)	0.45 (0.298)	
1110	2	2	5	10	8,2	0.650 (0.163)	0.400 (0.285)	0.900 (0.137)	0.567 (0.435)	0.462 (0.333)	
1170	2	2	10	10	8,2	0.717 (0.122)	0.500 (0.289)	0.933 (0.061)	0.600 (0.365)	0.533 (0.298)	

## Experiment 2: Training Hyperparameters

Next, the training hyperparameters are compared using the model hyperparameters  $d = 2, m_i = 5, l_i = 15, p_i = 10$  and LSTM Features [5, 1]. As shown in Figure 4.13, the BACC is never much higher than 0.7. As before, the iLR affects the results, performing best at 0.0005. In addition, L1 regularization has a positive impact on the result. However, since the sensitivity is never higher than 0.55, the F1-score for the best configurations is also only about 0.5 (Table 4.8).



**Figure 4.13.** CNN-LSTM Experiment 2 ensemble balanced accuracies. The figure shows the mean and standard deviation over all runs for each configuration

**Table 4.8.** CNN-LSTM Experiment 2 best performing configurations based on ensemble balanced accuracy. The average and standard deviation of the six best configuration are listed.

Hyperparameters			Evaluation				
LR	$\gamma$	$\lambda$	Bal. Accuracy	Sensitivity	Specificity	Precision	F1-Score
0.0001	1.00	0.0005	0.705 (0.23)	<b>0.550 (0.447)</b>	0.861 (0.062)	0.483 (0.291)	0.498 (0.352)
0.0002	0.85	0.0001	0.717 (0.145)	0.500 (0.373)	0.933 (0.099)	0.653 (0.409)	0.517 (0.303)
0.0005	0.85	0.0005	0.668 (0.178)	0.400 (0.365)	0.936 (0.059)	0.467 (0.447)	0.427 (0.393)
0.0005	0.85	0.001	0.701 (0.179)	0.467 (0.361)	0.936 (0.098)	0.600 (0.435)	0.493 (0.325)
0.0005	1.00	0.0005	0.702 (0.202)	0.467 (0.361)	0.938 (0.057)	0.600 (0.418)	0.513 (0.366)
0.0005	1.00	0.001	<b>0.717 (0.126)</b>	0.433 (0.253)	<b>1.000 (0.000)</b>	<b>0.800 (0.447)</b>	<b>0.560 (0.318)</b>

Note: Param.: Parameters; LR: Initial Learning Rate;  $\gamma$ : Step Scheduler Parameter;  $\lambda$ : L1 Regularization Parameter; Bal.: Balanced.

## 4.2.4 Comparison

For all three model architectures, similar observations are made. Throughout all experiments, the variation of the results is very high and the results are just limited reproducible. The model hyperparameters have a higher impact than the training hyperparameters, just the iLR influences the ability to train. In particular, models with more than 2000 parameters do not provide sufficient results. For most configurations the specificity is much higher than the sensitivity, meaning that all models tend to predict the negative class. Moreover, the precision is very low which leads to a low F1-Score. Overall, the CNN Inception achieves the best results (Table 4.9). Both the BACC and the F1-score are higher than for the other two architectures. The best configurations of the CNN Inception and the CNN-LSTM have almost the same number, while the Vanilla CNN has just half the number of model parameters. Noticeably, the standard deviation for the CNN-LSTM is slightly higher than for the other two models.

**Table 4.9.** Comparison of best performing configurations based on ensemble balanced accuracy. The average and standard deviation of the two best configuration for every model architecture and experiment are listed.

Model			Evaluation				
Architecture	Exp	Parameters	Bal. Accuracy	Sensitivity	Specificity	Precision	F1-Score
Vanilla CNN	Exp 1	350	0.675 (0.118)	0.400 (0.181)	0.950 (0.068)	0.800 (0.274)	0.527 (0.206)
Vanilla CNN	Exp 1	570	0.701 (0.177)	0.633 (0.375)	0.769 (0.191)	0.525 (0.181)	0.538 (0.237)
Vanilla CNN	Exp 2	570	0.801 (0.115)	0.667 (0.312)	0.936 (0.098)	0.853 (0.202)	0.677 (0.114)
Vanilla CNN	Exp 2	570	0.748 (0.173)	0.700 (0.298)	0.796 (0.292)	0.611 (0.369)	0.599 (0.290)
CNN Incep.	Exp 1	985	0.909 (0.089)	0.950 (0.112)	0.868 (0.066)	0.670 (0.053)	0.785 (0.071)
CNN Incep.	Exp 1	1075	0.721 (0.091)	0.550 (0.298)	0.893 (0.123)	0.747 (0.256)	0.542 (0.099)
CNN Incep.	Exp 2	985	0.861 (0.123)	0.950 (0.112)	0.773 (0.171)	0.650 (0.253)	0.751 (0.201)
CNN Incep.	Exp 2	985	0.826 (0.138)	0.833 (0.236)	0.819 (0.167)	0.669 (0.225)	0.720 (0.197)
CNN-LSTM	Exp 1	987	0.762 (0.173)	0.650 (0.379)	0.875 (0.153)	0.764 (0.234)	0.642 (0.274)
CNN-LSTM	Exp 1	1170	0.717 (0.122)	0.500 (0.289)	0.933 (0.061)	0.600 (0.365)	0.533 (0.298)
CNN-LSTM	Exp 2	987	0.717 (0.145)	0.500 (0.373)	0.933 (0.099)	0.653 (0.409)	0.517 (0.303)
CNN-LSTM	Exp 2	987	0.717 (0.126)	0.433 (0.253)	1.000 (0.000)	0.800 (0.447)	0.560 (0.318)

Note: Exp: Experiment; Param.: Parameters; Bal. Balanced.

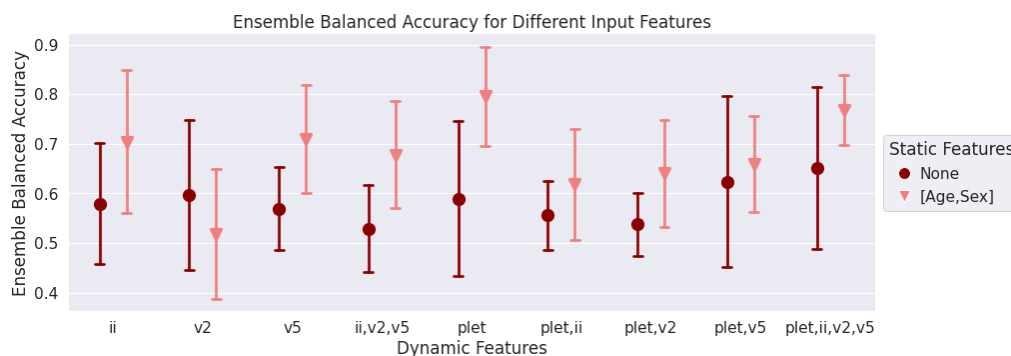
## 4.3 Input Data Study

This part covers the results of the experiments with varying input data to answer the minimum amount of data in terms of number of signals and length is, that is required for a prediction.

### 4.3.1 Input Features

To begin with, an average BACC over 0.5 is achieved with all input signals (Figure 4.14). The best performance without static features is archived when using all dynamic features, but the differences are very small especially when considering the standard deviation. However, not every combination of signals leads to improvement, as can be seen from the combination of the three ECG signals without static features.

The models without static features outperform their counter models in almost all cases. It is notable that especially with a reduced number of dynamic features, the static features make a considerable difference. The best results using just one signal in combination with that static features were achieved with the PPG signal (plet). Again, the combination of dynamic signals does not lead to an improvement in all cases.

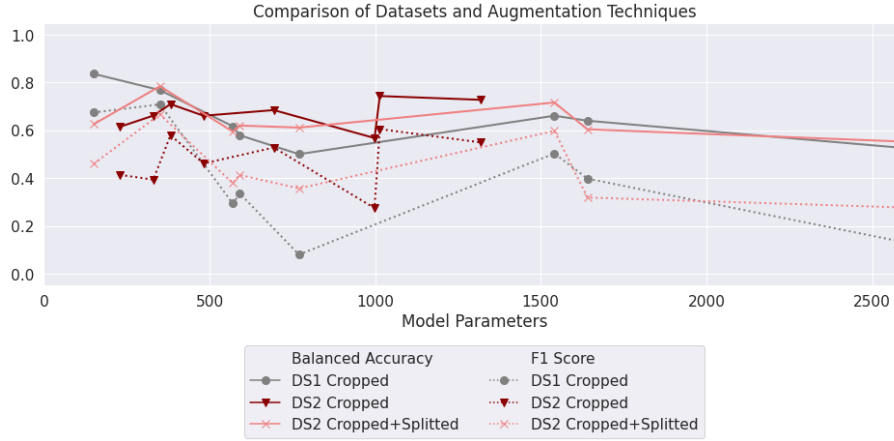


**Figure 4.14.** Vanilla CNN variation of input signals. The figure shows the mean and standard deviation of the ensemble balanced accuracy for all runs of one configuration.

### 4.3.2 Dataset

Additionally to the number of features, the input data can be varied regarding the signal length  $n_{input}$  and the augmentation technique. For the Vanilla CNN, the model performances of "DS1 Cropped" and "DS2 Cropped+Splitted" evolves similarly for different configurations (Figure 4.15). Using more samples by splitting the signal slightly improves the results in most cases. The best three results with both datasets are obtained by the same configurations (Table 4.10). "DS2 Cropped" performs not considerably different from those with shorter input lengths. However, it is notable that the specificity

is increased while the sensitivity is decreased. Also, a good average precision is yielded, but with a very high standard deviation which is also reflected in the F1-Score.



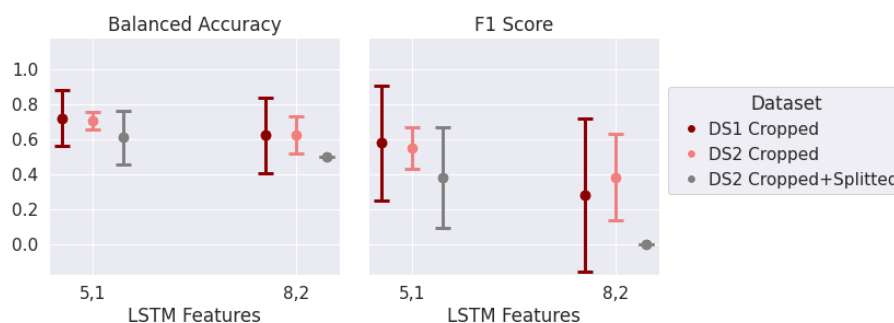
**Figure 4.15.** Vanilla CNN comparison of datasets and augmentation techniques. The graph shows the mean ensemble balanced accuracy and F1-Score over all runs of one configuration.

**Table 4.10.** Evaluation of Vanilla CNN for different datasets and augmentation techniques. For each dataset and augmentation technique, the best three configurations are listed. The mean and standard deviation over all runs of the same configuration are presented.

<i>Model</i>		<i>Evaluation</i>				
<b>Dataset</b>	<b>Param.</b>	<b>Bal. Accuracy</b>	<b>Sensitivity</b>	<b>Specificity</b>	<b>Precision</b>	<b>F1-Score</b>
DS1 C	150	<b>0.836 (0.101)</b>	<b>0.833 (0.236)</b>	0.839 (0.105)	0.633 (0.217)	0.674 (0.081)
DS1 C	350	0.768 (0.079)	0.650 (0.137)	0.886 (0.186)	<b>0.850 (0.224)</b>	<b>0.708 (0.099)</b>
DS1 C	1541	0.661 (0.053)	0.500 (0.167)	0.821 (0.066)	0.533 (0.075)	0.503 (0.094)
DS2 C	382	0.709 (0.194)	0.517 (0.303)	0.902 (0.097)	0.667 (0.408)	0.579 (0.342)
DS2 C	1014	0.743 (0.114)	0.533 (0.274)	0.953 (0.065)	<b>0.850 (0.224)</b>	0.605 (0.158)
DS2 C	1318	0.727 (0.174)	0.483 (0.393)	<b>0.971 (0.064)</b>	0.760 (0.434)	0.549 (0.359)
DS2 C&S	150	0.625 (0.194)	0.406 (0.274)	0.845 (0.215)	0.678 (0.409)	0.460 (0.267)
DS2 C&S	350	0.784 (0.095)	0.700 (0.183)	0.868 (0.087)	0.664 (0.215)	0.666 (0.157)
DS2 C&S	1541	0.716 (0.196)	0.625 (0.315)	0.807 (0.124)	0.579 (0.270)	0.597 (0.286)

*Note:* DS1: Dataset 1; DS2: Dataset 2; C: Cropped; S: Splitted

For the CNN-LSTM, the "DS2 Cropped+Splitted" performs worse for both trained configurations (Figure 4.16). In particular, for the more complex LSTM, the model is not able to train and predict just one class, resulting in a F1-Score of 0.0 (Table 4.11). The "DS2 Cropped" on the other hand improves the results. In particular, the standard deviation of both the BACC and the F1-score decreased compared to "DS1 Cropped", indicating a more stable prediction. This is even more notable considering that the effect for the Vanilla CNN is the opposite.



**Figure 4.16.** CNN-LSTM comparison of datasets and augmentation techniques. The graph shows the mean and standard deviation of the ensemble balanced accuracy and F1-Score over all runs of one configuration.

**Table 4.11.** Evaluation of CNN-LSTM for different datasets and augmentation techniques. The mean and standard deviation over all runs of the same configuration are presented.

<i>Model</i>		<i>Evaluation</i>				
<b>LSTM</b>	<b>Dataset</b>	<b>Bal. Accuracy</b>	<b>Sensitivity</b>	<b>Specificity</b>	<b>Precision</b>	<b>F1-Score</b>
5,1	DS1 C	<b>0.719 (0.161)</b>	<b>0.517 (0.303)</b>	0.921 (0.072)	0.683 (0.41)	<b>0.577 (0.327)</b>
5,1	DS1 C+S	0.608 (0.153)	0.333 (0.252)	0.883 (0.064)	0.458 (0.360)	0.381 (0.288)
5,1	DS2 C	0.703 (0.052)	0.450 (0.112)	0.956 (0.061)	<b>0.800 (0.274)</b>	0.547 (0.117)
8,2	DS1 C	0.621 (0.217)	0.267 (0.435)	0.975 (0.056)	0.300 (0.447)	0.280 (0.438)
8,2	DS1 C+S	0.500 (0.000)	0.000 (0.000)	<b>1.000 (0.000)</b>	0.000 (0.000)	0.000 (0.000)
8,2	DS2 C	0.623 (0.105)	0.300 (0.240)	0.946 (0.074)	0.633 (0.415)	0.380 (0.247)

*Note:* DS1: Dataset 1; DS2: Dataset 2; C: Cropped; S: Splitted

# Chapter 5

## Discussion

This chapter discusses the methodology for the data processing (section 5.1), the model development and performance (section 5.2), and comparison to previous results (section 5.3) and ends with other limitations (section 5.5) and future work (section 5.6).

### 5.1 Data Processing and Study Population

The Covid-19 pandemic was an exceptional situation that led to a unique study population but led also to compromises due to problems during the data recording. Through the possibility of prospectively collecting the data, high-frequency data was used which is usually not stored due to capacity reasons. The data was collected from patients with the same disease in a short period, a unique situation that leads to a more uniform dataset. Previous studies had to account for comorbidities instead as they have a high impact on the vital parameters [26, 44, 45]. However, although high-frequency data was recorded for over 200 patients, signals were missing and the recordings include long gaps and empty signals. This needs to be expected in clinical everyday life, but it reduces the size of the dataset drastically. Additionally, patients had to be excluded due to unclear time stamps. It was also difficult to interpret whether the patient was treated in an ICU, as wards were used as ICUs during the pandemic. Hence, the ICU timestamps could not be considered.

Overall, the final study population was very small, other Covid-19 datasets range from 181 to over 6,000 patients [26, 44–46]. It should be considered that not only the number of samples is relevant, but also the relation to the size of the input data and the model. In this work, the proposed models are very large, so it needs many independent inputs to develop a generalized model. The small population size influenced the ability to train as can be seen in the loss curves and the results. Consequently, individual patients had a high influence on the outcome. Cross validation was used to try to counteract overfitting, but it was not achieved in all cases. In particular, the tendency for the validation loss to be lower than the training loss indicates that randomization plays a large role in cross-validation. Due to the random distribution of the subsets, validation and test subset might

not be representative, resulting in an early terminated training due to a high validation loss. This in turn has a major impact on average performance and ensemble performance. It should be observed that some models perform very well while others do not train at all. This could be prevented by using a larger study population.

Additionally to the population size, other factors during the data processing might have influenced the results. Criteria had to be chosen when selecting thresholds for the study population and extracting the time series data, such as the length of gaps or the criteria to exclude patients. Further, it should be critically noted that the first available data was extracted without taking into consideration the admission and discharge time. This is relevant as some patients were already treated before they were monitored. Other studies tried to avoid this problem by using data depending on the time to death, which makes it easier to develop the model and classify the results [25, 47]. However, the question of clinical relevance must be raised here, since it is no longer possible to intervene when the prediction is made.

Another aspect that should be discussed is the time period that lays between the prediction and the event. In this work, the first available data was used to make a prediction about the discharge of the entire stay. However, the lengths of stay differ between the patients, therefore, sometimes the prediction was very close to the event and sometimes it was very long time period. During this time the patient's condition can change significantly and a later prediction might results in a different outcome. The output also does not include a time at which the patient will die, which is problematic when it comes to adjust the treatment.

Last, the filtering of the time-series data could be improved instead of applying simple linear filters. Although they work well in most cases, filtering methods such as the wavelet transform could avoid information loss [48].

## 5.2 Model Development and Performance

Three model architectures were selected and compared. While they were based on models developed for time-series data, they had to be adapted to high-frequency data. The main difficulty was selecting appropriate model hyperparameters for the long input sequences while training with a small study population. The focus of the thesis was on the feasibility of the models and not to find the perfect model. Therefore, a simplified grid search was used to find a good model. Instead, a more thorough random search should be used to be able to find better hyperparameters. Additionally, more runs are required to ensure reproducibility of results, reduce the high variance of the results, and to be able to make a more stable conclusion.

The Vanilla CNN required a high pooling size to keep the model complexity manageable to avoid too many parameters for the fully connected layer. However, the architecture used as a basis was tested only for input lengths of 60 to 427 and works therefore with much smaller pooling sizes [41]. If the signal is heavily pooled early,

information may be lost. Likewise, the CNN Inception model was developed for sequence lengths of a maximum of 1024 data points [42]. Due to a much longer input length and a low model complexity, the applied global average pooling has a much larger impact, which could cause a loss of important information. The CNN-LSTM tries to avoid extensive pooling and instead considers the time dimension. This works especially well for longer sequences as shown with the results for DS2, where the standard deviation could be reduced significantly.

Overall, it was observed that the models tend to predict the majority class. The datasets were highly unbalanced, having more patients of the survival class. To counteract this, class weights were included in the loss function. However, as the results show, other methods should be used to further reduce the influence of imbalance. This is important as imbalance can be the reason for biases and poor generalization [49]. A proven method to tackle imbalance is oversampling, where minority class samples are used more often during training; other methods include undersampling or adjusting the classification threshold [50].

Finally, the influence of the static variables on the result should be considered. The input data study showed that using the static variables improves the results significantly. As the demographic analysis pointed out, there are significant differences between the two classes. The question remains open, how well a model would perform which has only these two variables and whether the time-series data necessarily improves the result. Additionally, the experiment showed that the combination of dynamic factors does not always lead to an improvement in the results. This could be due to the correlation of the ECG values, although this was not investigated further due to the large standard deviations.

### 5.3 Comparison with Previous Results

The results are similar to the previous findings (Table 5.1). Although the results are not reproducible in all cases, it was possible to achieve balanced accuracies of more than 0.7 for several configurations. The model of Zhu et al. yields significantly better results, however, just the AUC is given as metrics which makes it difficult to compare [44]. Villegas et al. reports a high accuracy and F1-score and has a low variance of the results, confirming the previous assumption that the use of RNNs is highly suitable for the problem since they take into consideration the temporal component [46]. The models of Li et al. and Cheng et al. on the other hand have similar F1-scores and also a higher spread of results [26, 45]. However, the models developed in this thesis all work with a much smaller dataset and a lower number of input parameters, but instead with waveform data.

**Table 5.1.** Comparison of presented results with previous studies applying deep learning methods to Covid-19 populations to predict mortality.

Reference	Size of Study Population	Input Parameters	Model	ACC	AUC	F1-Score
Li [26]	1,108	DG, CM, VS, lab, symptoms	DNN	0.853	0.844	0.616
Zhu [44]	181	DG, CM, VS, lab, symptoms	DNN	-	0.968	-
Villegas [46]	6,087	DG, VS, lab, medication	Ensemble of RNNs	0.901	0.845*	0.844
Cheng [45]	654	DG, CM, VS, lab, X-rays	DNN, LTBN	0.732	0.727	0.707
3.1.1	105	DG, VS	CNN	-	0.701** 0.801**	0.538 (Exp1) 0.677 (Exp2)
3.1.2	105	DG, VS	CNN Inc	-	0.909** 0.861**	0.785 (Exp1) 0.751 (Exp2)
3.1.3	105	DG, VS	CNN-LSTM	-	0.762** 0.717**	0.642 (Exp1) 0.560 (Exp2)

*Note:* ACC: Accuracy; AUC: Area under the curve; DG: Demographics; CM: Comorbidity; VS: Vital Signs; lab: Laboratory Test Values; DNN: Deep Neural Network; RNN: Recurrent; Neural Network; LTBN: Longitudinal transformer-based network. Exp1: Experiment 1 comparing model hyperparameters; Exp2: Experiment 2 comparing training hyperparameters.

\* AUC was calculated from the reported sensitivity and specificity and thus is equal to the balanced accuracy.

\*\* AUC was not calculated based on the predictions and thus is equal to the balanced accuracy.

Although the comparison shows that the models developed here are similar to previous models, the question remains to what extent this can improve everyday clinical practice. In particular, how much better those prediction models are than the current assessments of the physicians or prediction scores based on logistic regression that are commonly used to assess the outcome of patients in the ICU. To assess this, it would be necessary to define a baseline based on human expertise, with physicians labeling patients without the help of deep learning models.

## 5.4 Explainability

Explainability of deep learning methods is crucial when it comes to working with medical data to make the physicians and patients understand the outcome. However, in this work, no explanation is provided along with the results, making the model unusable for clinical use at this time. One question is what such an explanation might look like with the input data used. A widely used explainability method is SHapley Additive exPlanation (SHAP), which points to specific features that are relevant, but this is not as easy when working with waveform data [51]. Instead, an explainability method could point out at what time the time-series change creating a temporal attention map [52]. For instance, Xu et al. developed an attention mechanism for waveform data incorporating external expertise

that indicates which feature influences the outcome most and at which time [53]. This could allow the physician to look at specific sequences to better understand the reasons for the outcome, rather than basing a treatment on a simple prediction.

## 5.5 Other Limitations

In addition to the limited amount of data and missing data, the work encountered other limitations. Although the data were collected prospectively, it was not possible to influence which data were collected. Next, just the data of one hospital was taken into consideration. In addition, the dataset may be biased because it is not clear for what reasons high-frequency data was recorded, considering that it was only available for a very small number of patients. Lastly, the number of experiments and runs was limited by computational capacity.

## 5.6 Future Work

Future work should be performed based on the presented results. First, the current model architectures can be improved by a more thorough hyperparameter search using a random approach, carrying out more runs, or pre-train the model with a convolutional auto-encoder. Next, the experiments should be repeated with a larger study population by either collecting more data or combining it with a publicly available dataset such as the MIMIC waveform database [54]. In the latter case, transfer learning could be used, where models are pre-trained with the disease-unspecific dataset and then trained with a smaller dataset for only one disease, such as Covid-19. Third, the interaction of static and dynamic factors should be studied in more detail. Low-frequency time-series data from EHR, which was used in other studies, can also be considered. This could increase the flexibility of the model if a signal is not available. Furthermore, the models developed use dynamic input signal, but only one output is produced. Instead, the model should be extended to allow for dynamic prediction, e.g. provide the mortality probability on an hourly/daily basis. Finally, explainability should be increased by inserting attention layers or by using other architectures such as Transformer models [31].

# Chapter 6

## Conclusion

The present thesis addressed the question of whether high-frequency monitoring data can be used to predict mortality for Covid-19 patients. The results are promising and show that it is indeed possible to make predictions. However, the experiments have mainly been concerned with general feasibility, so it is not possible to say how well the models can provide correct predictions. Therefore, further experiments must be performed until the model can be used for clinical application.

The major drawback of the thesis is the small size of the dataset. The first sub-question was how a clinically relevant data set can be formed from a small sample of patients. This work presented techniques to create different datasets based on the length of extracted data and augmentation techniques. Nevertheless, the high variance of the results shows that training a deep learning model with a low number of patients remains challenging. Therefore, a larger dataset is needed or the possibility of pre-training should be explored.

The second sub-question, which model architecture is suited best, can not be fully answered. None of the three model architectures introduced has consistently better results than the others. The highest balanced accuracies and F1-Scores were yielded with the CNN Inception. However, all three model architectures should be further tested with more data so that a truly conclusive evaluation can be made.

Last, it was investigated how much data is required to make a stable prediction. Not all high-frequency signals were required for a good prediction indicating that a model can be built based on a limited number of features. However, it was significant that adding static demographic features improved the prediction. Moreover, the use of a longer time interval could reduce the variance of the results, at least for the CNN-LSTM. The question of exactly how many features and what length is needed, cannot be answered due to the large deviation of the results. To conclude, not necessarily more data per patient is needed, but this has to be evaluated with a larger study population.

# References

- [1] Mihaela van der Schaar et al. “How artificial intelligence and machine learning can help healthcare systems respond to COVID-19.” In: *Machine Learning* 110.1 (Jan. 2021), pp. 1–14. ISSN: 15730565. DOI: [10.1007/S10994-020-05928-X/FIGURES/1](https://doi.org/10.1007/S10994-020-05928-X/FIGURES/1).
- [2] Fang Jiang et al. “Review of the Clinical Characteristics of Coronavirus Disease 2019 (COVID-19).” In: *Journal of General Internal Medicine* 35.5 (May 2020), p. 1545. ISSN: 15251497. DOI: [10.1007/S11606-020-05762-W](https://doi.org/10.1007/S11606-020-05762-W).
- [3] Lisa Rosenbaum. “Facing Covid-19 in Italy — Ethics, Logistics, and Therapeutics on the Epidemic’s Front Line.” In: *New England Journal of Medicine* 382.20 (May 2020), pp. 1873–1875. ISSN: 0028-4793. DOI: [10.1056/NEJMP2005492](https://doi.org/10.1056/NEJMP2005492).
- [4] Robert D. Truog, Christine Mitchell, and George Q. Daley. “The Toughest Triage — Allocating Ventilators in a Pandemic.” In: *New England Journal of Medicine* 382.21 (May 2020), pp. 1973–1975. ISSN: 0028-4793. DOI: [10.1056/NEJMP2005689](https://doi.org/10.1056/NEJMP2005689).
- [5] Ata Mahmoodpoor, Farnad Imani, and Hassan Soleimanpour. “COVID-19 Is Not Over and Needs Prediction Scores: An Endless Road!” In: *Anesthesiology and Pain Medicine* 11.6 (Dec. 2021), p. 121654. ISSN: 22287531. DOI: [10.5812/AAPM.121654](https://doi.org/10.5812/AAPM.121654).
- [6] Jean-Roger Le Gall. “The use of severity scores in the intensive care unit.” In: *Intensive Care Medicine* 2005 31:12 31.12 (Oct. 2005), pp. 1618–1623. DOI: [10.1007/S00134-005-2825-8](https://doi.org/10.1007/S00134-005-2825-8).
- [7] Jean Louis Vincent and Rui Moreno. “Clinical review: Scoring systems in the critically ill.” In: *Critical Care* 14.2 (Mar. 2010). DOI: [10.1186/CC8204](https://doi.org/10.1186/CC8204).
- [8] Mervyn Singer et al. “The Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3).” In: *JAMA* 315.8 (Feb. 2016), pp. 801–810. ISSN: 0098-7484. DOI: [10.1001/JAMA.2016.0287](https://doi.org/10.1001/JAMA.2016.0287).
- [9] Chieh Liang Wu et al. “Implementation of an electronic national early warning system to decrease clinical deterioration in hospitalized patients at a tertiary medical center.” In: *International Journal of Environmental Research and Public Health* 18.9 (May 2021). ISSN: 16604601. DOI: [10.3390/IJERPH18094550/S1](https://doi.org/10.3390/IJERPH18094550/S1).

- [10] RCP. “Royal College of physicians. National early warning score (news) 2: standardising the assessment of acute-illness severity in the NHS. updated report of a working Party.” In: RCP London, 2017.
- [11] Ricardo Vinuesa et al. “The role of artificial intelligence in achieving the Sustainable Development Goals.” In: *Nature Communications* 11.1 (Dec. 2020). ISSN: 20411723. DOI: [10.1038/S41467-019-14108-Y](https://doi.org/10.1038/S41467-019-14108-Y).
- [12] Sherry L. Kausch et al. “Physiological machine learning models for prediction of sepsis in hospitalized adults: An integrative review.” In: *Intensive and Critical Care Nursing* 65 (Aug. 2021), p. 103035. ISSN: 0964-3397. DOI: [10.1016/J.ICCN.2021.103035](https://doi.org/10.1016/J.ICCN.2021.103035).
- [13] Matthieu Scherpf et al. “Predicting sepsis with a recurrent neural network using the MIMIC III database.” In: *Computers in Biology and Medicine* 113 (Oct. 2019), p. 103395. ISSN: 0010-4825. DOI: [10.1016/J.COMPBIOMED.2019.103395](https://doi.org/10.1016/J.COMPBIOMED.2019.103395).
- [14] Travis J. Moss et al. “Signatures of Subacute Potentially Catastrophic Illness in the ICU: Model Development and Validation.” In: *Critical Care Medicine* 44.9 (Sept. 2016), pp. 1639–1648. ISSN: 15300293. DOI: [10.1097/CCM.0000000000001738](https://doi.org/10.1097/CCM.0000000000001738).
- [15] Shameek Ghosh et al. “Septic shock prediction for ICU patients via coupled HMM walking on sequential contrast patterns.” In: *Journal of Biomedical Informatics* 66 (Feb. 2017), pp. 19–31. ISSN: 1532-0464. DOI: [10.1016/J.JBI.2016.12.010](https://doi.org/10.1016/J.JBI.2016.12.010).
- [16] Hye Jin Kam and Ha Young Kim. “Learning representations for the early detection of sepsis with deep neural networks.” In: *Computers in Biology and Medicine* 89 (Oct. 2017), pp. 248–255. ISSN: 18790534. DOI: [10.1016/J.COMPBIOMED.2017.08.015](https://doi.org/10.1016/J.COMPBIOMED.2017.08.015).
- [17] Ran Liu et al. “Data-driven discovery of a novel sepsis pre-shock state predicts impending septic shock in the ICU.” In: *Scientific Reports* 9.1 (Dec. 2019). ISSN: 20452322. DOI: [10.1038/S41598-019-42637-5](https://doi.org/10.1038/S41598-019-42637-5).
- [18] James Todd et al. “Improving mortality models in the ICU with high-frequency data.” In: *International Journal of Medical Informatics* 129 (Sept. 2019), pp. 318–323. ISSN: 1386-5056. DOI: [10.1016/J.IJMEDINF.2019.07.002](https://doi.org/10.1016/J.IJMEDINF.2019.07.002).
- [19] Reza Sadeghi, Tanvi Banerjee, and William Romine. “Early hospital mortality prediction using vital signals.” In: *Smart Health* 9-10 (Dec. 2018), pp. 265–274. ISSN: 23526483. DOI: [10.1016/J.SMHL.2018.07.001](https://doi.org/10.1016/J.SMHL.2018.07.001).
- [20] Kelser de Souza Kock and Jefferson Luiz Brum Marques. “Use of photoplethysmography to predict mortality in intensive care units.” In: *Vascular Health and Risk Management* 14 (2018), p. 311. ISSN: 11782048. DOI: [10.2147/VHRM.S172643](https://doi.org/10.2147/VHRM.S172643).

- [21] Mohammad Amin Morid, Olivia R. Liu Sheng, and Samir Abdelrahman. “PPMF: A Patient-based Predictive Modeling Framework for Early ICU Mortality Prediction.” In: (Apr. 2017).
- [22] Suparna Ghanvatkar and Vaibhav Rajan. “Deep Recurrent Neural Networks for Mortality Prediction in Intensive Care using Clinical Time Series at Multiple Resolutions.” In: *40th International Conference on Information Systems, ICIS 2019*. 2019.
- [23] Hans Christian Thorsen-Meyer et al. “Dynamic and explainable machine learning prediction of mortality in patients in the intensive care unit: a retrospective study of high-frequency data in electronic patient records.” In: *The Lancet Digital Health* 2.4 (Apr. 2020), e179–e191. ISSN: 2589-7500. DOI: [10.1016/S2589-7500\(20\)30018-2](https://doi.org/10.1016/S2589-7500(20)30018-2).
- [24] Aya Awad et al. “Predicting hospital mortality for intensive care unit patients: Time-series analysis.” In: *Health Informatics Journal* 26.2 (June 2020), pp. 1043–1059. ISSN: 17412811. DOI: [10.1177/1460458219850323](https://doi.org/10.1177/1460458219850323).
- [25] Akash Gupta, Tieming Liu, and Christopher Crick. “Utilizing time series data embedded in electronic health records to develop continuous mortality risk prediction models using hidden Markov models: A sepsis case study.” In: *Statistical Methods in Medical Research* 29.11 (Nov. 2020), pp. 3409–3423. ISSN: 14770334. DOI: [10.1177/0962280220929045](https://doi.org/10.1177/0962280220929045).
- [26] Xiaoran Li et al. “Deep learning prediction of likelihood of ICU admission and mortality in COVID-19 patients using clinical variables.” In: *PeerJ* 8 (Nov. 2020), e10337. ISSN: 21678359. DOI: [10.7717/PEERJ.10337/SUPP-2](https://doi.org/10.7717/PEERJ.10337/SUPP-2).
- [27] Logan Ryan et al. “Mortality prediction model for the triage of COVID-19, pneumonia, and mechanically ventilated ICU patients: A retrospective study.” In: *Annals of Medicine and Surgery* 59 (Nov. 2020), p. 207. ISSN: 20490801. DOI: [10.1016/J.AMSU.2020.09.044](https://doi.org/10.1016/J.AMSU.2020.09.044).
- [28] Irfan Ullah Khan et al. “Computational Intelligence-Based Model for Mortality Rate Prediction in COVID-19 Patients.” In: *International Journal of Environmental Research and Public Health* 2021, Vol. 18, Page 6429 18.12 (June 2021), p. 6429. ISSN: 16604601. DOI: [10.3390/IJERPH18126429](https://doi.org/10.3390/IJERPH18126429).
- [29] Md Mohaimenul Islam et al. “A State-of-the-Art Survey on Artificial Intelligence to Fight COVID-19.” In: *Journal of clinical medicine* 10.9 (May 2021). ISSN: 2077-0383. DOI: [10.3390/JCM10091961](https://doi.org/10.3390/JCM10091961).
- [30] Hassan Ismail Fawaz et al. “Deep learning for time series classification: a review.” In: *Data Mining and Knowledge Discovery* 33.4 (July 2019), pp. 917–963. ISSN: 1384-5810. DOI: [10.1007/s10618-019-00619-1](https://doi.org/10.1007/s10618-019-00619-1).

- [31] Ashish Vaswani et al. “Attention Is All You Need.” In: *Advances in Neural Information Processing Systems* 2017-December (June 2017), pp. 5999–6009. ISSN: 10495258. DOI: [10.48550/arxiv.1706.03762](https://doi.org/10.48550/arxiv.1706.03762).
- [32] Jackson Kamiri and Geoffrey Mariga. “Research Methods in Machine Learning: A Content Analysis.” In: *Article in International Journal of Computer and Information Technology* 10.2 (2021), pp. 2279–0764. DOI: [10.24203/ijcit.v10i2.79](https://doi.org/10.24203/ijcit.v10i2.79).
- [33] *Emergency use ICD codes for COVID-19 disease outbreak*. Date Accessed: 11/05/2022. URL: <https://www.who.int/standards/classifications/classification-of-diseases/emergency-use-icd-codes-for-covid-19-disease-outbreak>.
- [34] Leif Sörnmo and Pablo Laguna. “ECG Signal Processing.” In: *Bioelectrical Signal Processing in Cardiac and Neurological Applications* (Jan. 2005), pp. 453–566. DOI: [10.1016/B978-012437552-9/50007-6](https://doi.org/10.1016/B978-012437552-9/50007-6).
- [35] Hrishikesh Limaye and V. V. Deshmukh. “ECG noise sources and various noise removal techniques: A survey.” In: *International Journal of Application or Innovation in Engineering & Management* 5.2 (2016), pp. 86–92.
- [36] Hyun Jong Jang and Kyung Ok Cho. *Applications of deep learning for the analysis of medical data*. 2019. DOI: [10.1007/s12272-019-01162-9](https://doi.org/10.1007/s12272-019-01162-9).
- [37] Ana Mincholé et al. *Machine learning in the electrocardiogram*. 2019. DOI: [10.1016/j.jelectrocard.2019.08.008](https://doi.org/10.1016/j.jelectrocard.2019.08.008).
- [38] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 978-1-4414-1269-0.
- [39] Adam Paszke et al. “Pytorch: An imperative style, high-performance deep learning library.” In: *Advances in neural information processing systems* 32 (2019).
- [40] *Singularity User Guide 3.8 documentation*. Date Accessed: 16/05/2022. URL: <https://apptainer.org/user-docs/master/index.html>.
- [41] Bendong Zhao et al. “Convolutional neural networks for time series classification.” In: *Journal of Systems Engineering and Electronics* 28.1 (Feb. 2017), pp. 162–169. ISSN: 16711793. DOI: [10.21629/JSEE.2017.01.18](https://doi.org/10.21629/JSEE.2017.01.18).
- [42] Hassan Ismail Fawaz et al. “InceptionTime: Finding AlexNet for Time Series Classification.” In: *Data Mining and Knowledge Discovery* 34.6 (Sept. 2020), pp. 1936–1962. DOI: [10.1007/s10618-020-00710-y](https://doi.org/10.1007/s10618-020-00710-y).
- [43] Diederik Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” In: *International Conference on Learning Representations* (Dec. 2014). DOI: [10.48550/arxiv.1412.6980](https://doi.org/10.48550/arxiv.1412.6980).

- [44] Jocelyn S. Zhu et al. “Deep-learning artificial intelligence analysis of clinical variables predicts mortality in COVID-19 patients.” In: *Journal of the American College of Emergency Physicians Open* 1.6 (Dec. 2020), pp. 1364–1373. ISSN: 2688-1152. DOI: [10.1002/EMP2.12205](https://doi.org/10.1002/EMP2.12205).
- [45] Jianhong Cheng et al. “COVID-19 mortality prediction in the intensive care unit with deep learning based on longitudinal chest X-rays and clinical data.” In: *European Radiology* (Feb. 2022), pp. 1–11. ISSN: 0938-7994. DOI: [10.1007/s00330-022-08588-8](https://doi.org/10.1007/s00330-022-08588-8).
- [46] Marta Villegas et al. “Predicting the Evolution of COVID-19 Mortality Risk: a Recurrent Neural Network Approach.” In: *medRxiv* (Jan. 2021), pp. 2020–12. DOI: [10.1101/2020.12.22.20244061](https://doi.org/10.1101/2020.12.22.20244061).
- [47] Jacob Calvert et al. “Using electronic health record collected clinical variables to predict medical intensive care unit mortality.” In: *Annals of Medicine and Surgery* 11 (Nov. 2016), pp. 52–57. ISSN: 20490801. DOI: [10.1016/J.AMSU.2016.09.002](https://doi.org/10.1016/J.AMSU.2016.09.002).
- [48] Chao-Chen Chen and Fuchiang Rich Tsui. “Comparing different wavelet transforms on removing electrocardiogram baseline wanders and special trends.” In: *BMC Medical Informatics and Decision Making* 20 (2020). DOI: [10.1186/s12911-020-01349-x](https://doi.org/10.1186/s12911-020-01349-x).
- [49] Zhuoyuan Zheng, Yunpeng Cai, and Ye Li. “Oversampling method for imbalanced classification.” In: *Computing and Informatics* 34.5 (2015).
- [50] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. “A systematic study of the class imbalance problem in convolutional neural networks.” In: *Neural Networks* 106 (2018). DOI: [10.1016/j.neunet.2018.07.011](https://doi.org/10.1016/j.neunet.2018.07.011).
- [51] Scott M Lundberg and Su-In Lee. “A unified approach to interpreting model predictions.” In: *Advances in neural information processing systems* 30 (2017).
- [52] Thomas Rojat et al. “Explainable Artificial Intelligence (XAI) on TimeSeries Data: A Survey.” In: (Apr. 2021).
- [53] Yanbo Xu et al. “RAIM: Recurrent attentive and intensive model of multimodal patient monitoring data.” In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2018. DOI: [10.1145/3219819.3220051](https://doi.org/10.1145/3219819.3220051).
- [54] Benjamin Moody et al. “MIMIC-III Waveform Database (version 1.0).” In: *PhysioNet* 3 (2020). ISSN: 20524463. DOI: [10.13026/c2607m](https://doi.org/10.13026/c2607m).
- [55] Pieter Kubben, Michel Dumontier, and Andre Dekker. *Fundamentals of Clinical Data Science*. Ed. by Pieter Kubben, Michel Dumontier, and Andre Dekker. Cham: Springer International Publishing, 2019. ISBN: 978-3-319-99712-4. DOI: [10.1007/978-3-319-99713-1](https://doi.org/10.1007/978-3-319-99713-1).

- [56] Reed T. Sutton et al. “An overview of clinical decision support systems: benefits, risks, and strategies for success.” In: *npj Digital Medicine* 3.1 (Dec. 2020), p. 17. DOI: [10.1038/s41746-020-0221-y](https://doi.org/10.1038/s41746-020-0221-y).
- [57] Arnaud Belard et al. “Precision diagnosis: a view of the clinical decision support systems (CDSS) landscape through the lens of critical care.” In: *Journal of Clinical Monitoring and Computing* 31.2 (Apr. 2017), pp. 261–271. DOI: [10.1007/s10877-016-9849-1](https://doi.org/10.1007/s10877-016-9849-1).
- [58] Amit Sapra, Ahmad Malik, and Priyanka Bhandari. “Vital Sign Assessment.” In: *StatPearls* (May 2021).
- [59] Christina Orphanidou. *Signal Quality Assessment in Physiological Monitoring*. SpringerBriefs in Bioengineering. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-68414-7. DOI: [10.1007/978-3-319-68415-4](https://doi.org/10.1007/978-3-319-68415-4).
- [60] Paul Ellis Marik. “Understanding the Vital Signs: BP, HR, RR, TEMP, SaO2 ... and SV.” In: *Evidence-Based Critical Care*. Cham: Springer International Publishing, 2015, pp. 169–196. DOI: [10.1007/978-3-319-11020-2\\_{\\\_}14](https://doi.org/10.1007/978-3-319-11020-2_{\_}14).
- [61] Hwee Ming Cheng and Felicita Jusof. *Defining Physiology: Principles, Themes, Concepts*. Singapore: Springer Singapore, 2018. ISBN: 978-981-13-0498-9. DOI: [10.1007/978-981-13-0499-6](https://doi.org/10.1007/978-981-13-0499-6).
- [62] Edward D. Chan, Michael M. Chan, and Mallory M. Chan. “Pulse oximetry: Understanding its basic principles facilitates appreciation of its limitations.” In: *Respiratory Medicine* 107.6 (June 2013), pp. 789–799. DOI: [10.1016/J.RMED.2013.02.004](https://doi.org/10.1016/J.RMED.2013.02.004).
- [63] Christina Orphanidou et al. “Telemetry-based vital sign monitoring for ambulatory hospital patients.” In: *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, Sept. 2009, pp. 4650–4653. DOI: [10.1109/IEMBS.2009.5332649](https://doi.org/10.1109/IEMBS.2009.5332649).
- [64] Alvin Har Teck Chia et al. “Explainable machine learning prediction of ICU mortality.” In: *Informatics in Medicine Unlocked* 25 (Jan. 2021), p. 100674. DOI: [10.1016/J.IMU.2021.100674](https://doi.org/10.1016/J.IMU.2021.100674).
- [65] Pasquale Pagliaro, Claudia Penna, and Raffaella Rastaldo. *Basic Cardiovascular Physiology: From Molecules to Translational Medical Science*. River Publishers, 2020.
- [66] Steve Meek and Francis Morris. “ABC of clinical electrocardiography: Introduction. I—Leads, rate, rhythm, and cardiac axis.” In: *BMJ : British Medical Journal* 324.7334 (Feb. 2002), p. 415. ISSN: 14685833. DOI: [10.1136/BMJ.324.7334.415](https://doi.org/10.1136/BMJ.324.7334.415).

- [67] Johannes W. Krug et al. “ECG-based gating in ultra high field cardiovascular magnetic resonance using an independent component analysis approach.” In: *Journal of Cardiovascular Magnetic Resonance* 15.1 (Nov. 2013). DOI: [10.1186/1532-429X-15-104](https://doi.org/10.1186/1532-429X-15-104).
- [68] Cornel Pater. “Methodological considerations in the design of trials for safety assessment of new drugs and chemical entities.” In: *Current Controlled Trials in Cardiovascular Medicine* 6 (Feb. 2005). DOI: [10.1186/1468-6708-6-1](https://doi.org/10.1186/1468-6708-6-1).
- [69] Karim Bendjelid. “The pulse oximetry plethysmographic curve revisited.” In: *Current Opinion in Critical Care* 14.3 (June 2008), pp. 348–353. DOI: [10.1097/MCC.0b013e3282fb2dc9](https://doi.org/10.1097/MCC.0b013e3282fb2dc9).
- [70] Seyedeh Somayyeh Mousavi et al. “Blood pressure estimation from appropriate and inappropriate PPG signals using A whole-based method.” In: *Biomedical Signal Processing and Control* 47 (Jan. 2019), pp. 196–206. ISSN: 1746-8094. DOI: [10.1016/J.BSPC.2018.08.022](https://doi.org/10.1016/J.BSPC.2018.08.022).
- [71] Chadi El-Hajj and Panayiotis A. Kyriacou. “A review of machine learning techniques in photoplethysmography for the non-invasive cuff-less measurement of blood pressure.” In: *Biomedical Signal Processing and Control* 58 (Apr. 2020), p. 101870. DOI: [10.1016/j.bspc.2020.101870](https://doi.org/10.1016/j.bspc.2020.101870).
- [72] Kinjarapu Manojkumar, Srinivas Boppu, and M. Sabarimalai Manikandan. “An Automated Algorithm for Estimating Respiration Rate from PPG Signals.” In: *Communications in Computer and Information Science* 1241 CCIS (July 2020), pp. 44–57. ISSN: 18650937. DOI: [10.1007/978-981-15-6318-8\\_{\\\_}5](https://doi.org/10.1007/978-981-15-6318-8_{\_}5).
- [73] Daryl Jones et al. “Defining clinical deterioration.” In: *Resuscitation* 84.8 (Aug. 2013), pp. 1029–1034. DOI: [10.1016/J.RESUSCITATION.2013.01.013](https://doi.org/10.1016/J.RESUSCITATION.2013.01.013).
- [74] Meera Joshi et al. “Wearable sensors to improve detection of patient deterioration.” In: *Expert Review of Medical Devices* 16.2 (Feb. 2019), pp. 145–154. DOI: [10.1080/17434440.2019.1563480](https://doi.org/10.1080/17434440.2019.1563480).
- [75] Alexander Jung. *Machine Learning*. Machine Learning: Foundations, Methodologies, and Applications. Singapore: Springer Singapore, 2022. ISBN: 978-981-16-8192-9. DOI: [10.1007/978-981-16-8193-6](https://doi.org/10.1007/978-981-16-8193-6).
- [76] Tianqi Chen and Carlos Guestrin. “XGBoost: A scalable tree boosting system.” In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 13-17-August-2016* (Aug. 2016), pp. 785–794. DOI: [10.1145/2939672.2939785](https://doi.org/10.1145/2939672.2939785).
- [77] Courville Aaron Goodfellow Ian Bengio Yoshua. *Deep Learning*. 2016. URL: <http://www.deeplearningbook.org>.

- [78] Carolin Danker et al. “AI and Dynamic Prediction of Deterioration in Covid-19.” In: *Artificial Intelligence in COVID-19*. Ed. by Niklas Lidströmer and Y Eldar. Springer Nature, 2022. ISBN: 978-3-031-08505-5.
- [79] Zahra Ebrahimi et al. “A review on deep learning methods for ECG arrhythmia classification.” In: *Expert Systems with Applications: X* 7 (Sept. 2020), p. 100033. ISSN: 2590-1885. DOI: [10.1016/J.ESWAX.2020.100033](https://doi.org/10.1016/J.ESWAX.2020.100033).
- [80] Muhammed Sit et al. “A Comprehensive Review of Deep Learning Applications in Hydrology and Water Resources.” In: *Water Science and Technology* 82.12 (2020), pp. 2635–2670.
- [81] Tyler J Loftus ID et al. “Ideal algorithms in healthcare: Explainable, dynamic, precise, autonomous, fair, and reproducible.” In: *PLOS Digital Health* 1.1 (Jan. 2022). DOI: [10.1371/JOURNAL.PDIG.0000006](https://doi.org/10.1371/JOURNAL.PDIG.0000006).
- [82] Dongwan Kim et al. “The Architecture of SARS-CoV-2 Transcriptome.” In: *Cell* 181.4 (May 2020), pp. 914–921. URL: [https://www.cell.com/cell/abstract/S0092-8674\(20\)30406-2](https://www.cell.com/cell/abstract/S0092-8674(20)30406-2).
- [83] Esmail Mehraeen et al. “Predictors of mortality in patients with COVID-19—a systematic review.” In: *European Journal of Integrative Medicine* 40 (Dec. 2020). ISSN: 18763839. DOI: [10.1016/J.EUJIM.2020.101226](https://doi.org/10.1016/J.EUJIM.2020.101226).
- [84] Jian Min Jin et al. “Gender Differences in Patients With COVID-19: Focus on Severity and Mortality.” In: *Frontiers in Public Health* 8 (Apr. 2020), p. 152. ISSN: 22962565. DOI: [10.3389/FPUBH.2020.00152/BIBTEX](https://doi.org/10.3389/FPUBH.2020.00152/BIBTEX).
- [85] Wenjie Tian et al. “Predictors of mortality in hospitalized COVID-19 patients: A systematic review and meta-analysis.” In: *Journal of Medical Virology* 92.10 (Oct. 2020), pp. 1875–1883. ISSN: 10969071. DOI: [10.1002/JMV.26050](https://doi.org/10.1002/JMV.26050).
- [86] Prathamesh Parchure et al. “Development and validation of a machine learning-based prediction model for near-term in-hospital mortality among patients with COVID-19 PT and AK contributed equally.” In: *BMJ Supportive & Palliative Care* 0 (2020), pp. 1–8. DOI: [10.1136/bmjspcare-2020-002602](https://doi.org/10.1136/bmjspcare-2020-002602).
- [87] Li Yan et al. “An interpretable mortality prediction model for COVID-19 patients.” In: *Nature Machine Intelligence* 2.5 (May 2020), pp. 283–288. DOI: [10.1038/S42256-020-0180-7](https://doi.org/10.1038/S42256-020-0180-7).
- [88] Akshaya Karthikeyan et al. “Machine Learning Based Clinical Decision Support System for Early COVID-19 Mortality Prediction.” In: *Frontiers in Public Health* 9 (May 2021), p. 626697. ISSN: 22962565. DOI: [10.3389/FPUBH.2021.626697/FULL](https://doi.org/10.3389/FPUBH.2021.626697/FULL).
- [89] Arjun S. Yadaw et al. “Clinical features of COVID-19 mortality: development and validation of a clinical prediction model.” In: *The Lancet. Digital Health* 2.10 (Oct. 2020), e516. ISSN: 25897500. DOI: [10.1016/S2589-7500\(20\)30217-X](https://doi.org/10.1016/S2589-7500(20)30217-X).

- [90] Guangyao Wu et al. “Development of a clinical decision support system for severity risk prediction and triage of COVID-19 patients at hospital admission: an international multicentre study.” In: *European Respiratory Journal* 56.2 (Aug. 2020). ISSN: 0903-1936. DOI: [10.1183/13993003.011104-2020](https://doi.org/10.1183/13993003.011104-2020).
- [91] Manuel Sánchez-Montañés et al. “Machine Learning for Mortality Analysis in Patients with COVID-19.” In: *International Journal of Environmental Research and Public Health* 2020, Vol. 17, Page 8386 17.22 (Nov. 2020), p. 8386. ISSN: 1660-4601. DOI: [10.3390/IJERPH17228386](https://doi.org/10.3390/IJERPH17228386).
- [92] Hoyt Burdick et al. “Prediction of respiratory decompensation in Covid-19 patients using machine learning: The READY trial.” In: *Computers in Biology and Medicine* 124 (Sept. 2020), p. 103949. ISSN: 18790534. DOI: [10.1016/J.COMPBIOMED.2020.103949](https://doi.org/10.1016/J.COMPBIOMED.2020.103949).
- [93] Siavash Bolourani et al. “A machine learning prediction model of respiratory failure within 48 hours of patient admission for COVID-19: Model development and validation.” In: *Journal of Medical Internet Research* 23.2 (2021). ISSN: 14388871. DOI: [10.2196/24246](https://doi.org/10.2196/24246).
- [94] Shenda Hong et al. “Opportunities and challenges of deep learning methods for electrocardiogram data: A systematic review.” In: *Computers in Biology and Medicine* 122 (July 2020), p. 103801. ISSN: 18790534. DOI: [10.1016/J.COMPBIOMED.2020.103801](https://doi.org/10.1016/J.COMPBIOMED.2020.103801).
- [95] Runnan He et al. “Automatic Cardiac Arrhythmia Classification Using Combination of Deep Residual Network and Bidirectional LSTM.” In: *IEEE Access* 7 (2019), pp. 102119–102135. ISSN: 2169-3536. DOI: [10.1109/ACCESS.2019.2931500](https://doi.org/10.1109/ACCESS.2019.2931500).
- [96] Kunyang Li et al. “A method to detect sleep apnea based on deep neural network and hidden Markov model using single-lead ECG signal.” In: *Neurocomputing* 294 (June 2018), pp. 94–101. ISSN: 0925-2312. DOI: [10.1016/J.NEUCOM.2018.03.011](https://doi.org/10.1016/J.NEUCOM.2018.03.011).

# Appendix A

## State of The Art

This chapter presents relevant background information and state of the art for the application of machine learning on continuous time series data in the clinical context. More specifically, the thesis deals with the development of a clinical decision support system for predicting mortality in patients with coronavirus disease 19 (Covid-19).

Section A.1 contains background information about clinical data and clinical decision support systems. Section A.2 provides basic knowledge of physiological monitoring. After that, relevant machine learning and deep learning methods to analyse time series are reviewed in section A.3. Last, prediction methods for Covid-19 mortality as well as other related studies are presented in section A.4.

### A.1 Clinical Data

Clinical data is the basis for hospital decision-making, and its amount and quality are growing quickly [55]. This section briefly summarizes types and sources of clinical data (section A.1.1) and the idea of clinical decision support systems (section A.1.2).

#### A.1.1 Data Types and Sources

Clinical data can have a variety of shapes and sources. In general, there can be distinguished between static data (e.g. gender, birth) and dynamic data that consists of multiple records and can change over time (e.g. laboratory (lab) results, vital signs) [21]. Due to the diverse nature, clinical data is saved in different ways. One type is tabular data that presents data in a column-row format. A specific type is time-series data in which columns represent data at different time stamps. Other data types are natural language in free text format (e.g. notes or reports), images or videos [55].

Clinical data can be obtained from different data sources. The most important sources are Electronic Medical Records (EMRs), an electronic version of patient information. The term EMR is often interchangeable used with Electronic Health Record (EHR) [55].

EMRs contain a variety of data, including lab values, medical images, but also physicians' notes and data obtained from other systems, such as scheduling systems. Other medical information systems that can be used as data sources are laboratory information systems, systems for radiology, and information from external care [55]. Another important source of clinical data are bedside monitors, which record high-resolution signals with a higher frequency than sequential data in EMR [22].

## **A.1.2 Clinical Decision Support Systems**

Currently, decisions in the hospital are often based on experience and knowledge of the physician and less on data-driven approaches [55]. Clinical Decision Support Systems (CDSSs) can be used to support physicians in their decision making [56]. CDSSs can have several application areas, including clinical management, diagnostics support, and patient decision support [56]. However, a lot of potential of the amounts of stored data is still unused [55]. High-quality CDSS could improve patient care and decrease costs [57]. But there are many challenges in developing such systems. First, lack of enough high-quality data to develop and validate systems [57]. Second, ethical and legal issues [57]. And third, skepticism of the physicians towards systems that don't explain results properly [56].

## **A.2 Physiological Monitoring**

A big amount of clinical data is gathered from physiological monitoring. Physiological monitoring is used to control the health status of the patient to prevent patient deterioration at an early stage and thus to avoid death. This section gives a brief overview of vital signs (section A.2.1), continuously monitored data (section A.2.2) and patient deterioration (section A.2.3).

### **A.2.1 Vital Signs**

Vital signs such as heart rate (HR), blood pressure (BP), respiratory rate (RR), peripheral oxygen saturation (SpO<sub>2</sub>), and temperature are used to measure and assess a patient's physiological state [58, 59]. Based on an analysis of vital signs, the treatment strategy can be defined and adapted to abnormalities indicating risks [60].

The HR is defined as the number of cardiac cycles per minute [61]. A cardiac cycle consists of a systolic phase (contraction of the heart muscle) and a diastolic phase (relaxation). An arterial pulse can be felt during the systolic phase, thus, the pulse rate corresponds to the HR. The second vital sign is BP. A distinction must be made between systolic BP (SBP) and diastolic BP (DBP). SBP is higher and describes the pressure during the contraction of the heart. Furthermore, it can be distinguished between arterial and venous blood pressure. Most commonly the mean arterial pressure (MAP) is used as a clinical feature. BP varies between people and can be influenced by several factors,

such as age, genes, or lifestyle [60]. Next, the RR is defined as the number of breaths per minute. Both a higher rate (tachypnea) and a lower rate (bradypnea) may have pathological causes [58]. To detect abnormalities, the RR is assessed in combination with the respiratory depth and pattern [58, 60]. The fourth vital sign is SpO<sub>2</sub> which is typically measured by pulse oximeters [62]. The measurement of SpO<sub>2</sub> can be affected by various physiological (e.g. poor perfusion), pathophysiological (e.g. carbon monoxide poisoning), and external factors (e.g. movement, venous pulsations) [62]. The last frequently used vital sign is temperature, which can be influenced by several internal and external factors such as diurnal and menstrual variations, physical fitness, or age [58].

## A.2.2 Continuous Physiological Monitoring

The nursing staff is responsible to check the vital signs manually every few hours [63]. However, a continuous monitoring is required in critical situations, e.g. on the intensive care units (ICUs), to detect abnormalities as early as possible [64]. This can be done using the electrocardiogram (section A.2.2.1) and the photoplethysmogram (section A.2.2.2).

### A.2.2.1 Electrocardiogram

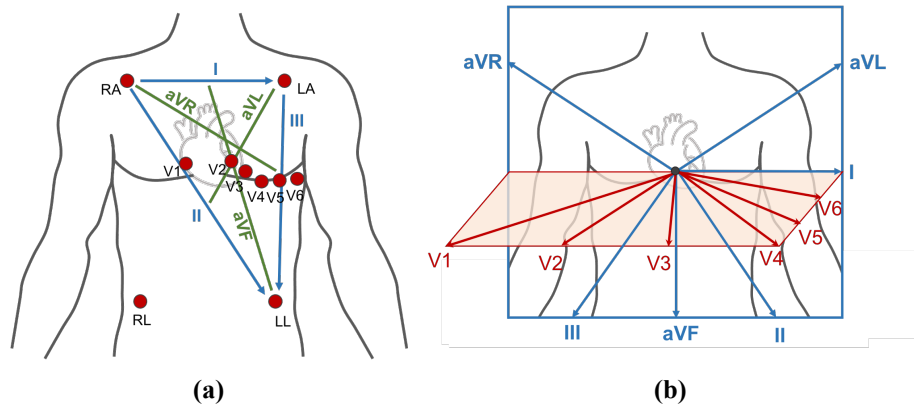
The electrocardiogram (ECG) is the most frequently continuously recorded signal and provides essential information about the heart condition. A standard ECG consists of 12 leads, but also just three or five leads can be used [59]. One lead measures the potential difference between two electrodes that are attached to the upper body of the patient (Figure A.1a). The potential differences are caused by electric fields generated during the contraction movement of the heart muscle. Depending on the position of the electrodes a different part of the heart is recorded.

An ECG wave displays the measured potential over time and consists of several waves and intervals displaying different phases during the cardiac cycle (Figure A.2). The most dominant wave is the "QRS complex", which results from the contraction of the heart ventricles. From the ECG wave, different features can be derived. To obtain the HR, the distance between two R-waves is measured (R-R-interval). Another often-used measure is the heart rate variability which is the variation of the R-R-intervals [65].

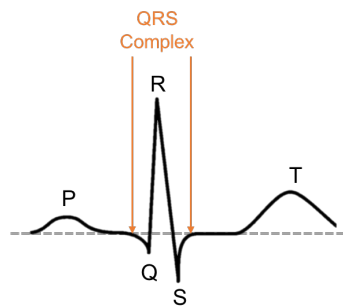
To display the cardiac activity in detail, several leads are needed. The leads are divided into the limb leads (I, II, III, aVR, aVL, aVF), that represent different vertical views through the heart, and chest leads (V1-V6), for horizontal views (Figure A.1b) [65]. The leads record different parts of the heart [66]:

- **II, III, aVF:** Inferior heart surface
- **V2-V4:** Anterior heart surface
- **I, aVL, V5, V6:** Lateral heart surface
- **V1, aVR:** Right atrium, cavity of the left ventricle

The wave morphology varies depending on the lead. For example, the amplitude of the QRS complex is greater the closer the electrode is placed to the heart. Different ECG wave parts and leads can provide evidence of pathologic findings in different parts of the heart [59]. However, measurements may vary depending on the individual, electrode type, and placement [59].



**Figure A.1.** (a) Electrode Placements for a standard 12-lead ECG. Four electrodes are placed on the limbs (RA: right arm; LA: left arm; RL: right leg; LL: left leg), which give the limb leads (I, II, III, aVR, aVL, aVF). Six electrodes are placed on the chest (V1-V6). Adapted from [67]. (b) Lead perspectives. The limb leads lay on a vertical plane while the chest leads lay on a horizontal plane. Adapted from [66].

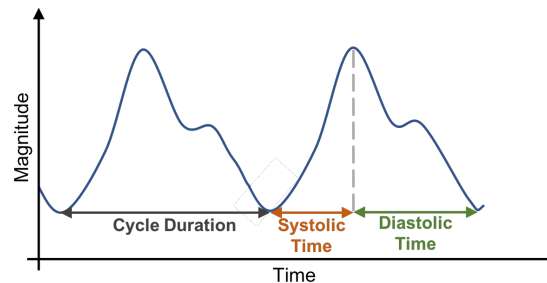


**Figure A.2.** ECG wave morphology. One cardiac cycle includes the P wave, the QRS complex and the T wave. Adapted from [68]

### A.2.2.2 Photoplethysmogram

Another continuously monitored physiological signal is the photoplethysmogram (PPG), an optical measurement technique to measure the blood volume in the periphery. Due to changes in blood volume during the cardiac cycle, a different amount of light is absorbed by the tissue, from which a pulse waveform can be derived (Figure A.3).

Wave morphology can change depending on the individual, measurement location, and attachment [69].



**Figure A.3.** PPG wave morphology. Adapted from [70].

The PPG is influenced by cardiovascular and respiratory activity and therefore provides inferences about various vital signs, although the complete explanation of the wave origin has not yet been fully understood [71]. The heart systole leads to an increased amount of blood volume and thus a rising of the PPG wave, while the heart diastole leads to a declining PPG wave [59]. The pulse can thus be directly derived from the signal. Additionally, respiration influences the PPG waveform in rate, amplitude, and width [72]. The  $SpO_2$  calculation is possible due to the use of the optical measurement method [59]. Last, deriving BP from the PPG signal is possible but not trivial. Recent studies have shown that non-linear models based on PPG alone or in combination with an ECG signal can provide information about the BP [71].

### A.2.3 Patient Deterioration

Abnormal vital signs can indicate clinical deterioration in real-time and can therefore help to detect cardiac, respiratory, shock, and sepsis [73, 74]. If patient deterioration stays undetected or is detected too late, it can lead to death [73]. In this context, it is particularly important to consider sepsis and septic shock. Sepsis is a life-threatening syndrome that is caused by an infection and coupled with a dysregulated patient response and organ dysfunction [8]. Septic shock is a form of sepsis in which acute abnormalities of the circulation and cellular metabolism lead to an even higher risk of death [8]. With the help of the Sequential Organ Failure Assessment (SOFA) score, which is taking into account respiration, coagulation, liver, cardiovascular, nervous, and renal values, the risk of sepsis-related mortality can be assessed.

## A.3 Machine Learning for Clinical Data

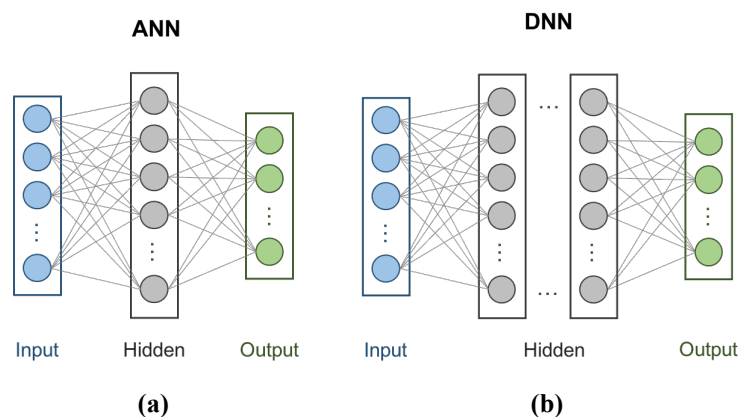
This section briefly presents common machine learning (section A.3.1) and deep learning approaches (section A.3.2) used to process clinical data. Thereafter, explainability for algorithms in healthcare is addressed (section A.3.3).

### A.3.1 Machine Learning Approaches

A subfield of artificial intelligence is machine learning (ML). A ML model is used to make a prediction, which can be a numeric value (regression) or a category (classification), given specific input data. By minimizing the difference between the true values and the predictions, model weights are trained [75]. A common method for a regression problem is linear regression, where the prediction is simply made using a linear function for given input parameters. Instead of a linear function a logistic function can be used to make a binary classification, which is called logistic regression (LoR). Another non-linear classification is made using support vector machines (SVM). Non-linear decision boundaries between classes are trained by using kernel functions [75]. A different method for classification is a decision tree (DTs). The prediction of DTs are often combined in ensembles named random forest (RF) to avoid overfitting [75]. Optionally, boosting can be used to train an RF, for example with XGBoost (Extreme Gradient Boosting) [76]. Last, to process sequential data Hidden Hidden-Markov Models (HMM) can be used.

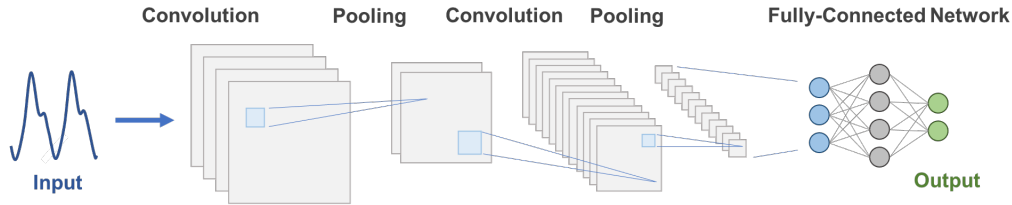
### A.3.2 Deep Learning Approaches

Deep Learning (DL) is a subcategory of ML that deals with the application of artificial neural networks (ANN) to solve complex problems where other methods fail [77]. Figure A.4 shows the basic structure of an ANN which is based on biological neural networks and consist of an input layer, a hidden layer and an output layer [75]. If several hidden layers are trained the model is referred to as a deep neural network (DNN) [75].



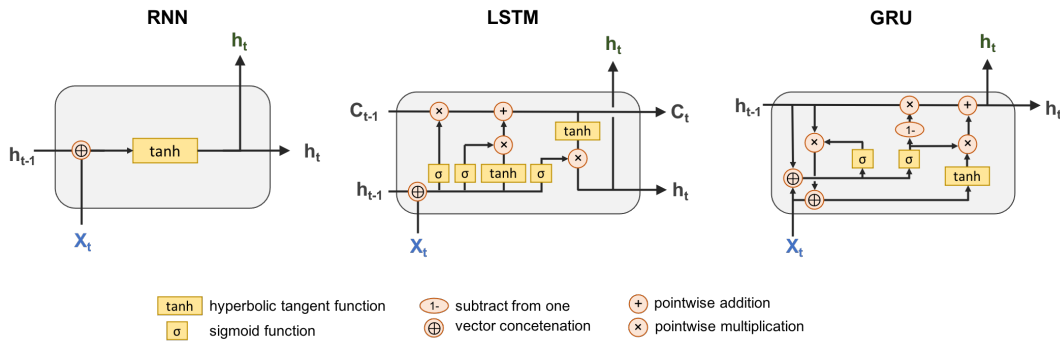
**Figure A.4.** Basic architectures of an Artificial Neural Network and a Deep Neural Network [78].

One category of neural networks are convolutional neural networks (CNN). CNNs were developed to process grid-like topologies like time-series data (1D) and images (2D). Figure A.5 shows an exemplary architecture of a CNN. By applying convolutions, pooling, and activation functions, the model is able to create abstract features and use those for a classification [77].



**Figure A.5.** Example of a convolutional neural network architecture for time-series data. The input is time-series data, convolution and pooling steps are applied before computing an output by a fully connected layer. [78], adapted from [79].

Another model category are recurrent neural networks (RNN) that are developed for sequential data (Figure A.6). The models are able to make predictions for every time step taking into account previous predictions. To be able to consider long-term dependencies, more complex architectures were developed such as the long short-term memory architecture (LSTM) and gated recurrent unit (GRU) [77].



**Figure A.6.** Comparison of RNN, LSTM and GRU cell architectures. Adapted from [80].

### A.3.3 Explainability of Algorithms

Loftus ID et al. presents six desirable characteristics for clinical algorithms: explainable, dynamic, precise, autonomous, fair, and reproducible [81]. In particular, explainability is relevant, as physicians and patients want to understand why a decision is made. The main problem with ML and DL approaches is their black-box character, which makes them very difficult to understand [81]. Several explainability methods have been developed in the past. One widely used method is SHapley Additive exPlanation (SHAP) which provides a score for the importance of each input feature [51].

## A.4 Prediction Models

This section presents relevant prediction models for healthcare. Population, type of data, used method, and aim of the models differ a lot. The primary focus is on relevant literature for mortality prediction. First, traditional scoring systems are reviewed (section A.4.1). Afterwards, more advanced mortality prediction models are presented using Covid-19 data (section A.4.2) and other datasets (section A.4.3). Lastly, related studies about other prediction models are summarized (section A.4.4).

### A.4.1 General Prediction Scores

Predictions scores can be used to assess the health of a patient when entering the ICU [6]. Mainly generic scores are applied to either predict an outcome based on the assessment or to evaluate the severity of organ dysfunction, but also disease-specific scores exist [7]. Three score systems became popular over the last years to predict mortality for a patient when entering the ICU: the Simplified Acute Physiology Score (SAPS), the Acute Physiology and Chronic Health Evaluation (APACHE), and the Mortality Probability Model (MPM) [6]. A different number of variables are used, including demographics and physiological variables. Eventually, LoR is used to predict the outcome. Another common score is the already mentioned SOFA score, an organ-specific score for indicating sepsis-related mortality (see section A.2.3) [8]. Several other early warning systems have been developed that require a manual calculation based on thresholds for measured vital signs [9]. A common example is the National Early Warning Score that requires six physiological parameters to provide a warning score for clinical deterioration [10]. Even though the scores are widely used, they lack accuracy on individual patient level [21, 24]. The result is only reliable if the patient has similar characteristics to the population used for the development of the scores [7].

### A.4.2 Prediction Models for Covid-19

Covid-19 is caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) [82]. Since the beginning of the Covid-19 outbreak, several factors have been associated with a severe outbreak and increased mortality [83]. Demographic factors that increase the risk are in particular higher age but also male gender [84]. Other factors are comorbidities (e.g. hypertension, diabetes, or cardiovascular diseases) and results of laboratory tests [85]. Patient risk assessment became important because of the large impact on healthcare systems and in terms of utilization of resources. [85].

To improve general prediction scores and be able to provide patient-specific care, several ML and DL models were developed to predict mortality for Covid-19 patients [28, 29]. The models use demographics, vital signs, comorbidities, medication, laboratory values, and imaging (X-ray, CT) [29]. Table A.1 summarizes relevant studies using DL

approaches. Li et al. aimed to identify key features to predict ICU admission and in-hospital mortality [26]. Dynamic features were aggregated to a single value. To choose relevant clinical variables, a RF algorithm is applied. The top predictors for mortality include age, SpO<sub>2</sub> and lab values and comorbidities. A DNN with five fully connected dense layers has been used to predict ICU admission and mortality, respectively. The same researchers published a similar study using a smaller study population and different DNN architecture for feature selection and classification [44]. Five key predictors were identified, the oxygen index, and four lab values. Villegas et al. uses an ensemble of RNNs to predict mortality for two datasets [46]. The model uses both static and dynamic data, with the dynamic data aggregated to one variable per day. The model uses LSTMs and GRUs and an attention layer was added for better interpretability. Another model has been developed by Cheng et al. using EMR data together with longitudinal X-rays [45]. All continuous EMR values (vital signs, lab values) are binarized. The clinical data is then processed by an ANN with three fully connected layers, while the X-ray images are fed into a longitudinal transformer-based network. The final prediction is made by a weighted sum of both models using a fully connected layer. Important features identified are age, comorbidities, and lab values, confirming the results from the other studies.

**Table A.1.** Comparison of relevant studies applying deep learning methods to Covid-19 populations to predict mortality. Adapted from [78].

Author	Aim	Size of Study Population	Input Parameters	Model	Results
Li [26]	Identify top predictors, predict ICU admission and in-hospital mortality	1,108	DG, CM, VS, lab, symptoms	DNN	<b>Mortality:</b> AUC: 0.844 ACC: 0.853
Zhu [44]	Identify top predictors, predict mortality	181	DG, CM, VS, lab, symptoms	DNN	AUC: 0.968
Villegas [46]	Mortality prediction	6,087	DG, VS, lab, medication	Ensemble of RNNs	<b>Dataset 1:</b> ACC: 0.890 <b>Dataset 2:</b> ACC: 0.901
Cheng [45]	Predict in-hospital mortality of ICU patients	654	DG, CM, VS, lab, X-rays	DNN, LTBN	<b>Clinical Data:</b> AUC: 0.653 ACC: 0.657 <b>With X-Ray:</b> AUC: 0.727 ACC: 0.732

*Note:* DG: Demographics; CM: Comorbidities; VS: Vital Signs; lab: Laboratory Test Values; DNN: Deep Neural Network; RNN: Recurrent Neural Network; LTBN: Longitudinal transformer-based network; AUC: Area under the curve; ACC: Accuracy.

Other studies applied ML approaches other than DL. Ryan et al. predicts mortality and the need for mechanical ventilation using time series data [27]. Used features include age, HR, RR, SpO<sub>2</sub>, temperature, BP, and lab values. Some of the features were directly generated from monitored raw data. Measurements were averaged and discretized into one-hour intervals. A XGBoost model was used to predict mortality 12, 24, 48, and 72 hours before death resulting in AUC of 0.91, 0.90, 0.86, and 0.87, respectively. A similar study has been conducted by Parchure et al. [86]. Several features including demographics (age, sex), vital signs, respiratory patterns, ECG-derived variables (P wave axis, PR interval, QRS duration) were used to predict death within 20 to 84 hours from the time of prediction. The last three observations before the prediction window started were used as input for an RF algorithm. The model achieved an AUC of 0.86.

Many other studies exist that differ in terms of input features and algorithms. Yan et al. identified relevant indicators of blood samples using XGBoost models [87]. Karthikeyan et al. also uses blood samples to identify key features and to predict mortality using a neural network [88]. Yadaw et al. developed a prediction model using just three commonly available clinical features: age, oxygen saturation, and type of patient encounter using an XGBoost model [89]. Wu et al. developed a prediction model based on CT images, laboratory and clinical features such as age and temperature using LoR [90]. Furthermore, Sánchez-Montañés et al. identifies age, gender, oxygen saturation, and HR, among others, as relevant factors for predicting mortality using survival analysis, RF, and LoR [91].

Some studies focus specifically on prediction close to admission. Burdick et al. developed a XGBoost model using vital signs measured in the first two hours of hospital admission [92]. Compared to other studies the aim was to predict the need for mechanical ventilation within the next 24 hours to provide short-term care. Likewise, Bolourani et al. was able to improve respiratory failure prediction within the first two days of admissions compared to traditionally used score systems [93]. Two XGBoost based models were developed using several features including vital signs, laboratory values, and demographics.

### **A.4.3 Mortality Prediction Models**

The above presented studies for Covid-19 populations use mainly demographics and laboratory values; features from waveform data are just rarely included. Several models using dynamic features have been developed for non-Covid-19 populations and can be classified as non-temporal (aggregation of dynamic features) or continuous prediction models (Table A.2).

In non-temporal models dynamic features are aggregated. For example, Todd et al. processes pulse and MAP waveforms of the first 24 hours after admission [18]. Five features are calculated (average, standard deviation, skewness, kurtosis, and proportion of turning points) and used together with static features (SAPS, age, comorbidity) for a LoR classification. Similarly, Sadeghi, Banerjee, and Romine calculates the HR from

ECG waveforms and extracts quantitative features [19]. Twelve statistical signal-based features from the first hour of ICU admission are used. Of the eight models applied, DT and RF give the best results. Kock and Marques predicts mortality based on static features (APACHE II score, diagnosis, age, gender, clinical profile) and a one-minute PPG signal [20]. Features are extracted from the PPG signal and its first two derivatives, and the median values over the entire interval are calculated. Using LoR and ANNs, it was shown that the PPG features improve the result of the model significantly. Morid, Sheng, and Abdelrahman on the other hand does not aggregate dynamic features over the entire time interval but computes one feature for every 2-hours-interval of 48 hours of time-series data, that are then fed into a k-nearest neighbour classifier [21].

In contrast, temporal models make a continuous prediction. Ghanvatkar and Rajan, for instance, develops a multi-time scale RNN model to process time-series data with two different frequencies [22]. A first RNN receives statistical features (mean and standard deviation) of an ECG-II signal every second and outputs a hidden representation every hour. A second RNN combines this output with other, more sparsely available data such as medications, demographics, and lab values. Hourly mortality predictions are made for the first 25 hours of available clinical data. Thorsen-Meyer et al. combines static features (demographics and diagnosis) with dynamic lab values and monitoring data [23]. For every one-hour interval, the maximum, minimum, and median of the continuous features are calculated. The LSTM-based model predicts the probability of the 90-day mortality hourly. The results are better for later predictions. Age and median HR were identified as the most important factors for mortality using SHAP. Likewise, Awad et al. aims to predict mortality early after admission using static features (age, gender, height, ICU type, initial weight) and 36 time-series from the first 48 hours after admission, that were aggregated hourly [24]. Using three different ML methods (RF, Bayesian Network, PART), it was shown that good predictions were possible after six hours after admission. On the contrary, Gupta, Liu, and Crick uses three dynamic features, BP, RR, and Glasgow Coma Scale [25]. The mean of two-hour intervals are processed by a two-state HMM. Again, the results get better closer to discharge.

#### **A.4.4 Other relevant studies**

Several similar studies were published to predict sepsis instead of mortality [12]. As with mortality prediction, most studies process static features or aggregated dynamic features and thus make non-continuous predictions. Scherpf et al. applies an RNN to vital signs to predict sepsis onset [13]. Vital sign signals are aggregated to 1-hour intervals and several hours of observation are given into the model as input. Moss et al. computes features from ECG waves and combines them with vital signs and other clinical information to predict a risk score every 15 minutes with LoR [14]. Ghosh et al. presents a model using coupled HMM to process different lengths of physiological signals [15]. 60- to 90-minute recordings of MAP, HR, and RR are used to predict future time series, based on which

the sepsis risk score is eventually determined. More studies applying RNNs and LSTMs for non-continuous predictions have been made (e.g. [16, 17]). However, in all presented studies features are calculated from the waveform beforehand instead of using the raw signal as input, so that the model can learn the features by itself.

Another field where automatic feature learning was studied is ECG-based feature extraction, disease detection, and sleep staging [79, 94]. They are usually based on CNNs or RNNs [94]. For example, He et al. classifies arrhythmias using a LSTM-based model, that automatically extracts features and classifies them [95]. Or Li et al., for instance, applies DNN, HMM, and SVM to detect apnea from ECG waveforms [96].

**Table A.2.** Comparison of relevant studies for mortality prediction using time-series data [78].

Author	Input Parameters	Processing of time-series data	Model	Results
<i>Non-temporal Models</i>				
Todd [18]	EMR data, Pulse wave, MAP wave	Five calculated features of 24 hours of waveform data.	LoR	<b>ACC:</b> EMR: 0.914 EMR+pulse: 0.916 EMR+MAP: 0.919
Sadeghi [19]	ECG wave	Twelve calculated features from one hour of waveform data.	i.a. DT, RF, SVM, LoR	AUC DT: 0.93
Kock [20]	EMR data, PPG wave	Median values of calculated features from a one-minute PPG waveform signals and its derivatives.	LoR, ANN	AUC LoR: 0.847 AUC ANN: 0.895
Morid [21]	EMR data	Average of a two-hour interval of 48 hours of time-series data.	k-NN	Precision: 0.65 F-Measure: 0.66
<i>Temporal Models</i>				
Ghanvatkar [22]	EMR data, ECG wave	RNN processes mean and standard deviation of ECG signal with a frequency 1 Hz. A second RNN combines the output with lower frequency data with a frequency 0.02 Hz	RNN	AUC: 0.7094
Thorsen-Meyer [23]	EMR data	Calculation of minimum, maximum, and median for one-hour intervals.	LSTM	<b>At Admission:</b> AUC DS1: 0.73 AUC DS2: 0.75 <b>After 72h:</b> AUC DS1: 0.85 AUC DS2: 0.82
Awad [24]	EMR data	One variable is aggregated for every one-hour interval.	RF, BN, PART	AUC RF: 0.82
Gupta [25]	EMR data	Aggregation of time-series data in two-hour intervals.	HMM	AUC: 0.87

*Note:* EMR: Electronic Medical Record; MAP: Mean Arterial Pressure; ECG: Electrocardiography; PPG: Photoplethysmography; LoR: Logistic Regression; DT: Decision Tree; RF: Random Forrest; SVM: Support Vector Machine; ANN: Artificial Neural Network; k-NN: k-nearest neighbours; RNN: Recurrent Neural Network; LSTM: Long short-term memory; BN: Bayesian Network; HMM: Hidden Markov Model; ACC: Accuracy; AUC: Area under the curve.

# Appendix B

## Model Summaries

This appendix chapter includes model summaries of one example of each architecture. The summaries were created in Python using `torchinfo` for PyTorch. The input for all models is given as a matrix, which looks for instance like this:

$$\text{input} = \begin{matrix} & t_0 & t_1 & t_2 & t_3 & \dots & t_n \\ \begin{matrix} ii \\ v2 \\ v5 \\ plet \\ sex \\ age \end{matrix} & \begin{pmatrix} -0.32 & -0.37 & -0.42 & -0.54 & \dots & -1, 81 \\ -0.14 & -0.21 & -0.25 & -0.29 & \dots & 0.12 \\ 0.25 & 0.13 & 0.04 & -0.04 & \dots & -0.15 \\ 0.10 & 0.13 & 0.14 & 0.14 & \dots & -0.00 \\ 1.15 & 1.15 & 1.15 & 1.15 & \dots & 1.15 \\ 0.62 & 0.62 & 0.62 & 0.62 & \dots & 0.62 \end{pmatrix} \end{matrix}$$

Note that the first four displayed features are dynamic (`ii`, `v2`, `v5`, `plet`) and the last two are static (`sex`, `age`). Furthermore, all features are normalized, which results in negative and unnatural values.

```

=====
Layer (type:depth-idx)      Output Shape      Param #
=====
Vanilla CNN                  [1, 2]            --
├─Sequential: 1-1            [1, 5, 10000]     --
│   └─Conv1d: 2-1             [1, 5, 70000]     200
│       └─BatchNorm1d: 2-2    [1, 5, 70000]     10
│           └─ReLU: 2-3        [1, 5, 70000]     --
│               └─MaxPool1d: 2-4 [1, 5, 10000]     --
├─Sequential: 1-2            [1, 5, 1428]      --
│   └─Conv1d: 2-5             [1, 5, 10000]     250
│       └─BatchNorm1d: 2-6    [1, 5, 10000]     10
│           └─ReLU: 2-7        [1, 5, 10000]     --
│               └─MaxPool1d: 2-8 [1, 5, 1428]      --
├─Sequential: 1-3            [1, 5, 204]        --
│   └─Conv1d: 2-9             [1, 5, 1428]      250
│       └─BatchNorm1d: 2-10   [1, 5, 1428]      10
│           └─ReLU: 2-11      [1, 5, 1428]      --
│               └─MaxPool1d: 2-12 [1, 5, 204]        --
├─Flatten: 1-4               [1, 1020]          --
├─Sequential: 1-5            [1, 2]             --
│   └─Linear: 2-13            [1, 2]             2,046
│       └─ReLU: 2-14          [1, 2]             --
└─Softmax: 1-6               [1, 2]             --
=====
Total params: 2,776
Trainable params: 2,776
Non-trainable params: 0
Total mult-adds (M): 16.86

Input size (MB): 1.68
Forward/backward pass size (MB): 6.51
Params size (MB): 0.01
Estimated Total Size (MB): 8.21
=====

```

**Figure B.1.** Model summary for a Vanilla CNN with  $d = 3$ ,  $m_i = 5$ ,  $l_i = 10$  and  $p_i = 7$  for 4 dynamic input features of length 70000 and 2 static features with a batch size of 1.

```

=====
Layer (type:depth-idx)      Output Shape      Param #
=====
CNN_Inception                [1, 2]            --
├─Conv1d: 1-1                [1, 3, 70000]     12
├─Conv1d: 1-2                [1, 5, 70000]     300
├─ReLU: 1-3                  [1, 5, 70000]     --
├─Conv1d: 1-4                [1, 5, 70000]     150
├─ReLU: 1-5                  [1, 5, 70000]     --
├─Conv1d: 1-6                [1, 5, 70000]     75
├─ReLU: 1-7                  [1, 5, 70000]     --
├─MaxPool1d: 1-8             [1, 4, 70000]     --
├─Conv1d: 1-9                [1, 5, 70000]     20
├─ReLU: 1-10                 [1, 5, 70000]     --
├─BatchNorm1d: 1-11          [1, 20, 70000]    40
├─ReLU: 1-12                 [1, 20, 70000]    --
├─AdaptiveAvgPool1d: 1-13    [1, 20, 1]         --
├─Flatten: 1-14              [1, 20]            --
├─Linear: 1-15               [1, 2]             46
└─Softmax: 1-16              [1, 2]             --
=====
Total params: 643
Trainable params: 643
Non-trainable params: 0
Total mult-adds (M): 38.99

Input size (MB): 1.68
Forward/backward pass size (MB): 24.08
Params size (MB): 0.00
Estimated Total Size (MB): 25.76
=====

```

**Figure B.2.** Model summary for a CNN Inception with  $m = 5$ ,  $c_b = 3$ ,  $l = [20, 10, 5]$  and  $d = 1$  for 4 dynamic input features of length 70000 and 2 static features with a batch size of 1.

```

=====
Layer (type:depth-idx)      Output Shape      Param #
=====
CNN_LSTM                    [1, 2]           --
├─Sequential: 1-1           [1, 10, 7000]   --
│   └─Conv1d: 2-1           [1, 10, 70000]  400
│       └─BatchNorm1d: 2-2  [1, 10, 70000]  20
│           └─ReLU: 2-3     [1, 10, 70000]  --
│               └─MaxPool1d: 2-4 [1, 10, 7000]  --
│                   └─Dropout: 2-5 [1, 10, 7000]  --
├─Sequential: 1-2           [1, 10, 700]    --
│   └─Conv1d: 2-6           [1, 10, 7000]   1,000
│       └─BatchNorm1d: 2-7  [1, 10, 7000]   20
│           └─ReLU: 2-8     [1, 10, 7000]   --
│               └─MaxPool1d: 2-9 [1, 10, 700]    --
│                   └─Dropout: 2-10 [1, 10, 700]    --
├─Sequential: 1-3           [1, 10, 70]     --
│   └─Conv1d: 2-11          [1, 10, 700]    1,000
│       └─BatchNorm1d: 2-12 [1, 10, 700]    20
│           └─ReLU: 2-13    [1, 10, 700]    --
│               └─MaxPool1d: 2-14 [1, 10, 70]     --
│                   └─Dropout: 2-15 [1, 10, 70]     --
├─LSTM: 1-4                 [1, 70, 5]      380
├─Sequential: 1-5           [1, 2]           --
│   └─Linear: 2-16          [1, 2]           12
│       └─ReLU: 2-17        [1, 2]           --
├─Flatten: 1-6              --
└─Softmax: 1-7              [1, 2]           --
=====
Total params: 2,852
Trainable params: 2,852
Non-trainable params: 0
Total mult-adds (M): 35.73
=====
Input size (MB): 1.68
Forward/backward pass size (MB): 12.43
Params size (MB): 0.01
Estimated Total Size (MB): 14.13
=====

```

**Figure B.3.** Model summary for a CNN-LSTM with  $d = 3$ ,  $m_i = 5$ ,  $l_i = 10$ ,  $p_i = 7$  and LSTM Features  $[5, 1]$  for 4 dynamic input features of length 70000 and 2 static features with a batch size of 1.

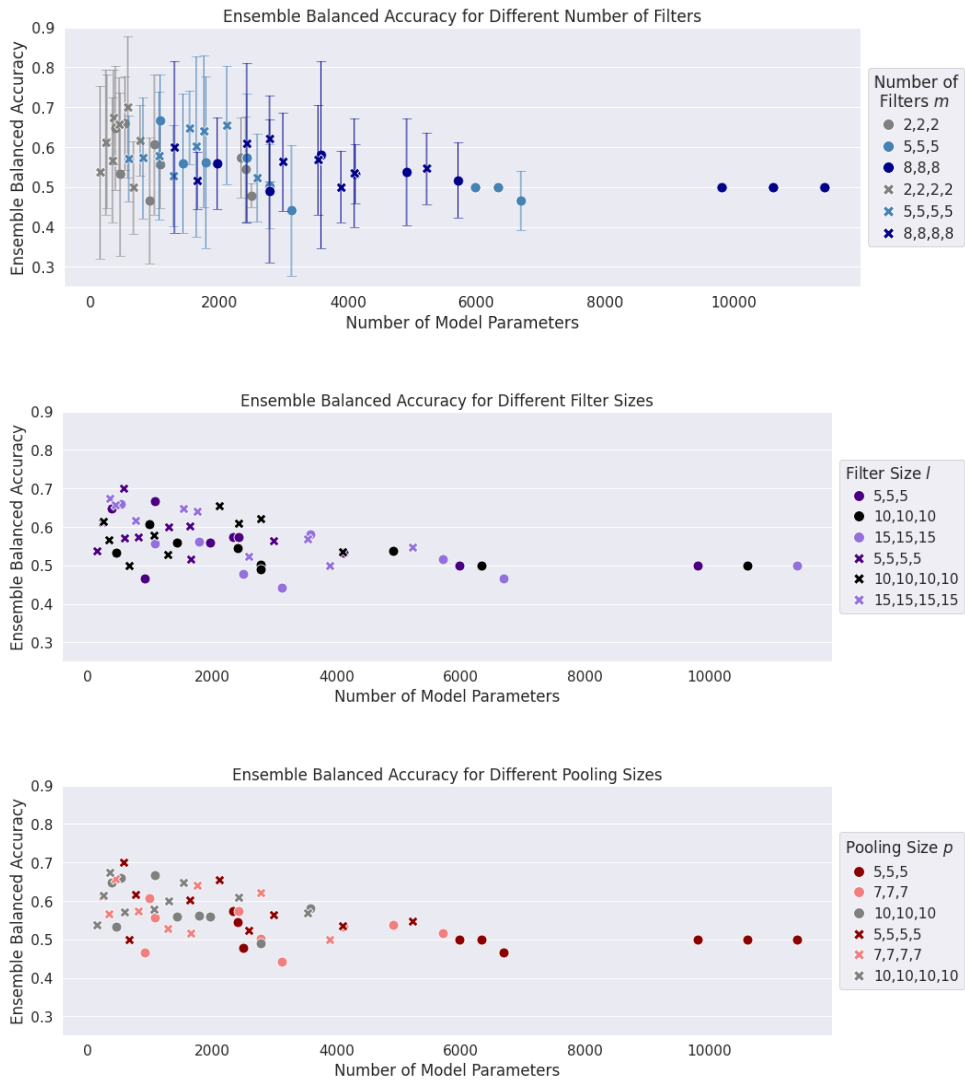
# Appendix C

## Additional Results

This appendix chapter includes additional results to the results of the experiments reported in section 4.2. A detailed analysis of the model hyperparameters and the results of all configurations for both experiments for each model architecture are given in section C.1, section C.2 and section C.3, respectively.

### C.1 Vanilla CNN

For model hyperparameter study, four different hyperparameters were varied: depth  $d$ , number of filters  $m$ , filter size  $l$  and pooling size  $p$ . Figure C.1 shows the ensemble balanced accuracies for each configuration depending on the model hyperparameters. The results of all configurations including standard deviations are listed in Table C.1. The result of all configurations of Vanilla CNN Experiment 2, variation of training hyperparameters, can be seen in Table C.2.



**Figure C.1.** Comparison of different model parameters from experiment 1 for Vanilla CNN using the balanced accuracies of the ensembles. For every configuration the mean and standard deviation was calculated over all runs and folds. For better visualization, the standard deviation is just shown in the first plot.

**Table C.1.** Vanilla CNN Experiment 1 results for ensembles. The average and standard deviation for every configuration is listed, the highest result is marked.

Model Parameters	Hyperparameters				Evaluation				
	d	m	l	p	Bal. Accuracy	Sensitivity	Specificity	Precision	F1-Score
150	4	2	5	10	0.537 (0.216)	0.267 (0.435)	0.807 (0.138)	0.28 (0.438)	0.214 (0.295)
238	4	2	5	7	0.612 (0.181)	0.4 (0.435)	0.823 (0.164)	0.36 (0.41)	0.314 (0.288)
250	4	2	10	10	0.615 (0.167)	0.35 (0.418)	0.88 (0.179)	0.34 (0.422)	0.27 (0.283)
338	4	2	10	7	0.567 (0.157)	0.367 (0.217)	0.767 (0.19)	0.38 (0.39)	0.34 (0.254)
350	4	2	15	10	0.675 (0.118)	0.4 (0.181)	0.95 (0.068)	<b>0.8 (0.274)</b>	0.527 (0.206)
378	3	2	5	10	0.648 (0.156)	0.4 (0.253)	0.895 (0.101)	0.6 (0.435)	0.454 (0.294)
438	4	2	15	7	0.658 (0.116)	0.5 (0.212)	0.817 (0.215)	0.607 (0.274)	0.51 (0.185)
458	3	2	10	10	0.532 (0.205)	0.25 (0.433)	0.815 (0.07)	0.233 (0.325)	0.227 (0.352)
538	3	2	15	10	0.659 (0.117)	0.567 (0.279)	0.752 (0.106)	0.431 (0.134)	0.477 (0.173)
570	4	2	5	5	<b>0.701 (0.177)</b>	<b>0.633 (0.375)</b>	0.769 (0.191)	0.525 (0.181)	<b>0.538 (0.237)</b>
591	4	5	5	10	0.571 (0.107)	0.217 (0.217)	0.925 (0.068)	0.433 (0.435)	0.281 (0.271)
670	4	2	10	5	0.499 (0.116)	0.233 (0.224)	0.764 (0.146)	0.217 (0.217)	0.224 (0.219)
770	4	2	15	5	0.616 (0.088)	0.467 (0.315)	0.766 (0.228)	0.483 (0.303)	0.384 (0.081)
811	4	5	5	7	0.572 (0.151)	0.4 (0.379)	0.745 (0.222)	0.275 (0.311)	0.317 (0.325)
914	3	2	5	7	0.465 (0.158)	0.1 (0.224)	0.83 (0.264)	0.1 (0.224)	0.1 (0.224)
994	3	2	10	7	0.606 (0.176)	0.533 (0.274)	0.679 (0.178)	0.36 (0.138)	0.427 (0.186)
1066	4	5	10	10	0.578 (0.16)	0.4 (0.365)	0.756 (0.145)	0.28 (0.284)	0.322 (0.305)
1074	3	2	15	7	0.558 (0.112)	0.35 (0.224)	0.765 (0.127)	0.25 (0.177)	0.283 (0.183)
1086	3	5	5	10	0.668 (0.115)	0.45 (0.162)	0.886 (0.079)	0.6 (0.253)	0.508 (0.194)
1286	4	5	10	7	0.528 (0.127)	0.167 (0.236)	0.89 (0.114)	0.267 (0.435)	0.18 (0.249)
1302	4	8	5	10	0.6 (0.216)	0.367 (0.28)	0.833 (0.189)	0.55 (0.447)	0.409 (0.313)
1436	3	5	10	10	0.559 (0.175)	0.367 (0.247)	0.751 (0.18)	0.4 (0.365)	0.373 (0.285)
1541	4	5	15	10	0.649 (0.093)	0.533 (0.139)	0.764 (0.152)	0.487 (0.307)	0.471 (0.16)
1641	4	5	5	5	0.601 (0.225)	0.433 (0.435)	0.769 (0.191)	0.317 (0.342)	0.358 (0.374)
1654	4	8	5	7	0.516 (0.072)	0.3 (0.274)	0.733 (0.142)	0.13 (0.12)	0.181 (0.166)
1761	4	5	15	7	0.64 (0.19)	0.55 (0.298)	0.729 (0.166)	0.433 (0.149)	0.459 (0.18)
1786	3	5	15	10	0.562 (0.216)	0.367 (0.375)	0.758 (0.192)	0.467 (0.361)	0.357 (0.25)
1974	3	8	5	10	0.559 (0.114)	0.367 (0.292)	0.752 (0.089)	0.327 (0.192)	0.324 (0.205)
2116	4	5	10	5	0.655 (0.148)	0.517 (0.291)	0.793 (0.13)	0.467 (0.315)	0.417 (0.148)
2338	3	2	5	5	0.573 (0.101)	0.167 (0.236)	0.98 (0.045)	0.3 (0.447)	0.2 (0.274)
2418	3	2	10	5	0.544 (0.133)	0.133 (0.298)	0.956 (0.061)	0.133 (0.298)	0.133 (0.298)
2422	4	8	10	10	0.61 (0.199)	0.4 (0.285)	0.82 (0.151)	0.55 (0.371)	0.45 (0.298)
2426	3	5	5	7	0.574 (0.161)	0.35 (0.418)	0.797 (0.212)	0.375 (0.415)	0.309 (0.306)
2498	3	2	15	5	0.479 (0.029)	0.0 (0.0)	0.958 (0.058)	0.0 (0.0)	0.0 (0.0)
2591	4	5	15	5	0.524 (0.11)	0.217 (0.217)	0.831 (0.184)	0.307 (0.413)	0.21 (0.201)
2774	3	8	10	10	0.49 (0.179)	0.383 (0.439)	0.597 (0.124)	0.181 (0.166)	0.226 (0.214)
2774	4	8	10	7	0.621 (0.108)	0.433 (0.149)	0.809 (0.168)	0.523 (0.327)	0.428 (0.161)
2776	3	5	10	7	0.501 (0.105)	0.183 (0.291)	0.819 (0.201)	0.167 (0.236)	0.156 (0.217)
2982	4	8	5	5	0.563 (0.123)	0.267 (0.253)	0.86 (0.167)	0.34 (0.422)	0.257 (0.251)
3126	3	5	15	7	0.441 (0.165)	0.067 (0.149)	0.816 (0.304)	0.067 (0.149)	0.067 (0.149)
3542	4	8	15	10	0.568 (0.138)	0.367 (0.342)	0.769 (0.091)	0.23 (0.228)	0.281 (0.271)
3574	3	8	15	10	0.58 (0.234)	0.3 (0.447)	0.861 (0.16)	0.32 (0.46)	0.283 (0.389)
3894	4	8	15	7	0.5 (0.09)	0.267 (0.181)	0.733 (0.168)	0.317 (0.207)	0.277 (0.166)
4102	4	8	10	5	0.536 (0.137)	0.217 (0.217)	0.856 (0.221)	0.307 (0.413)	0.244 (0.277)
4118	3	8	5	7	0.533 (0.075)	0.067 (0.149)	<b>1.0 (0.0)</b>	0.2 (0.447)	0.1 (0.224)
4918	3	8	10	7	0.538 (0.134)	0.3 (0.298)	0.776 (0.14)	0.247 (0.233)	0.27 (0.26)
5222	4	8	15	5	0.546 (0.091)	0.267 (0.253)	0.826 (0.094)	0.25 (0.25)	0.247 (0.233)
5718	3	8	15	7	0.517 (0.096)	0.167 (0.236)	0.868 (0.139)	0.15 (0.224)	0.147 (0.202)
5986	3	5	5	5	0.5 (0.0)	0.0 (0.0)	<b>1.0 (0.0)</b>	0.0 (0.0)	0.0 (0.0)
6336	3	5	10	5	0.5 (0.0)	0.0 (0.0)	<b>1.0 (0.0)</b>	0.0 (0.0)	0.0 (0.0)
6686	3	5	15	5	0.467 (0.075)	0.0 (0.0)	0.933 (0.149)	0.0 (0.0)	0.0 (0.0)
9814	3	8	5	5	0.5 (0.0)	0.0 (0.0)	<b>1.0 (0.0)</b>	0.0 (0.0)	0.0 (0.0)
10614	3	8	10	5	0.5 (0.0)	0.0 (0.0)	<b>1.0 (0.0)</b>	0.0 (0.0)	0.0 (0.0)
11414	3	8	15	5	0.5 (0.0)	0.0 (0.0)	<b>1.0 (0.0)</b>	0.0 (0.0)	0.0 (0.0)

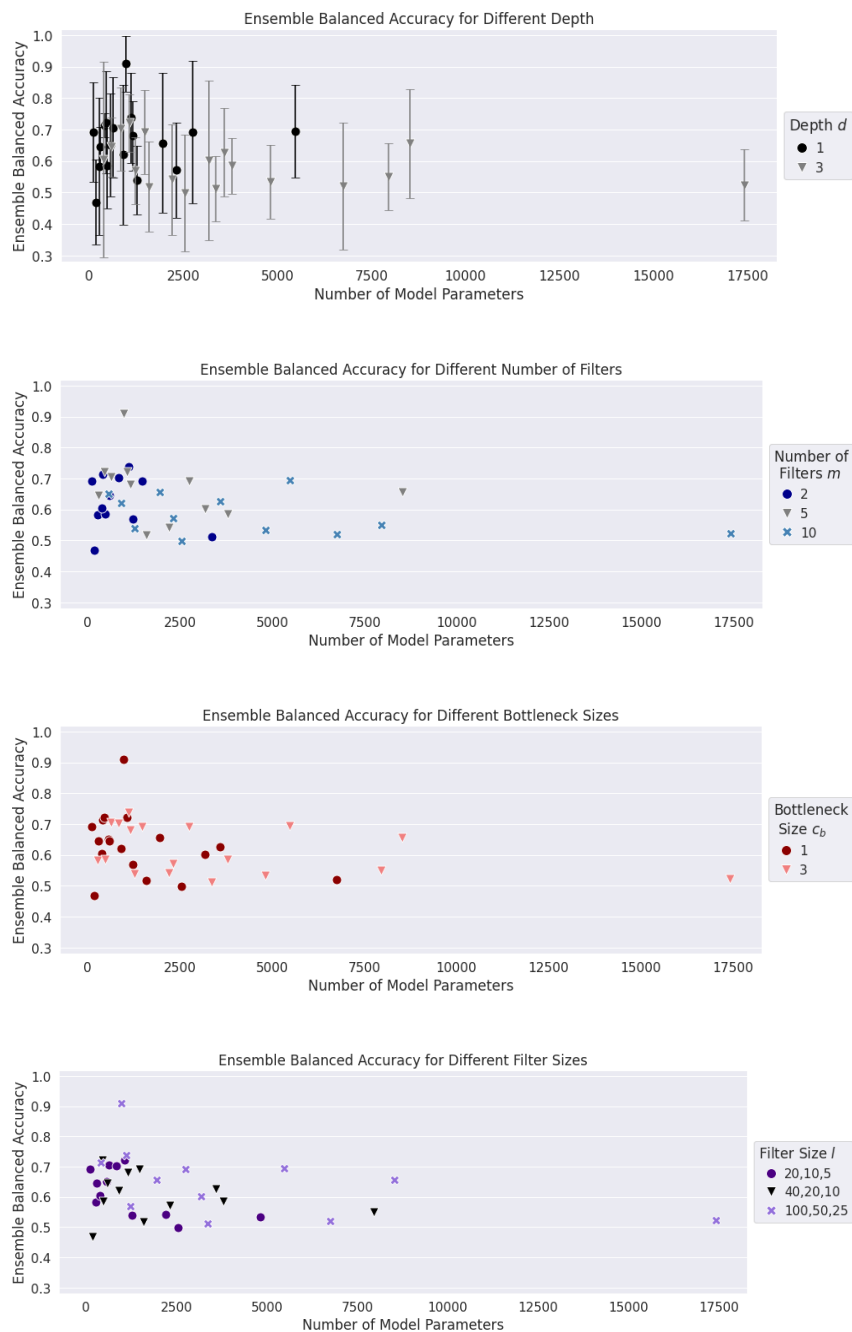
**Table C.2.** Vanilla CNN Experiment 2 results for ensembles. The average and standard deviation for every configuration is listed, the highest result is marked.

Model Parameter	Hyperparameters			Evaluation				
	LR	$\gamma$	$\lambda$	Bal. Accuracy	Sensitivity	Specificity	Precision	F1-Score
570	0.0001	0.70	0.00000	0.59 (0.098)	0.3 (0.326)	0.88 (0.154)	0.3 (0.274)	0.287 (0.278)
570	0.0001	0.70	0.00001	0.674 (0.198)	0.45 (0.371)	0.898 (0.1)	0.617 (0.439)	0.465 (0.324)
570	0.0001	0.70	0.00010	0.57 (0.174)	0.333 (0.358)	0.806 (0.162)	0.417 (0.449)	0.319 (0.321)
570	0.0001	0.70	0.00050	0.674 (0.154)	0.5 (0.289)	0.849 (0.124)	0.55 (0.274)	0.484 (0.173)
570	0.0001	0.70	0.00100	0.611 (0.234)	0.367 (0.415)	0.856 (0.087)	0.333 (0.312)	0.34 (0.344)
570	0.0001	0.85	0.00000	0.535 (0.134)	0.217 (0.217)	0.853 (0.161)	0.35 (0.418)	0.237 (0.229)
570	0.0001	0.85	0.00001	<b>0.801 (0.115)</b>	0.667 (0.312)	0.936 (0.098)	<b>0.853 (0.202)</b>	<b>0.677 (0.114)</b>
570	0.0001	0.85	0.00010	0.505 (0.08)	0.15 (0.335)	0.86 (0.219)	0.086 (0.192)	0.109 (0.244)
570	0.0001	0.85	0.00050	0.694 (0.052)	<b>0.767 (0.273)</b>	0.622 (0.205)	0.445 (0.135)	0.522 (0.12)
570	0.0001	0.85	0.00100	0.592 (0.155)	0.417 (0.373)	0.768 (0.147)	0.464 (0.375)	0.383 (0.269)
570	0.0001	1.00	0.00000	0.479 (0.144)	0.1 (0.224)	0.858 (0.151)	0.1 (0.224)	0.1 (0.224)
570	0.0001	1.00	0.00001	0.554 (0.106)	0.133 (0.183)	<b>0.975 (0.056)</b>	0.4 (0.548)	0.2 (0.274)
570	0.0001	1.00	0.00010	0.591 (0.191)	0.433 (0.365)	0.749 (0.146)	0.333 (0.204)	0.36 (0.239)
570	0.0001	1.00	0.00050	0.558 (0.142)	0.267 (0.253)	0.849 (0.147)	0.317 (0.41)	0.27 (0.283)
570	0.0001	1.00	0.00100	0.481 (0.153)	0.2 (0.209)	0.763 (0.201)	0.29 (0.413)	0.228 (0.272)
570	0.0003	0.70	0.00000	0.637 (0.057)	0.467 (0.075)	0.808 (0.119)	0.513 (0.096)	0.483 (0.065)
570	0.0003	0.70	0.00001	0.578 (0.087)	0.367 (0.217)	0.79 (0.149)	0.43 (0.37)	0.356 (0.21)
570	0.0003	0.70	0.00010	0.522 (0.178)	0.317 (0.239)	0.728 (0.125)	0.317 (0.239)	0.314 (0.237)
570	0.0003	0.70	0.00050	0.533 (0.179)	0.25 (0.306)	0.816 (0.171)	0.42 (0.427)	0.28 (0.284)
570	0.0003	0.70	0.00100	0.624 (0.077)	0.4 (0.253)	0.848 (0.159)	0.463 (0.385)	0.381 (0.23)
570	0.0003	0.85	0.00000	0.664 (0.185)	0.567 (0.365)	0.762 (0.274)	0.42 (0.388)	0.449 (0.33)
570	0.0003	0.85	0.00001	0.678 (0.143)	0.483 (0.291)	0.873 (0.164)	0.523 (0.412)	0.474 (0.32)
570	0.0003	0.85	0.00010	0.674 (0.106)	0.433 (0.253)	0.916 (0.116)	0.567 (0.435)	0.461 (0.28)
570	0.0003	0.85	0.00050	0.658 (0.252)	0.533 (0.447)	0.782 (0.193)	0.393 (0.314)	0.436 (0.352)
570	0.0003	0.85	0.00100	0.709 (0.22)	0.65 (0.487)	0.768 (0.128)	0.363 (0.267)	0.443 (0.321)
570	0.0003	1.00	0.00000	0.588 (0.109)	0.367 (0.247)	0.809 (0.149)	0.307 (0.243)	0.324 (0.237)
570	0.0003	1.00	0.00001	0.504 (0.119)	0.267 (0.181)	0.741 (0.101)	0.283 (0.183)	0.257 (0.145)
570	0.0003	1.00	0.00010	0.624 (0.196)	0.417 (0.373)	0.831 (0.207)	0.493 (0.396)	0.409 (0.297)
570	0.0003	1.00	0.00050	0.676 (0.2)	0.5 (0.395)	0.853 (0.15)	0.54 (0.42)	0.459 (0.349)
570	0.0003	1.00	0.00100	0.574 (0.103)	0.317 (0.41)	0.831 (0.207)	0.224 (0.219)	0.222 (0.208)
570	0.0005	0.70	0.00000	0.599 (0.164)	0.45 (0.389)	0.748 (0.268)	0.411 (0.376)	0.37 (0.287)
570	0.0005	0.70	0.00001	0.538 (0.084)	0.167 (0.236)	0.908 (0.146)	0.183 (0.291)	0.171 (0.256)
570	0.0005	0.70	0.00010	0.596 (0.126)	0.417 (0.118)	0.775 (0.185)	0.573 (0.292)	0.459 (0.136)
570	0.0005	0.70	0.00050	0.701 (0.09)	0.583 (0.186)	0.819 (0.097)	0.547 (0.117)	0.548 (0.139)
570	0.0005	0.70	0.00100	0.552 (0.213)	0.317 (0.41)	0.787 (0.252)	0.367 (0.415)	0.26 (0.241)
570	0.0005	0.85	0.00000	0.705 (0.212)	0.583 (0.373)	0.826 (0.14)	0.48 (0.398)	0.517 (0.384)
570	0.0005	0.85	0.00001	0.675 (0.192)	0.583 (0.289)	0.766 (0.185)	0.573 (0.3)	0.532 (0.211)
570	0.0005	0.85	0.00010	0.655 (0.167)	0.6 (0.285)	0.71 (0.088)	0.367 (0.126)	0.45 (0.177)
570	0.0005	0.85	0.00050	0.565 (0.091)	0.317 (0.207)	0.813 (0.142)	0.317 (0.207)	0.31 (0.195)
570	0.0005	0.85	0.00100	0.589 (0.17)	0.517 (0.171)	0.661 (0.216)	0.473 (0.313)	0.473 (0.21)
570	0.0005	1.00	0.00000	0.655 (0.181)	0.5 (0.306)	0.81 (0.251)	0.662 (0.362)	0.464 (0.213)
570	0.0005	1.00	0.00001	0.483 (0.257)	0.333 (0.408)	0.632 (0.222)	0.217 (0.217)	0.257 (0.277)
570	0.0005	1.00	0.00010	0.503 (0.095)	0.25 (0.25)	0.755 (0.144)	0.197 (0.187)	0.213 (0.203)
570	0.0005	1.00	0.00050	0.546 (0.078)	0.183 (0.171)	0.908 (0.094)	0.367 (0.415)	0.227 (0.209)
570	0.0005	1.00	0.00100	0.748 (0.173)	0.7 (0.298)	0.796 (0.292)	0.611 (0.369)	0.599 (0.29)

## C.2 CNN Inception

For model hyperparameter study, four different hyperparameters were varied: depth  $d$ , number of filters  $m$ , bottleneck size  $c_b$  and filter size  $l$ . Figure C.2 shows the ensemble balanced accuracies for each configuration depending on the model hyperparameters. The results of all configurations including standard deviations are listed in Table C.3.

The result of all configurations of CNN Inception Experiment 2, variation of training hyperparameters, can be seen in Table C.4.



**Figure C.2.** Comparison of different model parameters from experiment 1 for CNN Inception using the balanced accuracies of the ensembles. For every configuration the mean and standard deviation was calculated over all runs and folds. For better visualization, the standard deviation is just shown in the first plot.

**Table C.3.** CNN Inception Experiment 1 results for ensembles. The average and standard deviation for every configuration is listed, the highest result is marked.

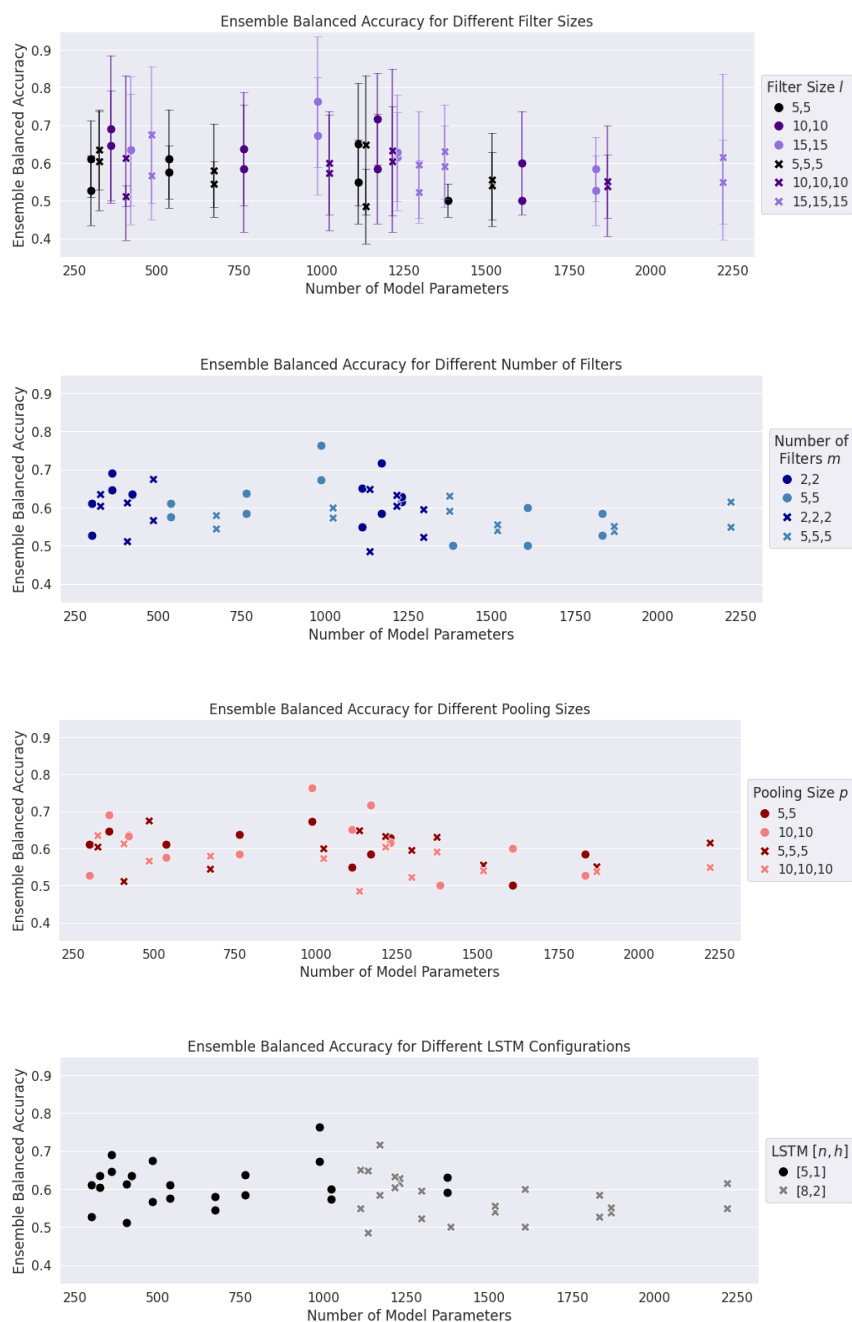
Model Param.	Hyperparameters				Evaluation				
	$c_b$	$m$	$l$	$d$	Bal. Accuracy	Sensitivity	Specificity	Precision	F1-Score
120	1	2	20,10,5	1	0.692 (0.158)	0.683 (0.325)	0.701 (0.115)	0.367 (0.075)	0.459 (0.14)
190	1	2	40,20,10	1	0.469 (0.135)	0.3 (0.298)	0.638 (0.126)	0.153 (0.166)	0.202 (0.211)
268	3	2	20,10,5	1	0.582 (0.218)	0.4 (0.418)	0.765 (0.102)	0.273 (0.3)	0.305 (0.312)
285	1	5	20,10,5	1	0.645 (0.062)	0.55 (0.112)	0.74 (0.042)	0.387 (0.157)	0.447 (0.141)
388	1	2	20,10,5	3	0.604 (0.31)	0.5 (0.5)	0.707 (0.135)	0.33 (0.327)	0.394 (0.388)
400	1	2	100,50,25	1	0.713 (0.037)	0.767 (0.224)	0.66 (0.173)	0.399 (0.098)	0.507 (0.079)
460	1	5	40,20,10	1	0.722 (0.162)	0.733 (0.279)	0.711 (0.127)	0.466 (0.183)	0.563 (0.205)
478	3	2	40,20,10	1	0.586 (0.137)	0.567 (0.253)	0.605 (0.217)	0.372 (0.132)	0.432 (0.122)
560	1	10	20,10,5	1	0.652 (0.163)	0.483 (0.384)	0.821 (0.089)	0.413 (0.25)	0.414 (0.262)
598	1	2	40,20,10	1	0.645 (0.094)	0.467 (0.075)	0.823 (0.13)	0.517 (0.291)	0.47 (0.141)
643	3	5	20,10,5	1	0.707 (0.159)	0.6 (0.253)	0.813 (0.083)	0.45 (0.201)	0.507 (0.214)
848	3	2	20,10,5	3	0.702 (0.133)	0.583 (0.276)	0.82 (0.179)	0.587 (0.384)	0.491 (0.2)
910	1	10	40,20,10	1	0.62 (0.223)	0.55 (0.447)	0.69 (0.188)	0.428 (0.387)	0.39 (0.282)
985	1	5	100,50,25	1	<b>0.909 (0.089)</b>	<b>0.95 (0.112)</b>	0.868 (0.066)	0.67 (0.053)	<b>0.785 (0.071)</b>
1075	1	5	20,10,5	3	0.721 (0.091)	0.55 (0.298)	<b>0.893 (0.123)</b>	<b>0.747 (0.256)</b>	0.542 (0.099)
1108	3	2	100,50,25	1	0.738 (0.143)	0.767 (0.325)	0.709 (0.184)	0.469 (0.15)	0.56 (0.192)
1168	3	5	40,20,10	1	0.68 (0.111)	0.65 (0.253)	0.71 (0.1)	0.422 (0.137)	0.505 (0.157)
1228	1	2	100,50,25	3	0.569 (0.105)	0.35 (0.253)	0.788 (0.168)	0.333 (0.236)	0.311 (0.21)
1268	3	10	20,10,5	1	0.54 (0.108)	0.217 (0.217)	0.863 (0.095)	0.267 (0.253)	0.227 (0.209)
1478	3	2	40,20,10	3	0.692 (0.133)	0.467 (0.38)	0.918 (0.131)	0.613 (0.425)	0.448 (0.259)
1600	1	5	40,20,10	3	0.518 (0.143)	0.217 (0.217)	0.82 (0.175)	0.367 (0.415)	0.257 (0.251)
1960	1	10	100,50,25	1	0.658 (0.223)	0.467 (0.361)	0.849 (0.124)	0.517 (0.384)	0.465 (0.324)
2213	3	5	20,10,5	3	0.541 (0.175)	0.283 (0.298)	0.799 (0.095)	0.267 (0.279)	0.27 (0.283)
2318	3	10	40,20,10	1	0.571 (0.151)	0.433 (0.091)	0.709 (0.258)	0.459 (0.332)	0.409 (0.171)
2540	1	10	20,10,5	3	0.498 (0.184)	0.217 (0.217)	0.78 (0.177)	0.367 (0.415)	0.247 (0.233)
2743	3	5	100,50,25	1	0.693 (0.226)	0.6 (0.418)	0.786 (0.074)	0.42 (0.239)	0.483 (0.291)
3175	1	5	100,50,25	3	0.603 (0.253)	0.383 (0.439)	0.822 (0.202)	0.467 (0.506)	0.369 (0.411)
3368	3	2	100,50,25	3	0.512 (0.104)	0.217 (0.217)	0.808 (0.068)	0.267 (0.253)	0.237 (0.229)
3590	1	10	40,20,10	3	0.628 (0.14)	0.467 (0.361)	0.788 (0.113)	0.381 (0.23)	0.412 (0.27)
3788	3	5	40,20,10	3	0.584 (0.087)	0.4 (0.253)	0.769 (0.165)	0.38 (0.247)	0.372 (0.218)
4808	3	10	20,10,5	3	0.534 (0.116)	0.283 (0.183)	0.784 (0.13)	0.383 (0.371)	0.28 (0.159)
5468	3	10	100,50,25	1	0.694 (0.147)	0.733 (0.279)	0.656 (0.127)	0.394 (0.15)	0.499 (0.173)
6740	1	10	100,50,25	3	0.52 (0.201)	0.3 (0.447)	0.74 (0.095)	0.15 (0.224)	0.2 (0.298)
7958	3	10	40,20,10	3	0.55 (0.106)	0.267 (0.253)	0.833 (0.117)	0.313 (0.301)	0.283 (0.266)
8513	3	5	100,50,25	3	0.656 (0.173)	0.467 (0.361)	0.845 (0.12)	0.513 (0.366)	0.429 (0.242)
17408	3	10	100,50,25	3	0.524 (0.112)	0.183 (0.171)	0.866 (0.096)	0.367 (0.415)	0.237 (0.229)

**Table C.4.** CNN Inception Experiment 2 results for ensembles. The average and standard deviation for every configuration is listed, the highest result is marked.

Model Parameters	Hyperparameters			Evaluation				
	LR	$\gamma$	$\lambda$	Bal. Accuracy	Sensitivity	Specificity	Precision	F1-Score
985	0.0001	0.70	0.0000	0.608 (0.113)	0.533 (0.075)	0.683 (0.216)	0.41 (0.183)	0.444 (0.116)
985	0.0001	0.70	0.0001	0.495 (0.097)	0.333 (0.204)	0.657 (0.094)	0.233 (0.181)	0.264 (0.178)
985	0.0001	0.70	0.0005	0.638 (0.12)	0.467 (0.274)	0.809 (0.14)	0.43 (0.37)	0.414 (0.262)
985	0.0001	0.70	0.0010	0.736 (0.17)	0.633 (0.28)	<b>0.838 (0.107)</b>	0.6 (0.253)	0.595 (0.214)
985	0.0001	0.85	0.0000	0.536 (0.12)	0.267 (0.181)	0.805 (0.13)	0.417 (0.373)	0.29 (0.182)
985	0.0001	0.85	0.0001	0.612 (0.143)	0.433 (0.253)	0.791 (0.123)	0.4 (0.379)	0.381 (0.259)
985	0.0001	0.85	0.0005	0.591 (0.108)	0.483 (0.171)	0.699 (0.122)	0.37 (0.134)	0.417 (0.148)
985	0.0001	0.85	0.0010	0.741 (0.147)	0.7 (0.298)	0.782 (0.01)	0.473 (0.134)	0.561 (0.193)
985	0.0001	1.00	0.0000	0.57 (0.111)	0.35 (0.224)	0.79 (0.113)	0.34 (0.259)	0.318 (0.208)
985	0.0001	1.00	0.0001	0.623 (0.189)	0.533 (0.38)	0.713 (0.204)	0.34 (0.268)	0.407 (0.306)
985	0.0001	1.00	0.0005	0.741 (0.164)	0.7 (0.209)	0.783 (0.171)	0.557 (0.322)	0.6 (0.26)
985	0.0001	1.00	0.0010	0.721 (0.201)	0.733 (0.365)	0.709 (0.192)	0.483 (0.303)	0.557 (0.289)
985	0.0005	0.70	0.0000	<b>0.861 (0.123)</b>	<b>0.95 (0.112)</b>	0.773 (0.171)	<b>0.65 (0.253)</b>	<b>0.751 (0.201)</b>
985	0.0005	0.70	0.0001	0.786 (0.13)	0.8 (0.209)	0.773 (0.097)	0.537 (0.153)	0.638 (0.164)
985	0.0005	0.70	0.0005	0.664 (0.13)	0.667 (0.204)	0.662 (0.057)	0.413 (0.166)	0.506 (0.186)
985	0.0005	0.70	0.0010	0.55 (0.09)	0.333 (0.204)	0.767 (0.119)	0.33 (0.242)	0.327 (0.213)
985	0.0005	0.85	0.0000	0.729 (0.11)	0.833 (0.156)	0.626 (0.133)	0.471 (0.124)	0.592 (0.119)
985	0.0005	0.85	0.0001	0.778 (0.168)	0.767 (0.325)	0.789 (0.083)	0.467 (0.139)	0.573 (0.198)
985	0.0005	0.85	0.0005	0.734 (0.236)	0.75 (0.433)	0.718 (0.118)	0.437 (0.287)	0.541 (0.332)
985	0.0005	0.85	0.0010	0.697 (0.167)	0.6 (0.435)	0.793 (0.136)	0.38 (0.247)	0.428 (0.258)
985	0.0005	1.00	0.0000	0.667 (0.104)	0.567 (0.253)	0.768 (0.079)	0.397 (0.108)	0.441 (0.094)
985	0.0005	1.00	0.0001	0.776 (0.093)	0.767 (0.224)	0.786 (0.063)	0.513 (0.096)	0.59 (0.043)
985	0.0005	1.00	0.0005	0.792 (0.176)	0.85 (0.224)	0.734 (0.164)	0.56 (0.288)	0.66 (0.259)
985	0.0005	1.00	0.0010	0.668 (0.192)	0.667 (0.312)	0.669 (0.167)	0.354 (0.229)	0.447 (0.243)
985	0.0010	0.70	0.0000	0.626 (0.128)	0.567 (0.091)	0.686 (0.204)	0.415 (0.249)	0.449 (0.189)
985	0.0010	0.70	0.0001	0.569 (0.208)	0.517 (0.384)	0.621 (0.224)	0.357 (0.287)	0.397 (0.284)
985	0.0010	0.70	0.0005	0.646 (0.083)	0.583 (0.118)	0.708 (0.156)	0.426 (0.175)	0.48 (0.145)
985	0.0010	0.70	0.0010	0.729 (0.195)	0.7 (0.298)	0.759 (0.124)	0.473 (0.169)	0.55 (0.183)
985	0.0010	0.85	0.0000	0.826 (0.138)	0.833 (0.236)	0.819 (0.167)	0.669 (0.225)	0.72 (0.197)
985	0.0010	0.85	0.0001	0.636 (0.074)	0.5 (0.212)	0.771 (0.091)	0.433 (0.091)	0.448 (0.129)
985	0.0010	0.85	0.0005	0.688 (0.154)	0.617 (0.261)	0.758 (0.082)	0.45 (0.24)	0.514 (0.243)
985	0.0010	0.85	0.0010	0.606 (0.237)	0.383 (0.439)	0.828 (0.087)	0.333 (0.312)	0.341 (0.352)
985	0.0010	1.00	0.0000	0.753 (0.112)	0.8 (0.274)	0.706 (0.186)	0.45 (0.132)	0.544 (0.13)
985	0.0010	1.00	0.0001	0.643 (0.167)	0.6 (0.181)	0.686 (0.211)	0.49 (0.223)	0.533 (0.2)
985	0.0010	1.00	0.0005	0.632 (0.295)	0.533 (0.506)	0.731 (0.177)	0.347 (0.409)	0.403 (0.422)
985	0.0010	1.00	0.0010	0.695 (0.104)	0.6 (0.224)	0.79 (0.124)	0.497 (0.296)	0.494 (0.133)

### C.3 CNN-LSTM

For model hyperparameter study, four different hyperparameters were varied: filter size  $l$ , number of filters  $m$ , pooling size  $p$  and LSTM Features  $[n, h]$ . Figure C.3 shows the ensemble balanced accuracies for each configuration depending on the model hyperparameters. The results of all configurations including standard deviations are listed in ???. The result of all configurations of CNN Inception Experiment 2, variation of training hyperparameters, can be seen in Table C.6.



**Figure C.3.** Comparison of different model parameters from experiment 1 for CNN-LSTM using the balanced accuracies of the ensembles. For every configuration the mean and standard deviation was calculated over all runs and folds. For better visualization, the standard deviation is just shown in the first plot.

**Table C.5.** CNN-LSTM Experiment 1 results for ensembles. The average and standard deviation for every configuration is listed, the highest result is marked.

Model Param.	Hyperparameters					Evaluation				
	d	m <sub>i</sub>	l <sub>i</sub>	p <sub>i</sub>	[n, h]	Ba <sub>i</sub> . Accuracy	Sensitivity	Specificity	Precision	F1-Score
300	2	2	5	10	5,1	0.528 (0.092)	0.1 (0.224)	0.955 (0.062)	0.133 (0.298)	0.114 (0.256)
300	2	2	5	5	5,1	0.611 (0.102)	0.367 (0.415)	0.855 (0.26)	0.383 (0.439)	0.294 (0.275)
324	3	2	5	10	5,1	0.636 (0.107)	0.4 (0.253)	0.871 (0.118)	0.433 (0.365)	0.394 (0.257)
324	3	2	5	5	5,1	0.605 (0.131)	0.4 (0.418)	0.81 (0.188)	0.261 (0.254)	0.314 (0.313)
360	2	2	10	10	5,1	0.69 (0.195)	0.6 (0.548)	0.78 (0.268)	0.25 (0.276)	0.34 (0.344)
360	2	2	10	5	5,1	0.646 (0.146)	0.483 (0.393)	0.808 (0.142)	0.4 (0.253)	0.427 (0.3)
404	3	2	10	10	5,1	0.612 (0.218)	0.3 (0.447)	0.925 (0.168)	0.28 (0.438)	0.289 (0.442)
404	3	2	10	5	5,1	0.512 (0.028)	0.1 (0.224)	0.925 (0.168)	0.08 (0.179)	0.089 (0.199)
420	2	2	15	10	5,1	0.634 (0.196)	0.35 (0.418)	0.918 (0.084)	0.4 (0.435)	0.32 (0.335)
420	2	2	15	5	5,1	0.635 (0.147)	0.35 (0.418)	0.92 (0.13)	0.38 (0.415)	0.294 (0.275)
484	3	2	15	10	5,1	0.568 (0.117)	0.3 (0.447)	0.835 (0.25)	0.183 (0.291)	0.194 (0.273)
484	3	2	15	5	5,1	0.676 (0.181)	0.417 (0.449)	0.935 (0.093)	0.45 (0.447)	0.383 (0.361)
537	2	5	5	10	5,1	0.575 (0.071)	0.3 (0.274)	0.85 (0.141)	0.217 (0.217)	0.247 (0.233)
537	2	5	5	5	5,1	0.611 (0.13)	0.4 (0.279)	0.822 (0.127)	0.35 (0.253)	0.371 (0.262)
672	3	5	5	10	5,1	0.58 (0.123)	0.4 (0.224)	0.76 (0.199)	0.413 (0.363)	0.382 (0.241)
672	3	5	5	5	5,1	0.544 (0.061)	0.183 (0.171)	0.906 (0.103)	0.267 (0.253)	0.217 (0.204)
762	2	5	10	10	5,1	0.586 (0.168)	0.3 (0.447)	0.871 (0.197)	0.286 (0.44)	0.253 (0.348)
762	2	5	10	5	5,1	0.637 (0.15)	0.333 (0.312)	0.94 (0.134)	0.45 (0.512)	0.36 (0.37)
987	2	5	15	10	5,1	<b>0.762 (0.173)</b>	<b>0.65 (0.379)</b>	0.875 (0.153)	<b>0.764 (0.234)</b>	<b>0.642 (0.274)</b>
987	2	5	15	5	5,1	0.672 (0.155)	0.483 (0.393)	0.861 (0.127)	0.507 (0.37)	0.45 (0.298)
1022	3	5	10	10	5,1	0.574 (0.154)	0.233 (0.325)	0.916 (0.048)	0.233 (0.325)	0.233 (0.325)
1022	3	5	10	5	5,1	0.6 (0.137)	0.35 (0.335)	0.85 (0.224)	0.353 (0.437)	0.317 (0.335)
1110	2	2	5	10	8,2	0.65 (0.163)	0.4 (0.285)	0.9 (0.137)	0.567 (0.435)	0.462 (0.333)
1110	2	2	5	5	8,2	0.55 (0.112)	0.1 (0.224)	<b>1.0 (0.0)</b>	0.2 (0.447)	0.133 (0.298)
1134	3	2	5	10	8,2	0.485 (0.1)	0.167 (0.236)	0.804 (0.213)	0.1 (0.149)	0.124 (0.182)
1134	3	2	5	5	8,2	0.648 (0.184)	0.367 (0.415)	0.93 (0.11)	0.433 (0.435)	0.36 (0.351)
1170	2	2	10	10	8,2	0.717 (0.122)	0.5 (0.289)	0.933 (0.061)	0.6 (0.365)	0.533 (0.298)
1170	2	2	10	5	8,2	0.584 (0.145)	0.233 (0.325)	0.936 (0.059)	0.233 (0.325)	0.233 (0.325)
1214	3	2	10	10	8,2	0.606 (0.145)	0.233 (0.325)	0.978 (0.05)	0.333 (0.471)	0.267 (0.365)
1214	3	2	10	5	8,2	0.633 (0.217)	0.267 (0.435)	<b>1.0 (0.0)</b>	0.4 (0.548)	0.3 (0.447)
1230	2	2	15	10	8,2	0.617 (0.119)	0.3 (0.298)	0.933 (0.099)	0.4 (0.418)	0.328 (0.314)
1230	2	2	15	5	8,2	0.628 (0.154)	0.3 (0.298)	0.956 (0.099)	0.467 (0.506)	0.36 (0.37)
1294	3	2	15	10	8,2	0.523 (0.083)	0.133 (0.298)	0.913 (0.145)	0.08 (0.179)	0.1 (0.224)
1294	3	2	15	5	8,2	0.596 (0.141)	0.283 (0.389)	0.908 (0.146)	0.23 (0.338)	0.25 (0.354)
1372	3	5	15	10	5,1	0.591 (0.108)	0.333 (0.312)	0.849 (0.242)	0.324 (0.409)	0.293 (0.289)
1372	3	5	15	5	5,1	0.63 (0.125)	0.35 (0.335)	0.91 (0.124)	0.387 (0.425)	0.347 (0.335)
1383	2	5	5	10	8,2	0.5 (0.0)	0.05 (0.112)	0.95 (0.112)	0.067 (0.149)	0.057 (0.128)
1383	2	5	5	5	8,2	0.5 (0.044)	0.1 (0.137)	0.9 (0.105)	0.167 (0.236)	0.124 (0.17)
1518	3	5	5	10	8,2	0.54 (0.089)	0.15 (0.224)	0.93 (0.11)	0.167 (0.236)	0.157 (0.228)
1518	3	5	5	5	8,2	0.556 (0.124)	0.183 (0.291)	0.928 (0.11)	0.2 (0.298)	0.19 (0.294)
1608	2	5	10	10	8,2	0.6 (0.137)	0.2 (0.274)	<b>1.0 (0.0)</b>	0.4 (0.548)	0.267 (0.365)
1608	2	5	10	5	8,2	0.5 (0.0)	0.0 (0.0)	<b>1.0 (0.0)</b>	0.0 (0.0)	0.0 (0.0)
1833	2	5	15	10	8,2	0.528 (0.092)	0.1 (0.224)	0.955 (0.062)	0.133 (0.298)	0.114 (0.256)
1833	2	5	15	5	8,2	0.584 (0.085)	0.233 (0.224)	0.936 (0.059)	0.3 (0.274)	0.26 (0.241)
1868	3	5	10	10	8,2	0.538 (0.084)	0.1 (0.224)	0.975 (0.056)	0.133 (0.298)	0.114 (0.256)
1868	3	5	10	5	8,2	0.552 (0.147)	0.15 (0.335)	0.955 (0.062)	0.15 (0.335)	0.15 (0.335)
2218	3	5	15	10	8,2	0.55 (0.112)	0.1 (0.224)	<b>1.0 (0.0)</b>	0.2 (0.447)	0.133 (0.298)
2218	3	5	15	5	8,2	0.616 (0.22)	0.367 (0.415)	0.866 (0.102)	0.3 (0.298)	0.327 (0.342)

**Table C.6.** CNN-LSTM Experiment 2 results for ensembles. The average and standard deviation for every configuration is listed, the highest result is marked.

Model Parameters	Hyperparameters			Evaluation				
	LR	$\gamma$	$\lambda$	Bal. Accuracy	Sensitivity	Specificity	Precision	F1-Score
987	0.0001	0.85	0.0000	0.527 (0.119)	0.2 (0.274)	0.853 (0.092)	0.133 (0.183)	0.16 (0.219)
987	0.0001	0.85	0.0001	0.635 (0.209)	0.383 (0.439)	0.886 (0.193)	0.367 (0.415)	0.356 (0.411)
987	0.0001	0.85	0.0005	0.638 (0.112)	0.35 (0.224)	0.925 (0.112)	0.6 (0.418)	0.433 (0.279)
987	0.0001	0.85	0.0010	0.62 (0.179)	0.35 (0.418)	0.89 (0.114)	0.267 (0.253)	0.29 (0.298)
987	0.0001	1.00	0.0000	0.552 (0.053)	0.267 (0.253)	0.838 (0.181)	0.19 (0.207)	0.204 (0.19)
987	0.0001	1.00	0.0001	0.592 (0.142)	0.3 (0.326)	0.885 (0.169)	0.5 (0.5)	0.333 (0.32)
987	0.0001	1.00	0.0005	0.705 (0.23)	<b>0.55 (0.447)</b>	0.861 (0.062)	0.483 (0.291)	0.498 (0.352)
987	0.0001	1.00	0.0010	0.529 (0.126)	0.1 (0.224)	0.958 (0.058)	0.2 (0.447)	0.133 (0.298)
987	0.0002	0.85	0.0000	0.489 (0.061)	0.117 (0.162)	0.861 (0.127)	0.133 (0.183)	0.124 (0.17)
987	0.0002	0.85	0.0001	0.717 (0.145)	0.5 (0.373)	0.933 (0.099)	0.653 (0.409)	0.517 (0.303)
987	0.0002	0.85	0.0005	0.48 (0.045)	0.0 (0.0)	0.96 (0.089)	0.0 (0.0)	0.0 (0.0)
987	0.0002	0.85	0.0010	0.625 (0.147)	0.35 (0.224)	0.9 (0.163)	0.583 (0.449)	0.431 (0.296)
987	0.0002	1.00	0.0000	0.601 (0.117)	0.317 (0.325)	0.886 (0.119)	0.287 (0.278)	0.3 (0.298)
987	0.0002	1.00	0.0001	0.625 (0.072)	0.3 (0.183)	0.95 (0.112)	0.7 (0.447)	0.4 (0.224)
987	0.0002	1.00	0.0005	0.525 (0.034)	0.1 (0.137)	0.95 (0.068)	0.2 (0.274)	0.133 (0.183)
987	0.0002	1.00	0.0010	0.56 (0.059)	0.167 (0.156)	0.953 (0.065)	0.4 (0.418)	0.227 (0.209)
987	0.0005	0.85	0.0000	0.589 (0.149)	0.2 (0.274)	0.978 (0.05)	0.4 (0.548)	0.267 (0.365)
987	0.0005	0.85	0.0001	0.557 (0.165)	0.3 (0.447)	0.815 (0.143)	0.214 (0.295)	0.245 (0.346)
987	0.0005	0.85	0.0005	0.668 (0.178)	0.4 (0.365)	0.936 (0.059)	0.467 (0.447)	0.427 (0.393)
987	0.0005	0.85	0.0010	0.701 (0.179)	0.467 (0.361)	0.936 (0.098)	0.6 (0.435)	0.493 (0.325)
987	0.0005	1.00	0.0000	0.593 (0.158)	0.233 (0.325)	0.953 (0.065)	0.333 (0.471)	0.267 (0.365)
987	0.0005	1.00	0.0001	0.525 (0.056)	0.1 (0.224)	0.95 (0.112)	0.1 (0.224)	0.1 (0.224)
987	0.0005	1.00	0.0005	0.702 (0.202)	0.467 (0.361)	0.938 (0.057)	0.6 (0.418)	0.513 (0.366)
987	0.0005	1.00	0.0010	<b>0.717 (0.126)</b>	0.433 (0.253)	<b>1.0 (0.0)</b>	<b>0.8 (0.447)</b>	<b>0.56 (0.318)</b>

