Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

Master Thesis Computer Science

Encoding electronic health records with temporal transformers for clinical decision support

Simon Frank

23.05.2024

Reviewers

Prof. Dr. Carsten Eickhoff
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

Prof. Dr. Nico Pfeifer
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

Supervisors

Dr. George Zerveas
Department of Computer Science
Brown University

Dr. Augusto Garcia
Department of Medicine
University of California San Francisco

Frank, Simon:
Encoding electronic health records with temporal transformers for clinical decision support
Master Thesis Computer Science
Eberhard Karls Universität Tübingen
Thesis period: 01.12.2023-01.06.2024

Abstract

This thesis presents a novel approach to online multi-horizon clinical decision support using machine learning models trained solely on unstructured textual content and corresponding temporal data from electronic health records (EHRs). Models that leverage clinical notes in EHRs can support clinical decisions through tasks such as estimating the probability of patient mortality. However, existing models are often trained to predict mortality for fixed, pre-specified time points, limiting their practical utility. The designed architecture employs hierarchical transformers with a late fusion strategy, leveraging a pre-trained Clinical Longformer for note embeddings to reduce computational overhead. The temporal transformer enables the model to operate in an online mode and predict mortality in multiple rolling time horizons based on the textual content available in a patient's EHR at any given point during their stay in the Intensive Care Unit (ICU).

A series of experiments were conducted with the objective of identifying the most effective training schemes and architectures for the purpose of accurately representing the evolving patient's state as extracted from textual notes. A multi-task learning objective was introduced to jointly predict mortality for all horizons, including short-term, medium-term, and total mortality. This objective enables the model to focus on forecasting patient mortality across a range of time horizons. The optimal results were achieved when the model was trained with a multi-tasking objective and when the model was trained to learn a shared patient representation for all time horizons, with the longer-term patient status being conditional on the shorter-term patient status. This performance improvement is most pronounced for the more complex short-term prediction time horizon. Moreover, the analysis demonstrates the efficacy of the model in diverse patient populations, delineating specific cohorts where performance is particularly robust, offering valuable insights for clinical applications. For example, the model demonstrates superior performance in predicting mortality among patients under the age of 40 with infectious and parasitic diseases and neoplasms for short-term predictions.

Zusammenfassung

In der vorliegenden Arbeit wird ein neuartiger Ansatz für die klinische Entscheidungshilfe präsentiert, bei dem Modelle des maschinellen Lernens ausschließlich auf Notizen und den entsprechenden Zeitstempeln aus elektronischen Patientenakten (EHRs) trainiert werden. Modelle, die klinische Notizen in EHRs nutzen, können klinische Entscheidungen durch Aufgaben wie die Abschätzung der Wahrscheinlichkeit der Patientensterblichkeit unterstützen. Bestehende Modelle werden jedoch häufig für die Vorhersage der Sterblichkeit für festgelegte, vorab spezifizierte Zeitpunkte trainiert, was ihren praktischen Nutzen einschränkt. Die entworfene Architektur verwendet hierarchische Transformers mit einer 'late fusion'-Strategie, welche einen vortrainierten Clinical Longformer nutzt, um den Rechenaufwand zu reduzieren. Der 'Temporal Transformer' ermöglicht dem Modell die Ausführung in einem Online-Modus und die Vorhersage der Sterblichkeit in mehreren sich verschiebenden Zeithorizonten auf Basis des Textinhalts, welcher zu einem bestimmten Zeitpunkt während des Aufenthalts auf der Intensivstation in der elektronischen Patientenakte verfügbar ist.
Im Rahmen einer Reihe von Experimenten wurde das Ziel verfolgt, die effektivsten Trainingsmethoden und -architekturen zu identifizieren, um den sich entwickelnden Patientenzustand, wie er aus Notizen extrahiert wurde, möglichst exakt zu repräsentieren. Es wurde ein Multitasking-Lernziel eingeführt, welches die Sterblichkeit für alle Horizonte gemeinsam vorhersagt, einschließlich der kurz- und mittelfristigen sowie der Gesamtsterblichkeit. Das genannte Lernziel erlaubt dem Modell eine Fokussierung auf die Vorhersage der Patientensterblichkeit über eine Reihe von Zeithorizonten. Die besten Ergebnisse wurden erzielt, wenn das Modell mit einem Multitasking-Ziel trainiert wurde und wenn das Modell so trainiert wurde, dass es eine gemeinsame Patientendarstellung für alle Zeithorizonte lernt, wobei der längerfristige Patientenstatus vom kürzerfristigen Patientenstatus abhängt. Die signifikanteste Leistungssteigerung ist für den komplexeren kurzfristigen Vorhersagezeithorizont zu verzeichnen. Des Weiteren offenbart die Analyse die Effektivität des Modells in diversen Patientengruppen und spezifiziert exemplarische Kohorten, in denen die Performance besonders robust ist. Dies ist von großem Nutzen für die klinische Anwendung. Das Modell zeigt beispielsweise eine überlegene Leistung bei der Vorhersage der Sterblichkeit von Patienten unter 40 Jahren mit infektiösen und parasitären Erkrankungen und Neoplasmen für kurzfristige Vorhersagen.

Acknowledgements

I would like to express my profound gratitude to Professor Eickhoff and Professor Pfeifer for their invaluable support and for providing me with the invaluable opportunity to undertake this research. Their guidance and insightful feedback have been instrumental throughout the journey of writing this thesis.
Furthermore, I am also immensely grateful to my supervisors, Dr. Zerveas and Dr. Garcia, for their unwavering support, expert advice, and the constructive discussions we had during our meetings. Their expertise and encouragement have significantly enhanced the quality of this work and have been instrumental in guiding the direction of my research.

Contents

List of Figures

List of Tables

List of Abbreviations

AUPRC	Area Under the Precision-Recall Curve
AUROC	Area Under the Receiver Operating Characteristic
BERT	Bidirectional Encoder Representations from Transformers
DRG	Diagnosis-Related Group
ECE	Expected Calibration Error
EHR	Electronic Health Record
ICD	International Classification of Diseases
ICU	Intensive Care Unit
LLM	Large Language Model
LSTM	Long-Short Term Memory
MIMIC	Medical Information Mart for Intensive Care
MLM	Masked Language Modeling
MLP	Multilayer Perceptron
NLP	Natural Language Processing
NPS	Next Sentence Prediction

Chapter 1

Introduction

In recent years, digitized health care data has become increasingly available, leading to a proliferation of research in machine learning applications for health care. It is widely accepted that such applications hold great potential for clinical decision support through tasks such as early triage, risk assessment, and prediction of patient re-admission [21, 47].
Estimating the mortality risk could be crucial in clinical practice for determining the frequency and intensity of in-hospital and post-discharge care or prioritizing allocating scarce resources [19]. When predicting mortality, two main parameters are considered: (1) the time the prediction is made during the patient's stay and (2) the time in the future to which the mortality prediction refers. Regarding the first parameter, most existing work only considers a few discrete points, usually 24 or 48 hours after admission [59]. This is often due to computational limitations in processing longer patient histories, such as for Transformer-based models. Similarly, concerning the second parameter, most existing models are trained to predict outcomes for a predetermined time point in the future, usually either at the end of the hospital stay or 30 days after discharge [59]. However, predicting mortality early in a patient's stay could be more inaccurate due to the varying length of stays, which can range from a few hours to multiple weeks. Predicting mortality at the end of a patient's stay can only be performed as a retrospective task and may not provide useful information for current patient care [59]. Predicting mortality would be more clinically useful if the risk estimate included a specific time frame or measure of urgency and was continuously updated as the patient's condition evolves and more information becomes available. Therefore, following prior work investigating online clinical decision support [59] this thesis presents novel models that predict mortality at multiple rolling time horizons, in addition to mortality at the end of the patient's stay in the Intensive Care Unit (ICU). These models provide updated 'online' estimates based on the information available in the patient's Electronic Health Record (EHR) at any time during their stay.
As part of their daily clinical routine, medical professionals produce a wealth

of free-text clinical notes, such as radiology and cardiology reports. These notes provide expert care staff with a synopsis of the most important aspects of a patient's physiology and are, therefore, highly descriptive sources of medical information [18]. With the recent rise of pre-trained, Transformer-based language models, an increasing number of models in the literature have used free-text notes [13, 25]. While these models are well-suited for natural language processing and perform well when fine-tuned on clinical notes, processing EHRs can be particularly challenging. The length of textual content in EHRs varies greatly, ranging from a few sentences to tens of thousands of sentences. Newer notes can update information in older notes, while some older notes can remain valid and relevant until the end of the stay. These characteristics have a significant impact on the performance of online predictions. It is clear that there is a need to develop models that are well-suited to the idiosyncrasies of EHRs and can effectively leverage their textual content. These models should also be compatible with the goal of online clinical decision support. To handle the unique characteristics of EHR data, the models use the textual information of the notes and the corresponding temporal information of each note. This thesis aims to lay the foundation for the under-explored research field of Transformer-based models that take textual and temporal input. The focus of this work is the development and investigation of such models.
The thesis is structured as follows. It begins with Chapter 2, which offers background information on Transformers, Uncertainty, and clinical decision support. Chapter 3 then introduces the dataset used throughout the thesis. The models developed for online clinical decision support are presented in Chapter 4, along with a detailed description of the training methods. The models are compared and evaluated in Chapter 5. Finally, the thesis concludes with a summary of the findings and an outlook on future research.

Chapter 2

Background

The following chapter provides background information to better understand the underlying theory of this thesis. First, the Transformer architecture with BERT as a special version will be explained. Secondly, information about model uncertainty will be provided. Finally, the related work section will provide an overview of clinical decision support.

2.1 Transformer

Transformers [52] are a prominent deep learning architecture, initially proposed as a sequence-to-sequence model for machine translation. Later works show that Transformer-based pre-trained models [46] can achieve state-of-the-art performances on various tasks; thereby, Transformers have become the dominant architecture in natural language processing (NLP). In recent years, Transformers have also been adopted in computer vision for tasks such as image generation [42] and object detection [7].

2.1.1 Architecture

The following section provides an overview of the Transformer architecture's key components pertinent to understanding this thesis.
The vanilla Transformer [52] comprises an encoder and a decoder. Each encoder block primarily consists of a multi-head self-attention module and a position-wise feed-forward network (FFN). A residual connection [23] is employed around each module to improve performance in a deeper model, followed by Layer Normalization [5]. In addition to the encoder, the vanilla Transformer also includes a decoder block with cross-attention modules. Furthermore, the self-attention modules in the decoder are adapted to prevent each position from attending to subsequent positions.

The Transformer model utilizes an attention mechanism with a Query-Key-Value (QKV) model. The attention function is computed on a set of queries simultaneously using packed matrix representations of queries $Q \in \mathbb{R}^{N \times d_k}$, keys $K \in \mathbb{R}^{M \times d_k}$ and values $V \in \mathbb{R}^{M \times d_v}$. The scaled dot-product attention employed by the Transformer is given by:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V = AV \tag{2.1}$$

where N and M denote the lengths of queries and keys; d_k and d_v denote the dimensions of keys (or queries) and values. A is often called the attention matrix.
The Transformer utilizes multi-head attention instead of a single-attention function. This involves projecting the original d_m-dimensional queries, keys, and values into d_k, d_k, and d_v, respectively, using h different learned linear projections. The value of h represents the number of heads. The attention function, according to equation 2.1, is then computed in parallel on each of these projected versions of queries, keys, and values, resulting in d_v dimensional output values. The outputs are concatenated by the model and projected back to a d_m-dimensional representation.

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.2}$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$. There are three types of attention in terms of the source of queries and key-value pairs:

- Self-attention: All keys, values, and queries come from the same place $Q = K = V$.

- Masked Self-attention: The self-attention is restricted such that the queries at each position can only attend to all key-value pairs up to a defined position. To enable parallel training, this is typically done by applying a mask function to the unnormalized attention mask $\hat{A} = \exp(\frac{QK^T}{\sqrt{d_k}})$, where the illegal positions are masked out by setting $\hat{A_{ij}} = -\infty$

- Cross-attention: The queries are projected from the outputs of the previous (decoder) layer, whereas the keys and values are projected using the outputs of the encoder.

[33]

2.1.2 BERT

In recent years, other variations have emerged in addition to the vanilla Transformer, such as the Encoder-only model, which uses the Transformer encoder as its backbone architecture. These models are predominantly used for language representation. An example of this category is BERT [13] (Bidirectional Encoder Representations from Transformers). Unlike some recent language representation models, BERT is specifically designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. The Bert model was pre-trained with two tasks: masked language modeling (MLM) and next sentence prediction (NSP). During MLM, 15% of the input words are masked and the model must predict the masked words. For NSP, two sentences are concatenated, and the model must predict whether they follow each other. BERT is a simple yet powerful concept. It achieved new state-of-the-art results on eleven natural language processing tasks, for example, pushing the GLUE score to 80.5% (a 7.7% absolute improvement) [13].
Subsequent research explored the possibility of further pretraining BERT with domain-specific corpora, such as medical text, to improve the quality of encoded representations and, thus, downstream task performance. One such model is ClinicalBERT [2], which was trained using text from all note types in the MIMIC-III dataset.
Despite achieving new state-of-the-art results, Transformer-based models like BERT are limited in their ability to process long sequences due to their self-attention operation, which scales quadratically with the sequence length. This design limitation restricts ClinicalBERT's attention window to 512 tokens or about 380 words in a clinical note tokenized by ClinicalBERT's vocabulary. To address this limitation, Longformer [6] was introduced. It utilizes an attention mechanism that scales linearly with the sequence length. Longformer's attention mechanism is a drop-in replacement for the standard self-attention and combines a local windowed attention with a task-motivated global attention. Longformer was evaluated for character-level language modeling and achieved state-of-the-art results on text8 and enwik8.
To address the limited input length of 512 tokens of the ClinicalBERT, a similar encoder was trained based on the Longformer architecture, with a maximum input limit of 4,096 tokens, called Clinical-Longformer [32]. The Clinical-Longformer study compared model performance with the ClinicalBERT model on several public datasets, including MIMIC-III. In the context of the MIMIC-III-based binary classification of acute kidney injury prediction (MIMIC-AKI [36]), the study reported an increase in AUROC of 0.762 compared to 0.738 for ClinicalBERT and 0.514 for BERT. The distinction between the base model and its clinically optimized counterpart is aligned with anticipated performance enhancements, including a modest additional benefit derived from long atten-

tion. Nevertheless, long attention has been subjected to scrutiny concerning accuracy [57] and computational cost [12]. Generally, a hierarchical pooling of representations with smaller attention windows remains competitive for most NLP tasks. Furthermore, this approach offers the additional benefit of lower training times and hardware requirements. Consequently, compared to windowing and pooling techniques, the potential contribution of long attention remains to be determined.

2.2 Uncertainty

Clinical decision support is a high-stakes decision-making application. Using model uncertainty to decide when to trust the model is crucial, as mistakes can have significant costs. The goal of model uncertainty is to *make systems know when they do not know*. There are two possible sources of uncertainty: Epistemic uncertainty, and Aleatoric uncertainty.
Informally, Epistemic uncertainty can be described as "I am not sure because I have not seen it before." and Aleatoric uncertainty as "I have experienced it before, I know what I am doing, but I think there is more than one good answer to your question, so I cannot choose just one.". To address Aleatoric uncertainty, the formulated model architecture accommodates multiple plausible outputs, which is common for classifiers. The learning strategy adopted allows the model to learn multiple plausible outputs instead of being restricted to just one, as is the case with cross-entropy loss for classification.
Epistemic uncertainty refers to the uncertainty in predictions due to the possibility of multiple models fitting the training data or insufficient data to distinguish between correct and incorrect models. There are two possibilities:

1. The training set size is too small, resulting in a high variance of the estimator.

2. The training data distribution does not cover certain significant regions in the input space, resulting in underexplored areas.

Ensembling is a highly successful method for addressing Epistemic uncertainty. Ensemble learning is usually performed as follows (popularized by Balaji et al. [29]):

1. Select M different random seeds. These are different starting points for the optimization.

2. Train the M models regularly

As the loss landscape is highly non-convex, optimization results in different local minima depending on the starting point, leading to a diverse set of models. After training the M models, the final prediction is obtained by averaging

the predictions of all ensemble members. Ensembles are known to improve accuracy. The rationale behind using an ensemble of models is that individual models tend to make distinct errors and overfit in different ways. By averaging their predictions, this variability is reduced, resulting in improved accuracy on test data.
In addition to reducing the model uncertainty, it is also crucial to evaluate the model uncertainty. One way to assess the uncertainty of a model is to quantify its calibration. To evaluate and quantify the calibration, let the input be $x \in X$, the output be $y \in Y = \{1, ..., K\}$ (multi-class classification problem) and the model output be a pair of the class prediction and the confidence estimate $h(x) = (\hat{y}, c(x))$. A model is perfectly calibrated if $P(\hat{Y} = Y|C = c) = c \quad \forall c \in [0, 1]$ [20]. Intuitively, the probability of correct prediction for confidence level c should be c, as it should accurately reflect the probability of correctness. For example, predictions for any sample in the dataset with a confidence score of $c = 0.8$ should be correct 80% of the time. A procedure that checks for this can be given as follows:

1. Collect all samples in the test dataset with confidence score $c = 0.8$
2. Compute the accuracy across all the samples
3. Check whether this gives us 80% accuracy,

Model calibration quantifies the deviation of the model from perfect calibration. However, the rough outline discussed above has a problem: the samples never have the exact same confidence score, so the model accuracy cannot be calculated in this way. One possible solution is to implement binning, which is a technique used to group data into intervals. The Expected Calibration Error (ECE) [40] metric is a method that utilizes binning to approximate model calibration:

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)| \tag{2.3}$$

where

$$acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} 1(\hat{y}_i = y_i),$$

$$conf(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} c_i$$

$acc(B_m)$ represents the accuracy of correct predictions in the mth bin, while $conf(B_m)$ represents the average confidence in the same bin. To ensure a more precise representation of the confidence values in this range, the average of the confidences in the bin is taken. It is also weighted by the bin size for the correct approximation of the expectation: $\hat{C}(c) = \frac{|B_m|}{n}$. The ECE is a

weighted average of the bin-wise miscalibrations.
In practice, the ECE can be calculated as follows:

1. Train the neural network on the training dataset.
2. Using the test dataset to create the predictions and confidence estimates.
3. Group the predictions into M bins based on the confidence estimates. Define bin B_m to be the set of all predictions $(\hat{y}_i, c_i)$ for which it holds that
$$c_i \in \left(\frac{m-1}{M}, \frac{m}{M}\right].$$
4. For each bin B_m compute the accuracy and confidence using the formulas for $acc(B_m)$ and $conf(B_m)$.
5. Compute the ECE by taking the mean over the bins weighted by the number of samples in them.

[39]

2.3 Related Work

Sequential physiological data in EHRs, such as heart rate or blood pressure, have often been used as numerical input features to develop time series classification models [1, 9, 21, 37]. In recent years, many neural sequential models have been developed for classification tasks that involve textual records. This is due to the success of neural models in the field of natural language processing (NLP). Word embeddings [38, 44] have been shown to be effective semantic representations of words that can scale to vocabularies of hundreds of thousands of terms across various domains [3, 60]. The standard methodology for neural textual EHR classification is first to derive a fixed-length patient (or admission) vector from the text and then predict a label, such as mortality or ICD codes, using the vector as a feature. Rajkomar et al. [47] show that such vectors can indeed provide a meaningful patient representation that generalizes to various prediction tasks. They combine a vector, which is the sum of all word embeddings in a medical note (event) with other numerical features, and use it as an input to a long-short term memory (LSTM) network and a Time-Aware Neural Network with attention.
Capturing interactions between distant words is necessary when computing a fixed-length patient vector using clinical notes. The need to model this long-range structure makes clinical notes suitable for contextual representations like bidirectional encoder representations from Transformers (BERT [13]). Therefore, Huang et al. [25] pre-trained BERT using clinical notes and fine-tuned

the network to predict hospital readmission. They predict 30-day hospital readmission at various time points of admission, including early stages and at discharge, solely based on the clinical notes. This model outperforms several baselines in predicting hospital readmission within 30 days using both discharge summaries and the first few days of notes in the ICU based on various clinically motivated metrics.

Following the release of pre-trained large language models (LLM) such as LLaMA [51], they have also been utilized for clinical decision support. Wang et al. [53] present DRG-LLaMA, an advanced LLM that has been fine-tuned on clinical notes to enhance Diagnosis-Related Group (DRG) assignment. This is a crucial aspect of the U.S. inpatient payment system. The model outperformed previous leading models in DRG prediction, such as ClinicalBERT.

Recently, Duan et al. [16] achieved better or competitive performance in assigning International Classification of Diseases (ICD) codes to clinical notes by using multi-hop label-wise attention. The study demonstrates the effectiveness of this approach in improving the accuracy of ICD code assignments. While developing a new multi-hop label-wise attention model to mimic the human reading process, the input to this module is a concatenation of encoded chunks of the clinical notes. Each chunk is independently encoded using a pre-trained XLNet [58] of [34].

In addition to the clinical notes, some models employ sequential physiological data in conjunction with the textual records. Wang et al. [54] introduce a novel, general, and unified pretraining framework called MEDHMP, specifically designed to make use of this information. While they are using an LSTM to model the temporal characteristics of the clinical monitoring readings like heart rate and respiratory rate, to create a representation of all the clinical notes, they directly use a pre-trained domain-specific encoder (Clinical-t5 [30]). There exist a variety of models that utilize clinical notes for a range of tasks, yet the temporal information associated with these notes is not incorporated.

Although mortality prediction has received significant attention, it is typically approached as a task performed at specific time points after admission, using either the first notes or all notes, and intended to provide predictions for the end of the stay or a few time points after discharge [26]. For instance, Grnarova et al. [19] developed a two-layer convolutional neural network to predict ICU mortality rates based on a patient's EHR data. The model predicts whether the patient will die during their hospital stay, within 30 days after discharge, or one year after discharge. Their model outperforms existing bag-of-words approaches and the popular doc2vec neural document embedding technique on all three tasks. Yuqi et al. [48] utilized this architecture and extended it to a multi-task learning setup, demonstrating minor but consistent improvements. Sushil et al. [50] employed a stacked denoising autoencoder and a paragraph vector model to learn task-independent dense patient repre-

sentations directly from clinical notes. They evaluated these representations, among other things, in a supervised setup to predict in-hospital mortality, mortality 30 days and one year after discharge. Hashir et al. [22] developed a hierarchical architecture consisting first of convolutional layers and subsequently of recurrent layers. They predicted in-hospital mortality using only the unstructured clinical notes from three different time windows: 12h, 24h, and 48h after ICU admission. They outperformed severity scores and clinical time series on the same cohort, demonstrating the potential of unstructured data to improve mortality prediction. Furthermore, the mortality prediction models utilize solely the clinical notes, excluding the corresponding temporal information. On the other hand, studies that account for temporality and use data from the first n hours of the stay have used structured data as input. For example, Lyu et al. [35] use 94 time series variables and a window size of 12 hours to predict whether patients will be discharged from the ICU in a stable state ("24h Discharge") or die within the next 24 hours ("24h Mortality").
Based on previous work focusing on a hierarchical convolutional neural network architecture [59], this work frames mortality prediction as a multi-task objective through the use of multiple rolling time horizons, training Transformer models on clinical notes and their corresponding timestamps through growing observation windows and dynamic labels. As a consequence, the models can successfully operate in an online regime, where predictions about mortality, both in multiple time horizons as well as for total mortality, can be made at any time during an ICU stay, using only the information available up to that moment and without any risk of "leaking" future information into the inference process.

Chapter 3

Dataset

This section first explains the general structure of the dataset, followed by an analysis to design an architecture suitable for it.

3.1 MIMIC-III

The experiments in this work utilized data from the MIMIC-III ('Medical Information Mart for Intensive Care') database [27], which is a large, single-center database containing information on patients admitted to critical care units at a tertiary care hospital. This dataset is still the most commonly used in the field of research, and there are also BERT models trained on this dataset, as shown in section 2.3. The database includes data from 46,520 patients over 58,976 hospital stays. The data collected comprises vital signs, medications, laboratory measurements, observations, notes charted by care providers, fluid balance, procedure codes, diagnostic codes, imaging reports, hospital length of stay, survival data, and other relevant information. MIMIC-III is a relational database consisting of 26 tables. The tables are linked by identifiers, which typically have the suffix 'ID'. For instance, SUBJECT_ID refers to a unique patient, while HADM_ID refers to a unique hospital admission.
The dataset for model training was created using only the ADMISSION and NOTEEVENTS tables. The ADMISSION table includes a unique SUBJECT_ID for each patient, a HADM_ID for each stay of a patient in the ICU, and the admission time. It also includes the time of death if the patient died in the hospital; otherwise, this entry is marked as NAN. The NOTEEVENTS table includes unstructured textual notes written by healthcare providers, such as physicians, nurses, and social workers. It also contains the HADM_ID to map each note to the corresponding EHR and the chart time of the note. However, it is essential to note that the hour component of the timestamp of each note may only sometimes be accurate due to the process by which the dataset was created. Therefore, this information will not be used in this work, and the

smallest unit of time considered is one day. To protect privacy, the same random number is added to the year component of the timestamps of each unique admission. Thus, the provided time information is relative, and the model will only use the days elapsed between the admission and the note. However, this setup also prevents the learning of spurious correlations between the year of admission and the likelihood of death.
To define cohorts for detailed model analysis based on patient information, the PATIENTS and DIAGNOSES_ICD tables are used. The PATIENTS table contains the SUBJECT_ID, gender, and date of birth. DIAGNOSES_ICD contains the sequence of hospital-assigned diagnoses, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system of the 9th edition for each admission. The ICD is a globally used medical classification maintained by the World Health Organization (WHO). The International Classification of Diseases, 9th Revision (IDC-9), comprises 17 chapters, each covering a specific group of diseases. For instance, Chapter 1 pertains to *infectious and parasitic diseases* and includes ICD codes 001 to 139 [8, 27].
Following [18] and [19], the data was filtered to only include adults ($\geq$18 years old) with a single hospital admission. Notably, any notes from the "Discharge Summary" category were excluded, as this summary explicitly contains information on the discharge or death of patients. However, other clinical notes in an EHR may contain additional references to a patient's death or discharge, which can inadvertently reveal target label information. Examples of such references include 'time of death', 'post-mortem', 'discharge paperwork', 'comfort measures only', 'deceased', 'expired', and others. After verifying that a patient's survival is strongly correlated with the presence or absence of specific references, a list was compiled of these 'survival' and 'death' references. Notes containing any of these expressions were filtered using regular expressions. A total of 897,111 clinical notes were used in this study, belonging to 30,849 distinct patients (equivalently, EHRs). The admission records reported that 13.4% of patients had died.

3.1.1 Dataset Analysis

As a first step, the number of notes per admission is analyzed (see Figure 3.1). There is a significant variance in the number of notes per admission. The shortest EHR has only one note, while the largest has 897. On average, each admission has 26.33 notes with a standard deviation of 42. Despite the significant variance, the majority of EHRs do not have an excessive number of notes, as 95% of admissions have 90 notes or less. Another important dimension to analyze is the distribution of notes over time. Figure 3.2 displays the number of notes per day elapsed since the first note. Most notes are taken at the beginning of the stay, with a significant drop after that, likely due to

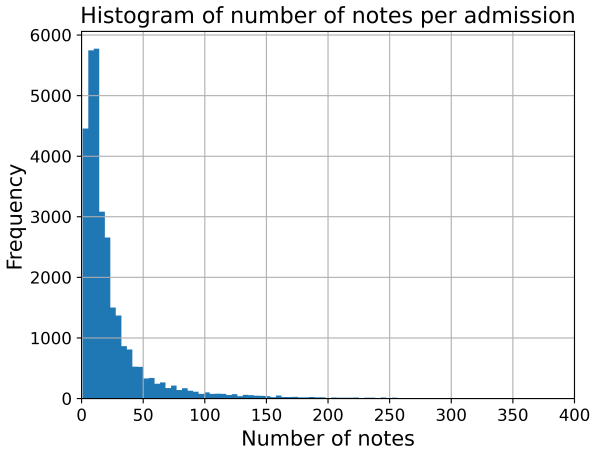

Figure 3.1: Histogram of the number of notes per admission

the short duration of most stays. 99% of the notes were taken before the 50th day, and the last note was taken 206 days after the first note. In addition

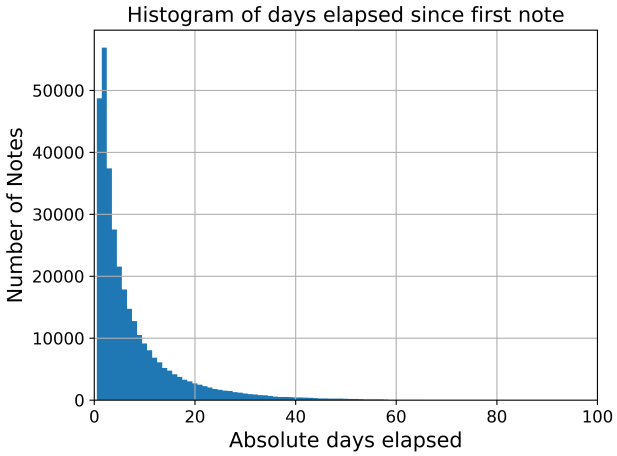

Figure 3.2: Histogram of the number of notes per day elapsed since the first note

to analyzing the above, it is also important to analyze the notes themselves. The histogram in Figure 3.3 shows the number of tokens per note using a BERT Tokenizer. Due to the time-consuming nature of tokenization, only a randomly chosen sub-sample of 5000 notes was used to create the histogram. On average, a note contains almost 600 tokens, with a large standard deviation of more than 600. More than 99% of the notes contain less than 3200 tokens. The analysis of MIMIC-III notes provided valuable insights into the dataset's structure and serves as the foundation for this work. The goal is to build a well-suited model for the unique properties of MIMIC-III notes and their

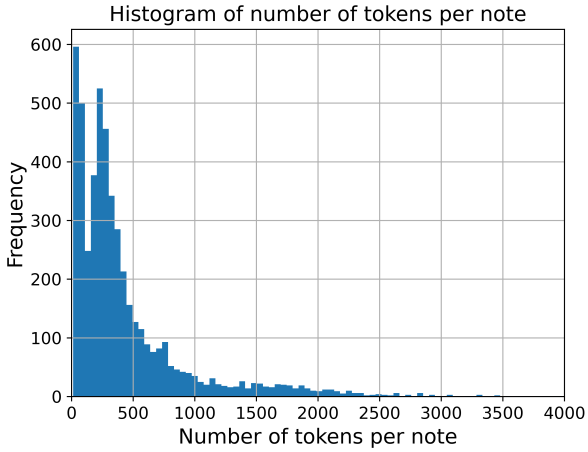

Figure 3.3: Histogram of the tokens per note using a random sub-sample of 5000 notes

temporal distribution.

Chapter 4

Methods

This section describes the rolling horizon ICU mortality prediction task and explains how the data is used to implement a solution. It then proceeds to describe the model and its architecture, which is designed to meet the specific requirements of the task and dataset and explains how the data is fed into the model. In addition, it outlines the methodology used in the experiments.

4.1 Task

The task on which the model will be trained and evaluated is rolling horizon ICU mortality prediction, where the aim is to estimate the mortality risk over different time horizons into the future and at different time points during a patient's stay in the ICU. The model uses clinical notes with their time stamp relative to admission time as input data X to predict mortality for different time horizons into the future. The model can make a prediction on each day of a patient's stay using all the clinical notes available up to that time point. Therefore, the observation window and corresponding input data X progressively increase throughout the patient's stay. Thus, the sample X corresponding to a single EHR is divided into n growing sub-samples for each of the n unique days. To provide better clinical utility, the model is trained to predict patient mortality over four different time horizons in the future and overall patient mortality. This results in five different target variables Y: Y_1 one-day mortality, Y_3 three-day mortality, Y_5 five-day mortality, Y_{10} ten-day mortality and Y_{Dead} total mortality. The prediction time point for the four different time horizons (Y_1, Y_3, Y_5, and Y_{10}) is always the beginning of the day. The five targets are 0 if the patient survives. If the patient dies during the stay, the targets Y_1, Y_3, Y_5, and Y_{10} can be different for each sub-sample X_n of the EHR because the prediction time point is different.
Figure 4.1 illustrates the setup explained above for a sample EHR of a patient with an eleven-day stay. The first row shows the EHR starting on day 0 with

the initial clinical note and ending with the patient's death on day 10, where each blue cross represents a clinical note. The second row illustrates the setup to predict day 0, with the information window containing only the notes from the admission day representing X_0. Each green arrow indicates the time the patient's status is checked to determine the label for Y_1, Y_3, Y_5, and Y_{10}. As the patient dies on day 10, Y_1, Y_3, Y_5 and Y_{10} are all 0 and only $Y_{Dead} = 1$. The third line shows the setup for day 1. The information window X_1 has grown and now contains the notes from days 0 and 1. Since the relevant time for Y_{10} is now after the patient's death, $Y_{10} = 1$ for X_1. Similarly to days 0 and 1, the information window expands for each subsequent day until the final day. Consequently, each X_n includes more information, and all Y values are updated accordingly. The final row displays the configuration of the last day, with the information window encompassing all clinical notes and all targets Y set to 1.

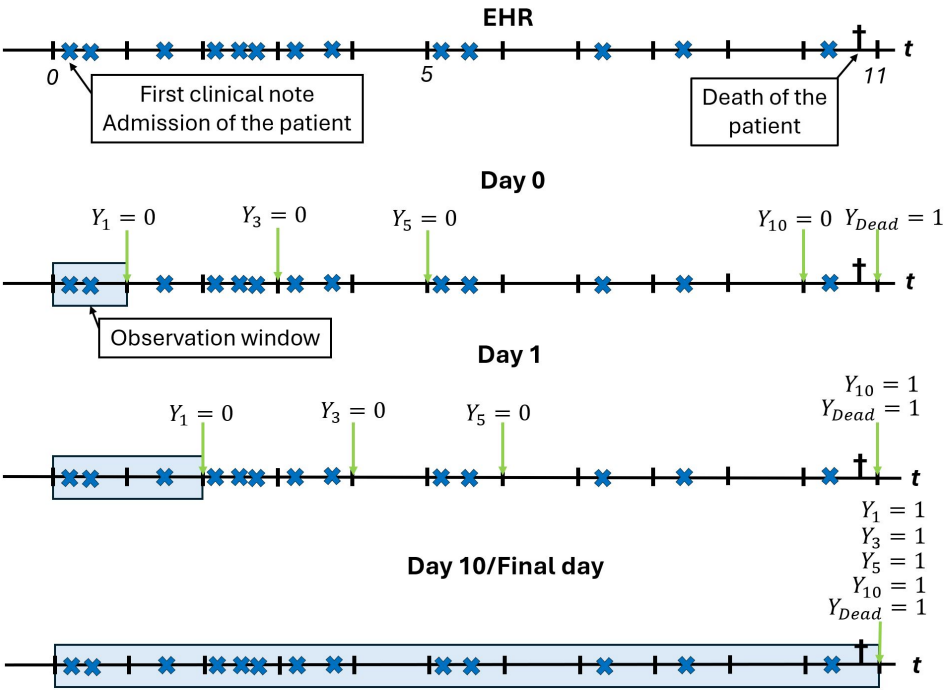

Figure 4.1: Rolling Horizon ICU mortality prediction

4.2 Model architecture

The following section provides a detailed explanation of the model's architecture and its rationale. On average, each note in the MIMIC-III dataset contains approximately 600 tokens and an EHR typically consists of around 26 notes. Therefore, at least 15,600 tokens must be processed to make a prediction, par-

ticularly for the final-day prediction. Additionally, some EHRs contain more notes, which results in a longer input sequence. If the entire text of the EHR is inputted into the Transformer model, a context length of more than 20,000 tokens is required. Recent years have seen significant efforts and improvements in handling the initial quadratic complexity in memory and FLOPs of Transformers, resulting in impressive outcomes [33]. With LongNet [14], it is now feasible to manage input sequences of more than 1 billion tokens. However, there is currently no open source Transformer model with the necessary context length that has already been pre-trained on medical data. Unfortunately, due to limited hardware resources, performing such pre-training within the scope of this thesis is impossible. Additionally, a pure textual Transformer cannot incorporate time information for each note through its architecture, but only through a prefix before each note. Therefore, a new architecture was developed that utilizes the relative time information of each note. This framework can also be applied to similar tasks in other domains.

In their 2022 survey on Transformers for time series, Wen et al. [55] reviewed various aspects of the Transformer architecture and highlighted its strengths and weaknesses. Positional embedding is one design choice that significantly impacts the model's performance. Although a sinusoid positional embedding can extract some positional information from time series, Li et al. [31] could not fully exploit the critical features of time series data. Timestamp information is often available when modeling time series in real-world scenarios, such as the timestamp for each note in the MIMIC-III dataset. These timestamps are informative in real-world applications but are rarely used in vanilla Transformers. To address this issue, Informer [61] proposed encoding timestamps as additional positional encoding using learnable embedding layers. The input representation of Informer comprises three distinct components: a scalar projection, the local timestamp (position), and global timestamp embeddings (minutes, hours, week, month, holiday, etc.). A comparable timestamp encoding scheme was employed in Autoformer [56] and FEDformer [62].

To also leverage the benefits temporal embeddings provide to a Transformer model, the architectural framework devised in this thesis is grounded on the late fusion approach. This approach integrates the temporal information through a temporal embedding after the computation of a feature vector for each data point. The architecture consists of two separate Transformer encoders. The first encoder only models the interactions of one data point. The second encoder models interactions between tokens from different temporal indices. This encoder's output token is used to perform the final classification task. A notable advantage of the late fusion methodology is its flexibility, allowing the initial encoder to be any generic encoder that has undergone pre-training.

This approach also produced good results in other domains. In [4] four Transformer based-model architectures were proposed for video classification, a specific time series task. The architecture that performed best on the small dataset

is also based on the late fusion approach. The Video Transformer Network, as presented in [41], similarly adopts the late fusion approach. Remarkably, it achieves a training speed that is 16.1 times faster and an inference speed that is 5.1 times faster, all while maintaining competitive accuracy when compared to other contemporary state-of-the-art methods.
The first encoder, the "Note Embedder", embeds each note independently, yielding a note-level representation. Subsequently, to this note-level representation a temporal embedding, which depends on the days elapsed since the first note and the timestamp of the specific note, is added. The augmented representation is then fed into the second encoder, a Transformer encoder, denoted as the "Temporal Transformer". The Temporal Transformer captures interactions among all tokens, considering each note's relative time information. A Multilayer Perceptron (MLP) readout is used to perform the final prediction task using an output token of the Temporal Transformer. The architectural design of the model allows it to capitalize on both the textual information embedded in each note and the additional temporal information provided by each note. This enables the model to consider textual information in the context of temporal factors. The design choices for the two transformers are elucidated in detail below.

4.2.1 Note Embedder

Given that all the information in a clinical note is relevant, truncating the note at the beginning or end is not advisable. As observed in section 3.1.1, the average note results in almost 600 tokens. In order to process these long notes and create a meaningful embedding for them, Clinical-Longformer [32] is used as the first encoder. Clinical-Longformer is a clinically enriched version of Longformer [6], which has been pretrained using MIMIC-III clinical notes. This makes it a suitable choice for the current task. It accepts up to 4,096 tokens as model input. Clinical-Longformer consistently outperforms Clinical-BERT [2] by at least 2 percent over 10 baseline data sets [32]. With 99% of the notes resulting in fewer than 3200 tokens, almost all notes can be processed without truncation.
The following workflow is applied to each note to compute the corresponding embedding. Firstly, a MIMIC-III-specific preprocessing step is applied to each note. During this preprocessing, the following list of terms is removed from each note: 'admission date:', 'discharge date:', 'date of birth:', 'sex: m', 'sex: f', 'incomplete dictation:', 'dictator hung up:', 'dictated by: medquist:', 'completed by:' to prevent any shortcut learning and introduction of bias into the model. Subsequently, the note is tokenized with the Clinical-Longformer tokenizer, truncating the end of the note if necessary. The `CLS` token is appended at the beginning and the `SEP` at the end of the tokenized note. The tokens are then fed into the pre-trained Clinical Longformer, and a forward pass of the

model is performed. The 768-dimensional feature embedding ϕ corresponding to the `CLS` token of the last hidden layer is extracted and used as the embedding for the whole note.
Without any further training of Clinical Longformer the note embeddings are precomputed for all notes. This significantly reduces training time. However, to still be able to modify the input of the Temporal Transformer according to its needs and to perform a nonlinear operation before the linear attention of the Temporal Transformer, the note embeddings are fed into an input projection $f(x)$, resulting in the final note embedding $\phi^{'}$, which serves as input to the Temporal Transformer.

4.2.2 Temporal Transformer

Before feeding the final note embedding $\phi^{'}$ into the Temporal Transformer, a fully learnable temporal embedding is added to the note embedding. The days elapsed between the first and current notes are the relevant temporal information used to select the appropriate time embedding. Considering Figure 3.2, there is a unique temporal embedding for the first 50 elapsed days. There is only one temporal embedding for all elapsed days between 51 and 100, and one temporal embedding for all elapsed days greater than 100. Given that most notes are recorded within the initial 50 days, there is enough data to learn meaningful temporal embeddings for each day individually. There are fewer notes for the following days, so they must be grouped together to learn meaningful temporal embeddings. To avoid using just one embedding for all the remaining days, there is a split at 100 days to distinguish between extended stays (50-100 days) and all stays longer than 100. The finest granularity in time at which note timestamps are reliable is one day; more exact timing information is often unreliable. Therefore, the smallest time unit for each note is a day, and only day embeddings are used. Additionally, other temporal information, such as month and year, is not used either, since the prediction of the model should only depend on the elapsed days. The model should not learn spurious correlations between mortality and specific periods in the past. There is no order between the notes within a day, so all notes of a day must be treated on equal footing. Therefore, no positional embedding is used for the Temporal Transformer, as this would introduce an arbitrary order. The Temporal Transformer consists of a standard Transformer encoder first introduced in [52]. It has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise wise fully connected feed-forward network.
Figure 4.2 shows the entire architecture with all its components making a prediction on day five of a sample EHR containing notes on days one, two, three, and five. The Clinical Longformer precomputes the embedding of a note ϕ. The input projection $f(x)$ transforms it into $\phi^{'}$, to which the corresponding

temporal embedding is added. A learnable CLS token is placed between the notes of each unique day, marking the boundary between different days. The same temporal embedding that is added to the notes before the CLS token is also added to the CLS token itself. For the final prediction, the embedding corresponding to the final CLS token at the output of the Temporal Transformer is passed to an MLP readout head, which consists of two dense layers with a ReLU function between and the sigmoid activation function to obtain a probability prediction. A typical EHR has records with multiple unique

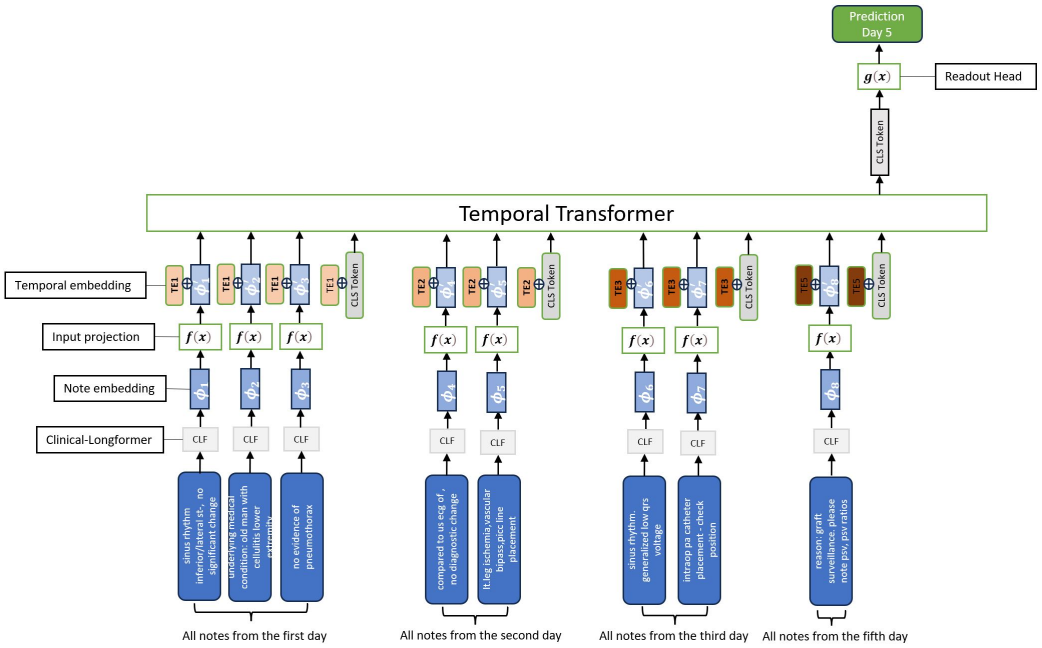

Figure 4.2: Model architecture

days. Thus, multiple CLS tokens will be in the input sequence of the Temporal Transformer, each with a unique target. As mentioned in section 4.1, the model should make a unique prediction for each individual day using all the notes up to the previous day, thus using the information of an EHR multiple times. The EHR should be processed in a batch to speed up the computation for these different targets and provide a more complete view of the loss landscape for each weight update. Therefore, the input sequence of the model obtained from an EHR is repeated in the batch dimension for each unique day. This results in an input sequence of shape $n \times T \times m$, where n is the number of unique days with at least one note, T is the length of the input sequence, which is equal to the number of notes in the EHR plus one CLS token for each unique day, and m is the dimensionality of the final note embedding ϕ' and the temporal embedding.

To ensure causality, the EHR should be processed as a single batch without leaking future information into the present. For instance, notes from day

three should not be used for predictions on day two. This can occur if the note embeddings after the `CLS` token, which serve as the input for the readout head $g(x)$, are used to compute the multi-head self-attention in the Temporal Transformer. To prevent this, a key padding mask is utilized in the multi-head attention. A padding mask for the key indicates which elements should be excluded from attention. The mask is defined so that, for example, only the tokens up to and including the first `CLS` token can be used for calculating attention weights in the Temporal Transformer for the first batch element. This means that subsequent tokens cannot be used and no information flows from the future to the present. The second batch element can only use all tokens up to the second `CLS` token. The following masks will be constructed in the same manner. The final element in the batch will utilize the entire input sequence. After processing the batch with the Temporal Transformer, only the `CLS` tokens corresponding to the batch number will be extracted. For example, the first `CLS` token will correspond to the first batch element, and the last `CLS` token will correspond to the last element in the batch. This will result in an output of dimensions $n \times m$, which will then be fed into the MLP readout head to make the final predictions of dimensions $n \times 1$.

4.3 Multi-tasking

As discussed in section 4.6.2, predicting a target such as mortality at multiple time horizons can be helpful for clinicians. To achieve this objective, the model will be trained on multiple sub-samples of the main sample (EHR), each with its own set of labels, such as 'mortality in 1 day'. If a patient's record is about to expire, only one of the sub-samples will have a positive mortality label. This sub-sample will terminate one day before the end of the EHR, while all the others will have a negative label. Therefore, the model parameters will be adjusted over multiple steps to penalize a positive mortality prediction for the same patient before encountering the single positive label. However, suppose the patient's death was not completely sudden and unexpected. In that case, there may be indications of their declining state in the clinical notes of the EHR several days before their death. In such cases, the model should be rewarded for learning to recognize these indications rather than penalized. Therefore, it is logical to move away from using a single binary label and instead train the models using a multi-task objective. This involves jointly predicting mortality at multiple time horizons, which increases the label information available during training. The model is trained to predict overall mortality starting from day 1 for a patient whose condition is deteriorating. As additional days pass, the labels for mortality in 10, 5, 3, etc. days will progressively switch from negative to positive.
In a survey of multi-task learning with deep neural networks [10], the authors

state that they believe that multi-task learning more accurately reflects the human learning process than single-task learning, as the integration of knowledge across domains is a central tenet of human intelligence. Such approaches offer advantages such as improved data efficiency, reduced overfitting through shared representations, and fast learning by exploiting auxiliary information. In computer vision, for example, many multi-task architectures focus on partitioning the network into task-specific and shared components. In this setup, a global feature extractor shared by all tasks first extracts a shared feature, followed by an individual output branch for each task. Dai et al. [11] introduce multi-task network cascades (MNCs). The architecture of MNCs also extracts shared features first. However, the output of each task-specific branch is appended to the input of the next task-specific branch, forming the "cascade" of information flow for which the method is named.
To implement a multi-task objective, one may reuse the same model architecture as in Figure 4.2 using the `CLS` token as the shared feature and modifying the readout head $g(x)$ by adding an additional task-specific branch consisting of an MLP followed by the sigmoid function (σ) to compute the probability for each additional target (time horizon) (Figure 4.3).

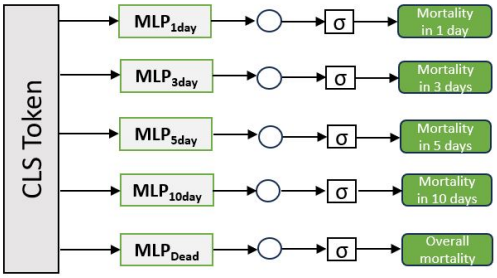

Figure 4.3: Readout head ($g(x)$) for the multi-tasking version 1

4.3.1 Conditionality constraints for multi-task objective

The previous section discussed how mortality prediction can be viewed as an independent multi-task objective. However, individual tasks/targets are not truly independent. To account for this, the model can be subject to the following consistency assumption as an inductive bias: a patient predicted to be dead in a certain time horizon should always be predicted to be dead at any later time. The model should predict a monotonically non-decreasing probability of mortality for labels in chronological order. This is achieved by imposing a soft conditionality constraint on predicting successive targets. The logit computed from the previous time horizon is added to the current logit (see Figure 4.4). The model parameters are learned to compute a residual term, which is added to the logit corresponding to the preceding target, in-

stead of computing the logit independently from the CLS token for each target. Although the subsequent target is conditioned on the previous logits through

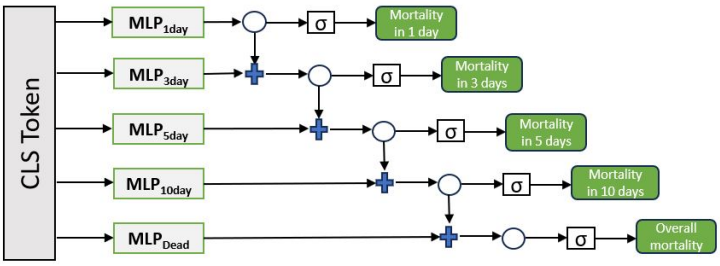

Figure 4.4: Readout head ($g(x)$) for the constraint multi-tasking version 1

a residual connection, the effect of this connection may be weaker than it is only conditioned on a one-dimensional logit. In addition, there is no direct parameter sharing in the readout head $g(x)$ for the different targets, forcing the model to learn a common feature representation. To mitigate these effects, a second multitasking architecture was designed (see Figure 4.5). In order to condition the subsequent target through a multidimensional residual connection, a multidimensional feature vector ϕ_1 is first computed with a two-layer MLP without reducing dimensionality. This feature vector can be seen as the predicted patient state in one day. Through the $MLP_{Prediction}$ the probability for the mortality in one day is computed. $MLP_{Prediction}$ is a two-layer MLP with the middle layer having half the input dimension. For the following day, MLP_{3day} calculates only the difference $\phi_2^{'}$ to predict the patient's status in three days (ϕ_2). The same $MLP_{Prediction}$ computes the probability of mortality in three days, forcing the model to learn a common patient representation on the ϕ feature vectors. Similarly, the prediction for the next target is calculated. Another multi-tasking approach is to not only condition on learned

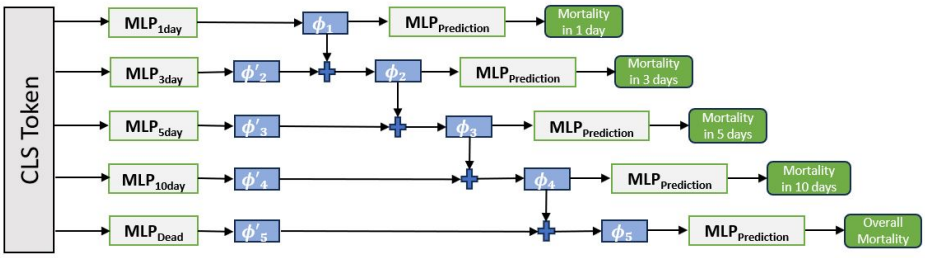

Figure 4.5: Readout head ($g(x)$) for the multi-tasking version 2

intermediate features but to use the previous feature representation of the patient in combination with the CLS token to predict the subsequent target (see Figure 4.6). In this architecture (multi-tasking version 3), as in version 2, the feature vector ϕ represents the patient status at a given time in the future,

for example, one or three days, and the same $MLP_{Prediction}$ always estimates mortality. However, MLP_{3day} receives as input the sum of the `CLS` token and the previous feature vector ϕ_1 to compute the patient status ϕ_2 directly without any additional residual connections. The subsequent targets are computed in the same way. The architecture of all MLPs is the same for multi-target versions 2 and 3.

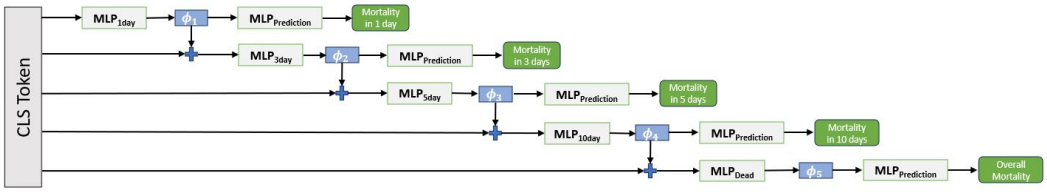

Figure 4.6: Readout head ($g(x)$) for the multi-tasking version 3

4.4 Information bottleneck

In the architecture depicted in Figure 4.2, only the final `CLS` token processed by the Temporal Transformer serves as input to the readout head $g(x)$. Therefore, the Temporal Transformer is trained to aggregate all patient information into a single output token. This inductive bias acts as a powerful information bottleneck. Due to the multi-head self-attention mechanism, the Temporal Transformer computes one output token for all input sequence tokens. To mitigate this information bottleneck, the average of all output tokens from the Temporal Transformer can be used as input to the readout head $g(x)$, as shown in Figure 4.7. This setup enables investigation of the impact of the information bottleneck on task performance.

4.5 Multi-sampling

With the setup explained above, there is a unique prediction for each day from the EHR of a patient, thus each day is equally contributing to the loss and thereby to the parameter updates. However, as patient stays vary in length, the model makes more predictions for patients with longer stays, resulting in implicit oversampling. This oversampling can lead to poorer performance in patients with shorter stays. It can also lead to poorer performance in predicting a patient's mortality during the first days of their stay, as these predictions contribute less to the overall loss of each patient during training. To mitigate these issues and emphasize the beginning of a patient's stay, the entire EHR can be segmented into partial samples (sub-samples). Each subsample should be terminated at a specific time t after admission, containing all clinical notes from

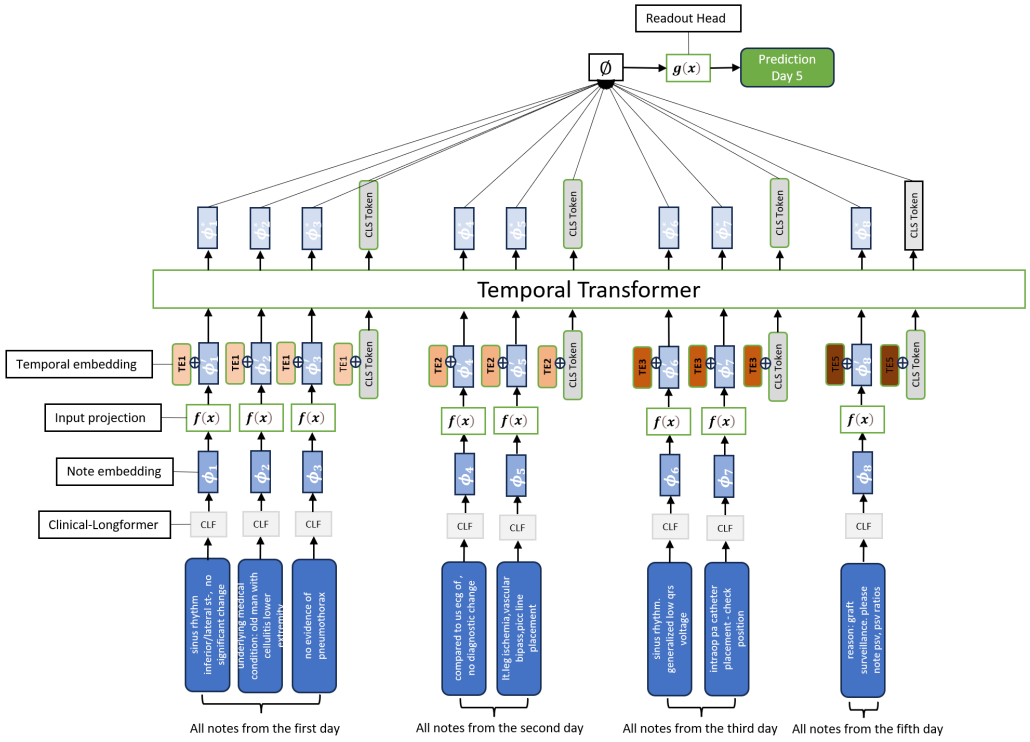

Figure 4.7: Model architecture without information bottleneck

the point of admission up to that time t . Therefore, during training with *multi-sampling*, each EHR will be segmented into the same number of sub-samples defined by a multi-sampling grid. The grid is defined by the time points each sub-sample terminates, denoted by t. For example, the multi-sampling grid $[0, 4, 8, end]$ creates four sub-samples. The first sub-sample contains only the notes from the day of admission, the second contains all notes until day four, the third contains all notes until day eight, and the last sub-sample contains all notes in the EHR. As explained in section 4.1, one prediction is computed for each day in each sub-sample.

4.6 Experiments

In the following, the experiments to obtain the results of Section 5 are explained in detail.

4.6.1 Data

To define a test set, 20% of the patients were randomly sampled. As a first step, the remaining 80% were divided further into a training set (80%) and

a validation set (20%), whereby the validation set was used to tune the hyperparameters of the models. As a second step, after the hyperparameters had been selected, all of these 80% were used to train the models, which were then evaluated on the held-out test set. In order to mitigate the significant imbalance in the class distribution, the majority class (mortality negatives) was down-sampled until a ratio of 5:1 was achieved concerning the minority class (mortality positives). Subsequently, the minority class was up-sampled by replicating positive samples until a final ratio of 2:1 was reached. This measure was critical for training well-performing models, especially regarding recall for the minority class (positives). The ratio of negative to positive samples in the test set was left unchanged; therefore, the test set follows the actual data distribution.

4.6.2 Models

To evaluate the robustness of the architecture to changes in the model configuration, experiments were conducted using two different versions of the model. The first version's Temporal Transformer has only one encoder layer, with an embedding and feedforward dimension of 768. The input projection $f(x)$ is a two-layer MLP with ReLu as the nonlinear activation function between the two layers. Thus, this model configuration is able to adjust the precomputed note embeddings ϕ in a nonlinear way before the linear multi-head self-attention. The Temporal Transformer of the second version consists of two encoder layers but only has an embedding and feedforward dimension of 330 for the single task and 300 for the multi-tasking setup due to limited hardware. The input projection $f(x)$ of this setup is a linear projection of the 768 dimensional note embeddings ϕ onto the 330/300 dimensions of the Transformer. These two versions balance the trade-off between the dimensionality of the Transformer dimensions and the number of encoder layers, which arises due to the limited hardware. Both models use six parallel attention heads. Due to hardware constraints, it was impossible to train a model with a Temporal Transformer that consists of more than two encoder layers.
As described in section 4.3, there are three different versions of multi-tasking, with the first version having a constrained and unconstrained option. Figure 4.8 displays the total number of trained models, with each leaf representing a unique model and indicating the number of parameters in millions for that model.

4.6.3 Training

As shown in Figure 4.8, there are ten different model combinations. Due to feasibility constraints, an extensive hyperparameter search was not conducted

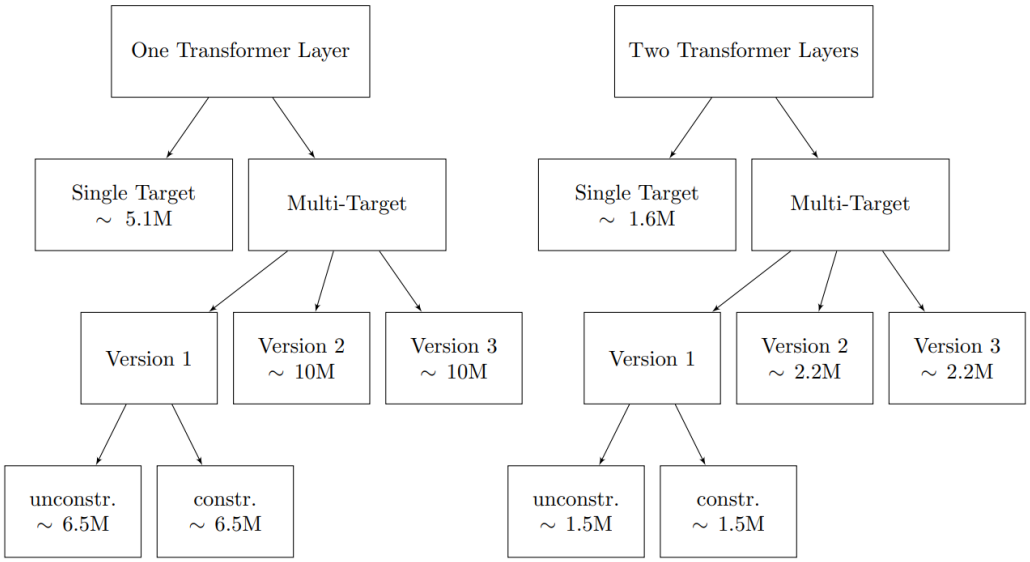

Figure 4.8: Overview of all the model versions trained during the experiments

for all of them. Therefore, the single target single-layer architecture without using *multi-sampling* was chosen as the default model version for most experiments. Predicting total mortality was selected as it demonstrates the best overall performance and is also the most universal task. It is reasonable to assume that the results obtained with this setup can be extrapolated to the other models.

The chosen performance metric is the Area Under the Precision-Recall Curve (AUPRC), which is suitable for datasets with pronounced class imbalance, such as the MIMIC-III dataset. This metric enables better differentiation between models and hyperparameters. During training, the AUPRC is calculated after each epoch for both the training and validation sets to monitor model performance. To reduce training time, only a randomly selected 20% subset of the training set is used for the model evaluation, in addition to the complete validation set.

The binary cross-entropy loss function was chosen for the prediction task, as it is binary. To ensure that the loss is not dependent on the number of predictions for one patient, the average of the individual losses was computed afterwards. For the multi-task setup, the average of the individual task losses was computed, as the model should treat all tasks equally. Each training run consists of 50 epochs, which is a reasonable length to obtain valid results. To determine the number of epochs used to train the final models with the complete training set, the epoch in which the corresponding model had the best performance on the validation set using all notes for prediction was selected. This approach helped to reduce the impact of overfitting. For the multi-tasking versions, only the performance on total mortality prediction was considered to determine the

number of epochs. The Adam [28] optimizer from PyTorch [43] was used. The best results were achieved using the optimizer without L2 regularization. Since dropout has proven to be an effective technique for regularization and preventing the co-adaptation of neurons [24], it was applied with the same rate P_{drop} in the input projection, Transformer encoder layer and the readout head. Vaswani et al. [52] applied dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks, therefore input-dropout was also applied to the sum of the note embeddings ϕ' and the temporal embedding. The best result for the single-layer Transformer for the total mortality task was achieved with a dropout rate of 0.3 and an input dropout rate of 0.15. For the multi-target versions a dropout rate of 0.35 and in input-dropout rate of 0.2 was used.
The effect of learning rate scheduling was also investigated. To do so, the initial learning rate was linearly increased for a set number of epochs until it reached a defined maximum. Afterward, it was linearly decreased to a defined final learning rate for the remaining training steps. Various configurations of this learning rate scheduling were tested, but the best performance was achieved with a constant small learning rate of $lr = 10^{-5}$.
The dataset's small size and limited variance lead to epistemic uncertainty, as explained in section 2.2. Ensembling is a useful technique for handling epistemic uncertainty. To create a model ensemble, four different random seeds for the data dispenser and model parameters were set. Then, the models were trained as usual. Due to feasibility constraints, only one ensemble was trained for the multi-tasking version 2, which achieved the best performance on the validation set during training.

Batchsize

Depending on the sampling method, the entire EHR of a patient or only a sub-sample up to a specific time point t (e.g., the first four days) is used as input for the model. To ensure consistency, a fixed patient size was used for each batch instead of a fixed batch size, as the number of unique days and predictions varied between patients. Due to hardware limitations, a GPU could only handle a patient size of one.
Smith et al. [49] showed the impact that batch size can have on training. Popel et al. [45] showed that batch size also impacts the performance of Transformer models. Therefore, an experiment with two patients in a batch and an experiment with three patients in a batch were conducted. In both experiments, the entire EHR of the patients was used in the batch. The resulting batch must be divided between several GPUs to run. This increased the total run-time of an epoch due to the increased overhead. There were no significant differences in performance between the three experiments. Therefore, a patient size of one was used for all experiments since the total run time per epoch is the shortest

in this setting.

Multi-sampling

To investigate the effect of *multi-sampling* explained in section 4.5 three different *multi-sampling* grids were tested:

1. $[0, 1, 2, 3, 4, 8, 16, 32, 64, end]$
2. $[0, 1, 2, 3, 4, 8, 16, end]$
3. $[0, 2, 4, 8, end]$

All three grids have a higher resolution at the beginning of a patient's stay, resulting in more sub-samples from the first days. This places greater emphasis on the initial days of a patient's stay. Each sub-sample of a patient is fed into the model as an individual batch, while the order of the sub-samples is shuffled. For example, the first batch contains only the notes from the first day and the second batch contains all the notes. Figure 4.9 displays the AUPRC for the single-layer Transformer for the total mortality prediction task with (red color) and without (blue color) the use of *multi-sampling* during training. The AUPRC is calculated at seven different time points during a patient's stay using the notes up to and including this day, e.g. at prediction time point 0 only the notes from the admission days are used and at "end" all the notes in the corresponding EHR are used. If *multi-sampling* is utilized during training, the model's performance on the training set is independent of the prediction time point, achieving an AUPRC of almost 1. Without *multi-sampling* the AUPRC is worse, mainly if the prediction time point is earlier during the patient's stay. On the validation set, the model performance using *multisampling* peaks after just a few epochs and subsequently drops. Without *multi-sampling* the performance peaks after about ten epochs and outperforms the other model for all prediction time points. The results in Figure 4.9 were achieved using the second *multi-sampling* grid. The results of the other two multi-sampling grids are similar. It can be concluded that through *multi-sampling* the model massively overfits to the data in the training set and is unable to perform well on the validation set. This effect may be because the model is trained more often on the first days of patient stays through *multi-sampling*, but there is not enough variation in these training samples, causing the model to fail to generalize well. *Multi-sampling* was not used to train the final models due to poor performance on the validation set.

Two-layer Transformer

The regularization hyperparameters were adjusted due to the significant difference in the number of parameters between the one- and two-layer versions

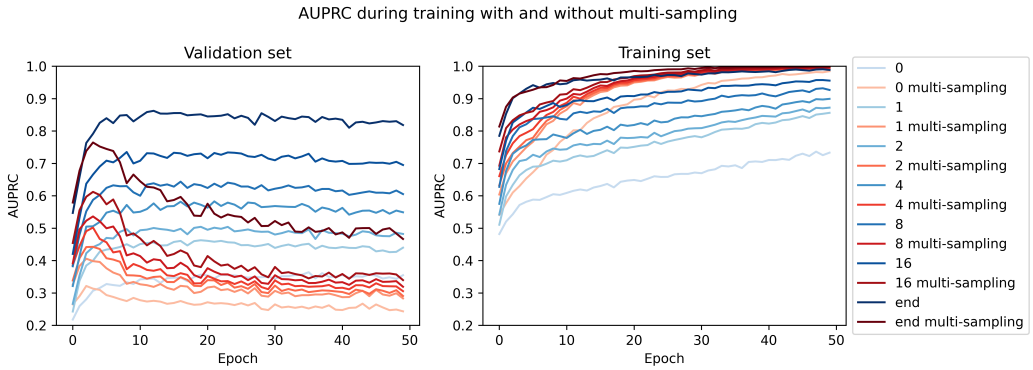

Figure 4.9: AUPRC during training for the single layer Transformer for the total mortality prediction task with and without the use of *multi-sampling*

of the Transformer. The best results were achieved with a dropout rate of 0.25 and an input dropout rate of 0.1. Figure 4.10 displays the AUPRC for the single- and two-layer Transformer versions for the total mortality prediction. The model with one Transformer layer outperforms the model with two Transformer layers. Although there is a noticeable difference in performance on the training set, the difference on the validation set is still relatively small. Similar results were obtained for the different multi-tasking versions. Despite

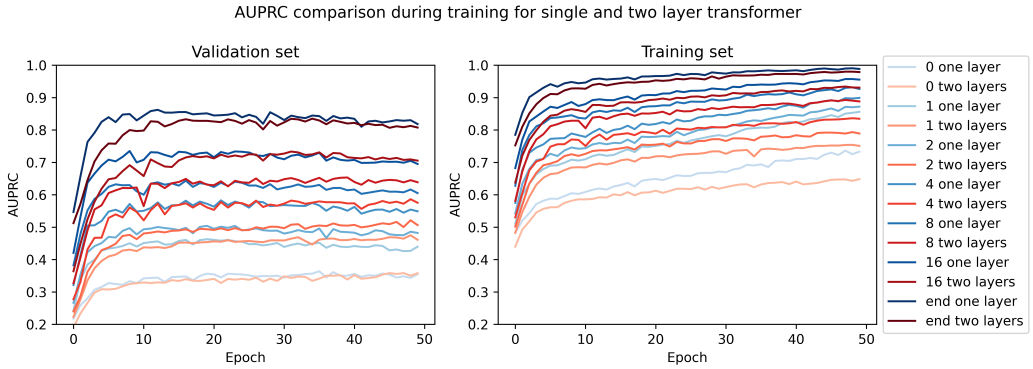

Figure 4.10: AUPRC during training for the one- and two-layer Transformer for the total mortality prediction task

having only around 23% of the parameters of the multi-target versions and approximately 31% of the single target version compared to the corresponding one-layer Transformer model, the two-layer version's performance on the test set was only slightly worse. It was found that the architecture is not sensitive to the number of model parameters and can achieve good results even with a small model. The final results were achieved using the one Transformer layer version to attain optimal performance.

Information bottleneck

Figure 4.11 shows the AUPRC for the single-layer Transformer for the total mortality prediction task with (blue) and without (red) use of the information bottleneck, as explained in Section 4.4. The model without the information bottleneck performs slightly better at later prediction time points on the training set. In contrast, the model with the information bottleneck performs better on the validation set. The information bottleneck is a strong inductive bias, reducing overfitting and improving the model's performance. The model architecture with the information bottleneck, as shown in Figure 4.2, is used to obtain the final results.

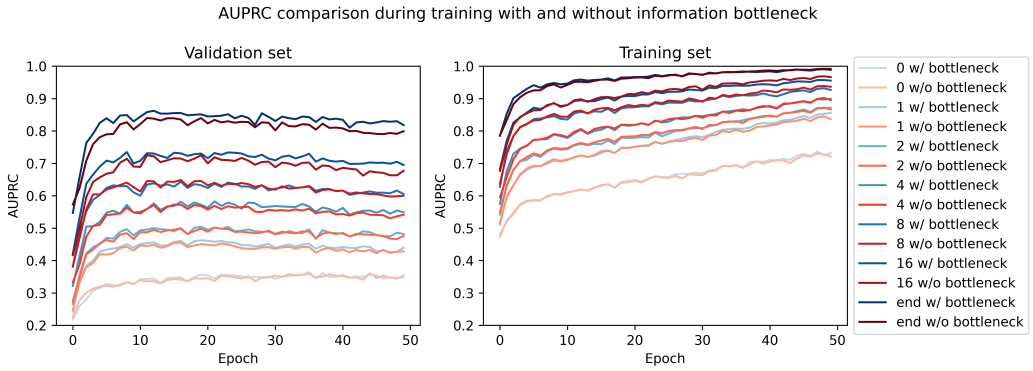

Figure 4.11: AUPRC comparison during training with and without information bottleneck for the total mortality prediction task

Chapter 5

Results and Discussion

In Section 4, several model architectures and various prediction tasks are presented. In this section, the different model architectures are first evaluated and then the performance of different patient cohorts is analyzed.

5.1 Model & task comparison

The one-day, five-day, and total mortality task is compared below to investigate how the different models perform at short, middle, and long-term prediction time horizons. Figure 5.1 shows the performance of all models on the one-day mortality task. To analyze how the performance of a model evolves during the stay of a patient, the x-axis shows the days that have passed since admission. Day 0 is the day of admission and marks the beginning of the stay. For each prediction time point shown on the x-axis, the model uses all the notes available up to and including this day for the prediction. For the last time point on the x-axis "end", the model uses all the notes of the patient. On the y-axis is the AUPRC, which is also utilized as a performance metric to analyze the final result, of all the predictions made at this specific time point. The performance of all models improves over time, with a sharp increase during the first four days and a slight increase after day four from admission. Multi-tasking models exhibit a significant performance increase when the prediction is made one day after admission instead of on the day of admission. The performance difference between models is smaller on the day of admission compared to during the stay. However, the difference decreases at later prediction time points. Except for the day of admission, all multi-tasking models outperform the single target model at all other prediction time points. The multi-tasking version 2 produces the best results, with the ensemble version performing slightly better. In accordance with the hypothesis presented in Section 4.3.1, it can be observed that instructing the model to condition its long-term predictions on its short-term predictions results in the calibration of the latter (see also Section

5.2). This approach also significantly improves the accuracy of the predictions, with a maximum increase of 0.13 in the AUPRC for one-day mortality predictions on day one. The results show that multi-tasking version 1 has slightly worse performance, with no significant difference between the constrained and unconstrained versions. Multi-tasking version 3 achieves worse results compared to the other multi-tasking versions, but still performs better than the single-target version. It thus appears that adding a single logit (i.e., the output logit of shorter-term predictions) to longer-term output logits introduces a weak constraint that produces an insufficient conditioning effect. Although using short-term predictions and the `CLS` token for the conditioning, as in the case with multi-tasking 3, imposes a strong constraint, results in worse performance. The AUPRC for the five-day mortality is depicted in Figure 5.2.

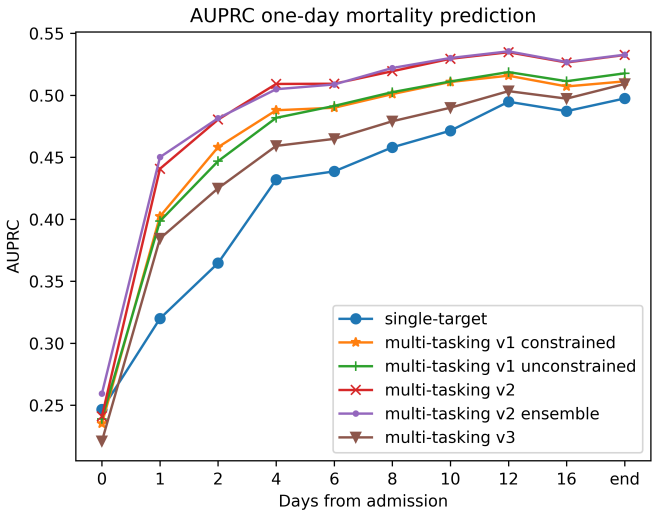

Figure 5.1: Model comparison for the one-day mortality task

Similarly to the one-day mortality task, performance improves over time, with a sharper increase in the first days. The difference in performance is small for all prediction time points. There is no significant difference in the performance between the constrained and unconstrained versions of multi-tasking version 1. Across all the prediction time points, the multi-tasking version 2 ensemble yields the best performance. Figure 5.3 shows the AUPRC for predicting total mortality. The performance of all models increases consistently over time. Although the difference in performance between the models is small, the multi-tasking version 2 ensemble achieves the best results considering all prediction time points.

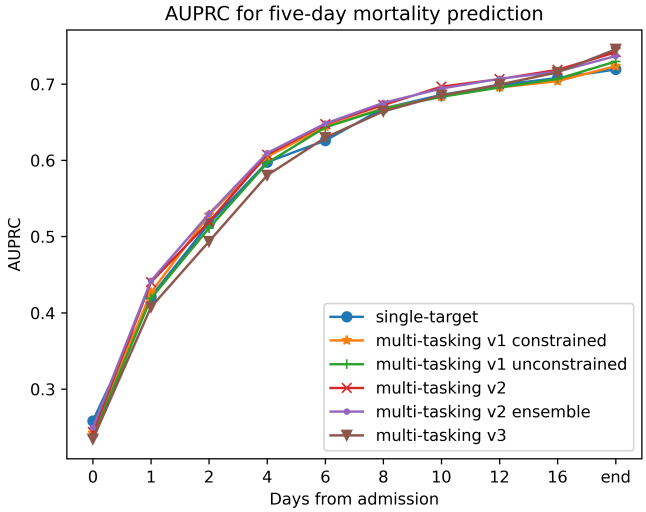

Figure 5.2: Model comparison for the five-day mortality task

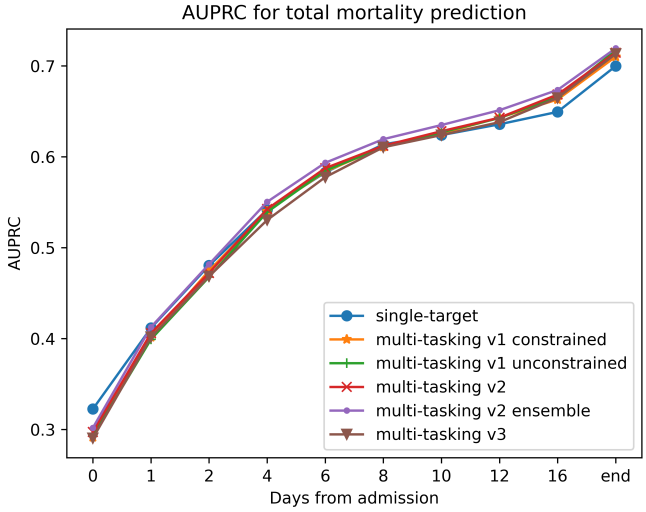

Figure 5.3: Model comparison for the total mortality task

A comparison of the absolute performance across the three different time horizons (Table 5.1) reveals that the short time horizon (one-day mortality) exhibits inferior performance compared to the middle (five-day mortality) and long (total mortality) time horizons. The performance of the middle and long time horizons is comparable, with the AUPRC for the five-day mortality marginally superior for all prediction time points except the initial day. The discrepancy in performance between the short time horizon and the middle/long time horizons is considerably less pronounced on the initial day (max 0.07 AUPRC) than on the final day (max 0.24 AUPRC). Furthermore, the ob-

		Prediction time point				
Model	Task	0	4	8	16	end
single target	one-day mortality	0.25	0.43	0.46	0.49	0.50
	five-day mortality	0.26	0.60	0.67	0.70	0.72
	total mortality	0.32	0.54	0.61	0.65	0.70
multi-tasking v1 const.	one-day mortality	0.24	0.49	0.50	0.51	0.51
	five-day mortality	0.24	0.60	0.67	0.70	0.72
	total mortality	0.29	0.54	0.61	0.66	0.71
multi-tasking v1 unconst.	one-day mortality	0.24	0.48	0.50	0.51	0.52
	five-day mortality	0.24	0.60	0.67	0.70	0.73
	total mortality	0.29	0.54	0.61	0.67	0.72
multi-tasking v2	one-day mortality	0.24	0.51	0.52	0.53	0.53
	five-day mortality	0.24	0.61	0.67	0.72	0.74
	total mortality	0.30	0.54	0.61	0.67	0.71
multi-tasking v2 ensemble	one-day mortality	0.26	0.50	0.52	0.53	0.53
	five-day mortality	0.25	0.62	0.68	0.72	0.74
	total mortality	0.30	0.55	0.62	0.67	0.72
multi-tasking v3	one-day mortality	0.22	0.46	0.48	0.50	0.51
	five-day mortality	0.23	0.58	0.66	0.72	0.75
	total mortality	0.29	0.53	0.61	0.66	0.71

Table 5.1: AUPRC for all models and tasks

served increase in performance over time is less pronounced for the short time horizon (max 0.29 AUPRC) compared to the middle/long time horizons (max 0.52 AUPRC). This observation holds for all models. It can be concluded that the models are more adept at predicting the overall trajectory of a patient's status than at predicting one-day mortality. This is likely due to the fact that it is easier to infer the severity of a patient's status (long-term prognosis) than to estimate its urgency. Furthermore, for more complex tasks, the additional information provided at later prediction does not help the models as much as it does for simpler tasks.

In the context of one-day mortality prediction, all multi-tasking models demonstrate superior AUPRC values compared to the single target model. About the five-day mortality task, only the multi-tasking version 2 and the multi-tasking version 2 ensemble achieve enhanced results across multiple prediction time points in comparison to the single target model. It can be concluded that the model benefits from the additional information provided in the multi-tasking setup for the more complex task of predicting one-day mortality. The performance difference between the constrained and unconstrained multi-tasking version 1, considering all three tasks, is not significant. It can be seen that conditioning long-term predictions on short-term predictions through a

one-dimensional residual connection does not impose a sufficiently strong constraint, as already suspected in Section 4.3.1.
Although the models for multi-tasking versions 2 and 3 have the same number of parameters, multi-tasking version 2 performs better at all tasks. This observation underscores the importance of even minor architectural alterations on the model's efficacy. In this multi-tasking setup, it is crucial to utilize the parameters in a sophisticated way to allow the model to learn a meaningful common feature representation ϕ , which represents the patient's status at a given time in the future. To condition the long-term patient status on the short-term patient status, it is preferable not to compute the subsequent feature representation using the previous and the `CLS` token as input. Instead, it is advisable to utilize the `CLS` token to compute the difference, ϕ', which will be added to the previous feature representation to compute the subsequent one. This results in a residual connection between the feature representations, which may also explain the superior performance of this approach. Following the literature presented in Section 2.2, ensembling is a beneficial approach to enhance the performance of the model. Consequently, the multi-tasking version 2 ensemble model demonstrates optimal performance across all tasks and prediction time points.

5.2 Model calibration

Figure 5.4 displays the ECE for model calibration at short, middle, and long prediction horizons and different prediction time points. The ECE is relatively small for all tasks yet exhibits considerable volatility. For five-day and total mortality, the ECE is initially higher and declines over time. In contrast, for one-day mortality, the ECE exhibits the smallest values and remains relatively constant for all prediction time points. Overall, the ECE for total mortality is greater than the ECE for five-day mortality. It can be concluded that, following

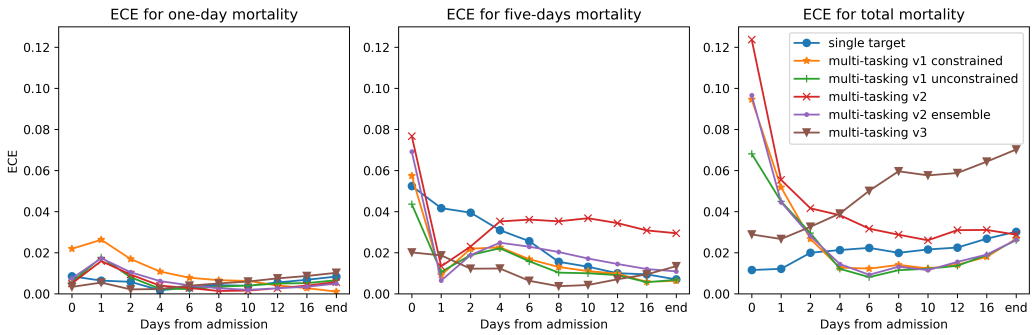

Figure 5.4: Comparison of the ECE of the different models for the one-day, five-day and total mortality

the theory presented in section 2.2, the training regimen outlined in section 4.6.3 has resulted in the calibration of all models for all three-time horizons, thereby rendering them trustworthy. However, the calibration is slightly worse for the longer time horizons, particularly at the beginning of the patient's stay. Despite the models' performance improving significantly over time for middle and long prediction horizons, they cannot make reliable predictions when only limited data is available. This is likely because a long-term prognosis is less reliable with less data.

5.3 Temporal performance

As discussed in the previous section, all models perform better when predictions are made later in a patient's stay. On the one hand, the closer the prediction time point is to the patient's discharge or death, the easier the task becomes. In addition, as more patients are discharged over time, all patient information is used for the prediction for the corresponding patients. This also affects the AUPRC for all subsequent prediction time points, as no new predictions are made for terminated patients. For example, if a patient's stay ends on day six, the prediction made for that day is also factored into the calculation of the AUPRC for day eight and all subsequent days. This raises the question of how the models' performance improvement over time is related to the duration of the patient's stay. Therefore, patients are divided into three groups based on the duration of their stay: up to 4 days (35%), 5 to 8 days (30%), and longer than eight days (35%). The performance metric is then calculated for each group individually.
Figure 5.5 displays the AUPRC for predicting total mortality in three different patient cohorts. For the cohort with short stays, the AUPRC for all models is around 0.5 on the day of admission, while it is around 0.2 for the other two cohorts. This highlights that models perform better when the end of the patient's stay is near, even with limited data available for prediction. One possible reason is that the patient's status becomes clearer as their stay nears its end. The performance increase for short and medium stays is sharper at the beginning compared to those with longer stays. When the end of the stay lies still in the far future, the improvement that comes with data from each additional day of stay is less pronounced. However, even patients with longer stays show an increase in performance at the beginning of their stay, although the end of their stay is still a few days away. The final AUPRC for the medium and long stay cohorts is approximately 0.7, while for the short stay cohort, it is around 0.75. Consequently, the architecture demonstrates comparable predictive performance at the end of the stay, regardless of the length of the stay. Furthermore, the improvement in AUPRC for the last five days is comparable to approximately 0.2, regardless of the total duration of the stay. It can be

concluded that all models are able to achieve similar good results for short and long input sequences using the temporal Transformer architecture. In all cases, the predictive performance of all models improves as additional information becomes available over time. Furthermore, the farther away the end of the stay is, the more challenging it is to predict the outcome.

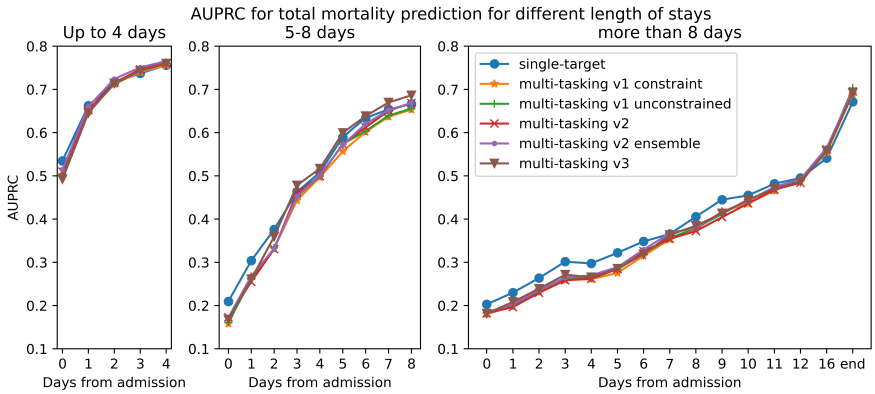

Figure 5.5: AUPRC for total mortality prediction for stays up to 4 days, 5 to 8 day and longer than 8 days

To analyze the temporal performance of a model, one method is to determine the day when the prediction reaches its final value. The multi-tasking version 2 ensemble model was used for this analysis, which yielded the best results. Total mortality was analyzed, as it is the only task with constant labels throughout the patient's stay. The histograms for the final prediction flip days are computed separately for each stay duration cohort (up to 4 days, 5 to 8 days, and longer than 8 days) for correct and incorrect predictions. Due to the severe class imbalance, a distinction is made between the patients who survive (Figure 5.6) and those who die (Figure 5.7). The left column represents false positives and the right column represents true negatives for Figure 5.6. The left column represents false negatives and the right column represents true positives for Figure 5.7. For stays up to four days, the model does not change the initial prediction made on the admission day at a later time point (the last/final label flip occurs on day 0) for the majority of patients, regardless of whether the prediction is correct or false, and whether the patient survives or expires. Furthermore, the number of false predictions that remain unchanged after the admission day is significantly higher for stays up to 4 days compared to stays between 5 and 8 days for survivors and expiring patients. For the two cohorts with longer stays, there are more correct prediction flips after the day of admission than on the day of admission for expiring patients. These flips are irregularly distributed over the days. It can be concluded that the model uses the additional information provided during a longer stay of a patient to change the prediction to the correct value. However, additional information used by the model can also lead to a false change in prediction at a later time

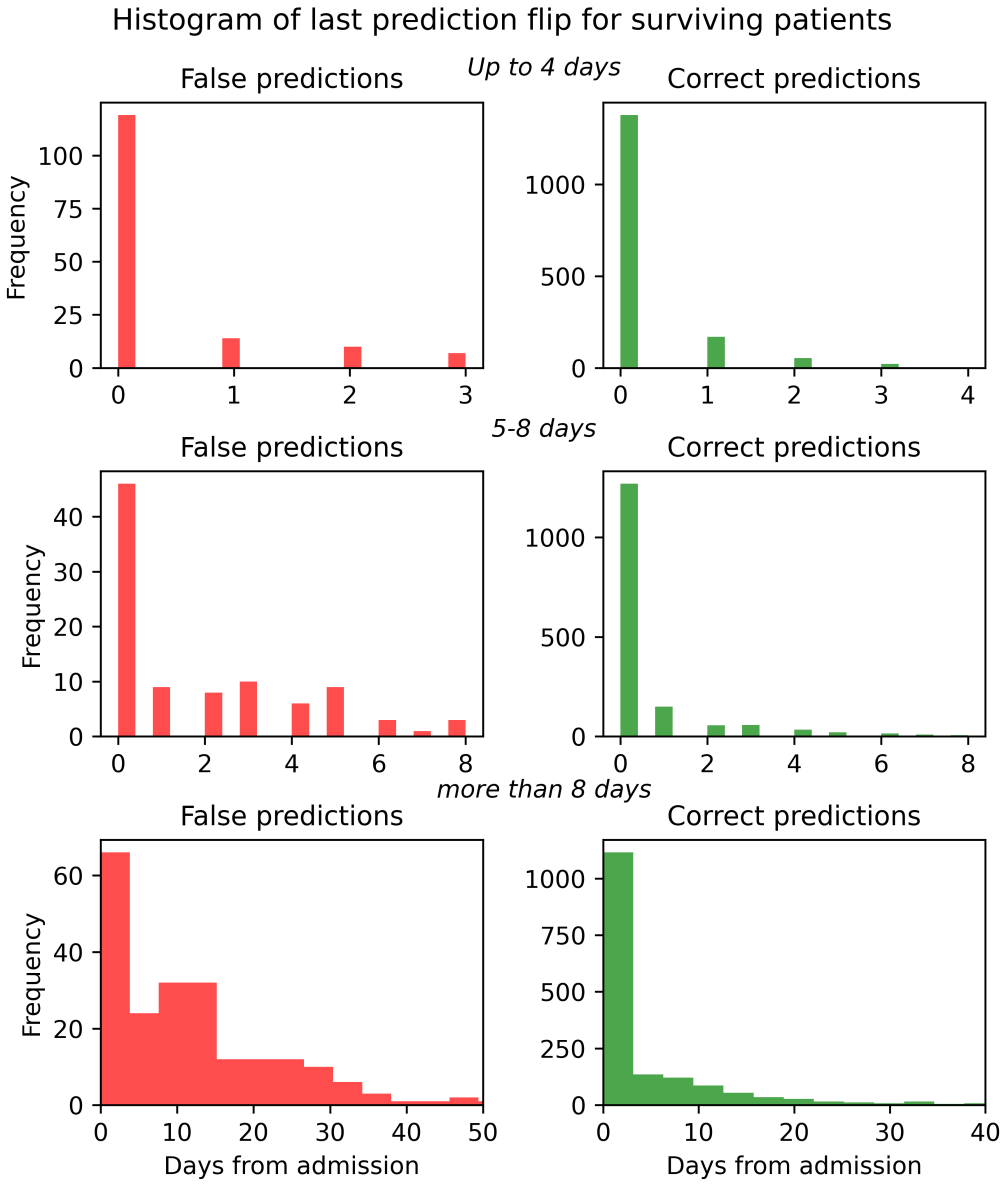

Figure 5.6: Histogram of the last prediction flip for surviving patients for the total mortality task using the multi-tasking version 2 ensemble model

point. Nevertheless, most prediction updates correct the initial wrong prediction. The analysis also demonstrates that the designed architecture enables the model to use clinical notes' temporal sequence to adjust the prediction accurately.

Regarding absolute numbers, the most significant changes after the admission day are predictions changing to true negatives. This behavior is also evident in Figure 5.8. For all three cohorts, the number of patients predicted to survive

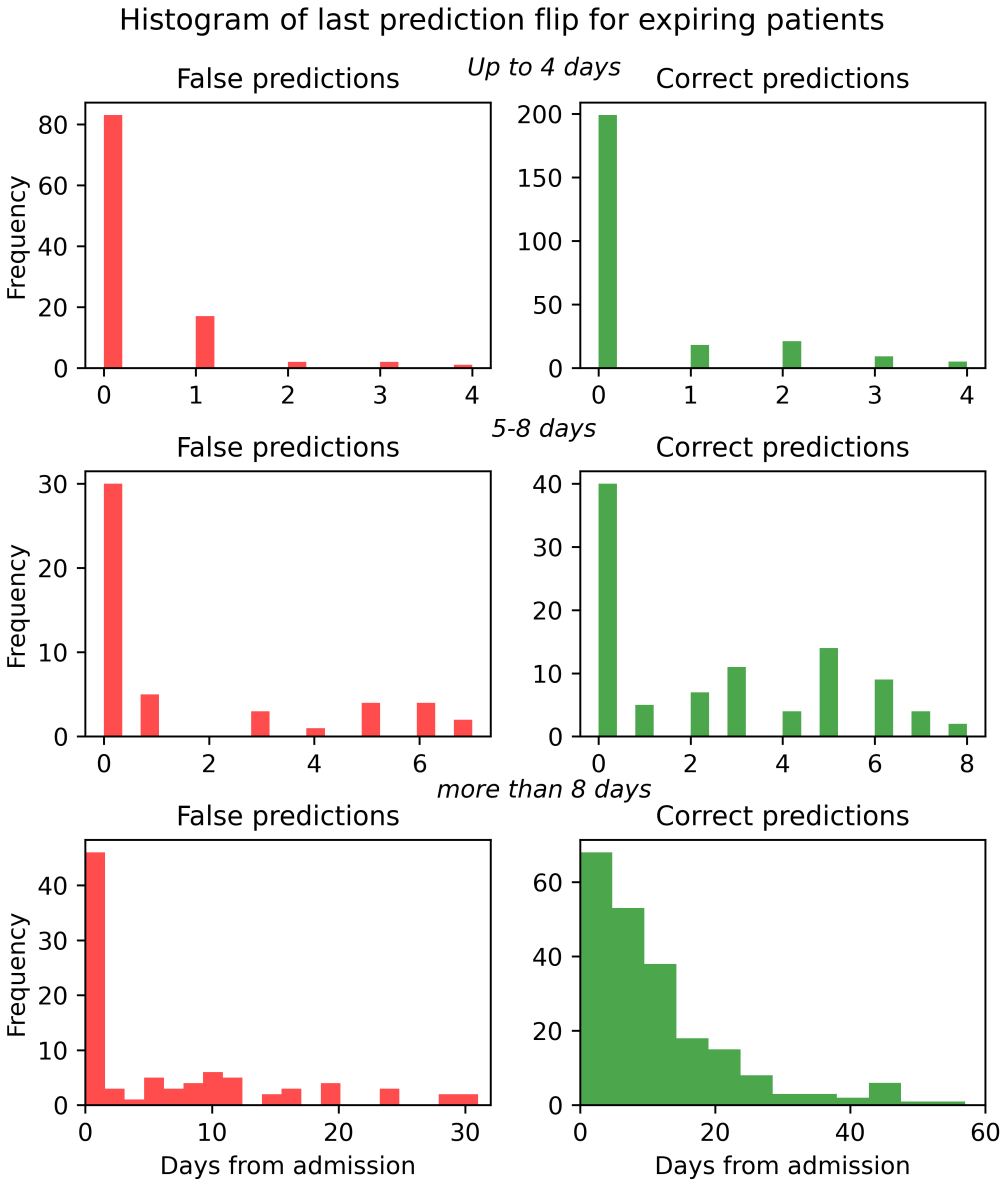

Figure 5.7: Histogram of the last prediction flip for expiring patients for the total mortality task using the multi-tasking version 2 ensemble model

increases over time but does not reach the ground truth ratio. The increase in true negatives is reducing the number of false positives, resulting in a sharp increase in precision at the beginning, but it is still below the recall. The model exhibits bias towards false positives, predicting that patients will die when they will actually survive, particularly at the beginning of longer stays. Although recall increases for longer stays, it has a lower initial value, causing the model to struggle more to select dying patients at the beginning. In con-

clusion, the initial predictions of the model are biased towards predicting that patients will not survive.

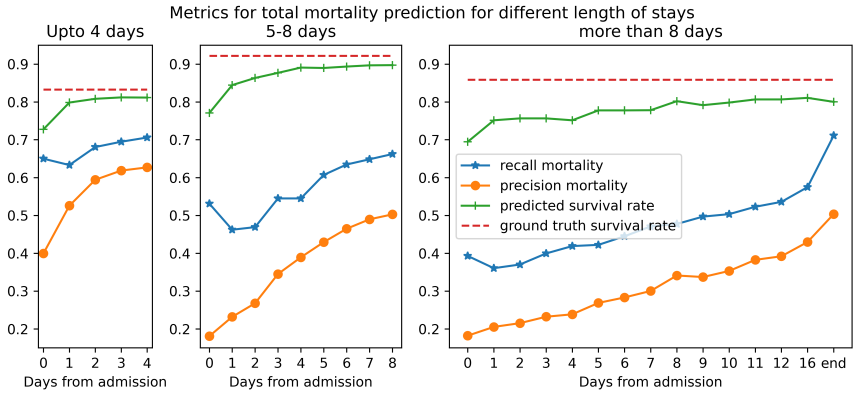

Figure 5.8: Metrics for total mortality prediction for different length of stays using the multi-tasking version 2 ensemble model

5.4 Performance comparison with the state-of-the-art

In the preceding section, the various versions of the model and the tasks were subjected to a comparative analysis. This section will present a comparison of the performance of the models with the state of the art reported in the literature. The models developed in this thesis can predict mortality at multiple rolling time horizons and arbitrary time points throughout the patient's stay. However, as previously stated in Section 2.3, the majority of the results reported in the literature, which are based solely on the clinical notes from the MIMIC-III dataset, refer only to overall mortality prediction, either in-hospital or post-discharge, evaluated at the end of the stay or a few selected time points shortly after admission. Consequently, the comparison is confined to the prediction of total mortality.

Table 5.2 compares the performance of the designed models with that of state-of-the-art models trained to predict in-hospital mortality using only the clinical notes from the same cohort in the MIMIC-III dataset at the end of the patient's stay. As the AUROC metric is the most commonly used in the literature, it must also be used for comparison. All models perform only slightly worse than the state-of-the-art, while the single target model achieves slightly higher AUROC than all the other models, although all design choices and hyperparameter selection were made to maximize the AUPRC, not the AUROC.

Table 5.3 compares the performance of the models developed in this study

Paper	Model	Mortality @ end AUROC
Yuqi et al. [48]	multi-tasking multi-level CNN	**0.9457**
Sushil et al. [50]	Stacked denoising autoencoder	0.9383
This work	single target	0.9046
This work	multi-tasking v1 const.	0.8995
This work	multi-tasking v1 unconst.	0.8991
This work	multi-tasking v2	0.8973
This work	multi-tasking v2 ensemble	0.9025
This work	multi-tasking v3	0.8975

Table 5.2: Performance of state-of-the-art models predicting mortality at the end of the stay

		Mortality @ 24h		Mortality @ 48h	
Paper	Model	AUROC	AUPRC	AUROC	AUPRC
Hashir et al. [22]	hierarchical CNN-RNN	**0.8675**	**0.3327**	**0.8887**	0.3853
This work	single target	0.7380	0.3225	0.7883	0.4117
This work	multi-tasking v1 const.	0.7130	0.2914	0.7709	0.4015
This work	multi-tasking v1 unconst.	0.7122	0.2928	0.7695	0.3992
This work	multi-tasking v2	0.7177	0.2972	0.7703	0.4060
This work	multi-tasking v2 ensemble	0.7211	0.3018	0.7766	**0.4127**
This work	multi-tasking v3	0.7112	0.2904	0.7732	0.4017

Table 5.3: Performance of state-of-the-art models predicting mortality at 24h & 48h

with that of the hierarchical CNN-RNN model presented by Hashir et al. [22] in predicting mortality 24 and 48 hours after admission. With respect to both tasks, the AUROC of the CNN-RNN model is higher. While the AUPRC of the CNN-RNN is also higher for the 24-hour prediction, the AUPRC for the 48-hour prediction is higher for all models, with the multi-tasking v2 ensemble achieving the best performance. This indicates that the designed architecture is well suited for online prediction, which aligns with the findings in section 5.3 and with the goal of designing an architecture for online multiple-horizon mortality prediction. The higher AUPRC despite the lower AUROC for the prediction 48 hours after admission may indicate that a similar behavior exists for all subsequent prediction time points. This would relativize the poorer performance according to Table 5.2. It is essential to consider that transformers lack certain inductive biases inherent in CNNs, which can lead to a lack of generalization when trained with insufficient data [15]. This may also ex-

plain the occasionally inferior performance of the proposed transformer-based architecture, given that the MIMIC-III is a comparable modest dataset. In light of the aforementioned considerations, it can be posited that the designed architecture is well suited to clinical decision support.

5.5 Clinical use cases

In this section, the multi-tasking version 2 ensemble is analyzed in detail to identify specific patient groups in which the model performs well and those in which it does not. This will help identify possible cohorts for clinical applications and those in which the model should not be used. Performance is investigated first for different age groups, then for different ICD-9 codes, and finally for a combination.

5.5.1 Age performance

Six age bins were defined to compare the performance of the models depending on the age of the admitted patients.

1. $< 40(12\%)$
2. $40 - 50(11\%)$
3. $50 - 60(17\%)$
4. $60 - 70(19\%)$
5. $70 - 80(20\%)$
6. $> 80(18\%)$

The bins were defined to contain a similar number of persons in each bin, despite the presence of more elderly patients. Figure 5.9 shows the AUPRC for one-day and total mortality in each age group and a baseline using all patients. The performance difference between age groups is smaller for the total mortality task, and all age groups are closer to the baseline compared to the one-day mortality task. The AUPRC for one-day mortality is more volatile over time, with greater differences compared to baseline. Taking into account all prediction time points, the cohort over 80 yields the worst performance. In contrast, the cohort under 40 has the best performance and is consistently significant above the baseline for the one-day mortality task. However, for the total mortality task, the cohort over 80 yields the best results, while the four youngest cohorts are mainly below the baseline. It is essential to exercise

caution when using one-day mortality as a predictor, as it may be more difficult to accurately predict depending on the patient's age and the timing of the prediction. However, total mortality predictions tend to be more stable. In clinical practice, predicting one-day mortality may be useful for patients under 40 years of age.

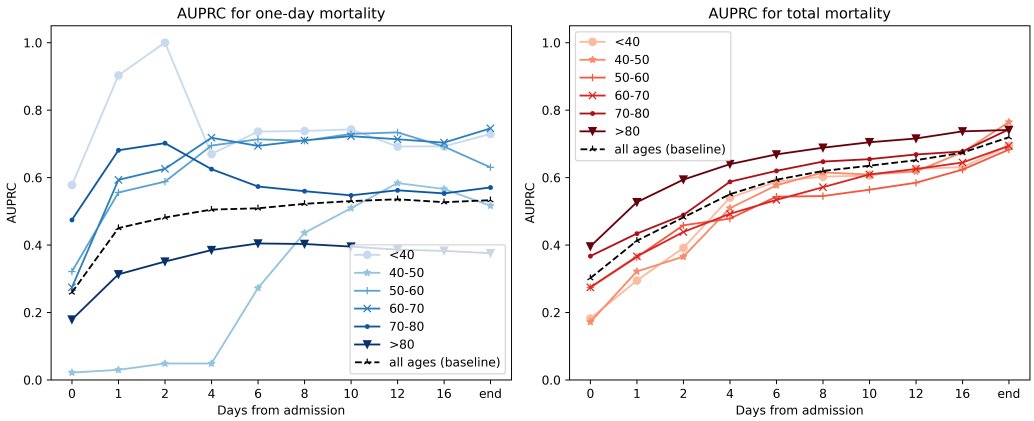

Figure 5.9: Age-specific AUPRC for one-day and total mortality

5.5.2 ICD-9 performance

Another way to create patient cohorts is by using ICD-9 codes. The MIMIC-III database provides a sequence of ICD-9 codes for each admitted patient, the first code indicating the reason for admission to the ICU. These codes were used to establish the patient cohorts. The analysis compared the following six most common codes:

- *Infectious and parasitic diseases* (7%)
- *Neoplasms* (7.8%)
- *Diseases of the circulatory system* (40%)
- *Diseases of the respiratory system* (6,9%)
- *Diseases of the digestive system* (9%)
- *Injury and poisoning* (18%)

Figure 5.10 shows the AUPRC for one-day, total mortality for each ICD9 code, and a baseline using all patients. Regarding one-day mortality, only the category of *infectious and parasitic diseases* is significantly above the baseline for all prediction time points. In contrast, the category of *diseases of the*

circulatory system remains consistently below the baseline. The performance of the other cohorts is subject to fluctuations. In terms of total mortality, the AUPRC of *injury and poisoning* and *respiratory diseases* are around the baseline, while rates for *diseases of the circulatory system* are slightly above the baseline, for *infectious and parasitic diseases* significantly above the baseline, and for *digestive system diseases* and *neoplasms* are below the baseline. Given that the AUPRC for *infectious and parasitic diseases* is significantly above the baseline for both tasks, this would be an appropriate clinical use case of the model.

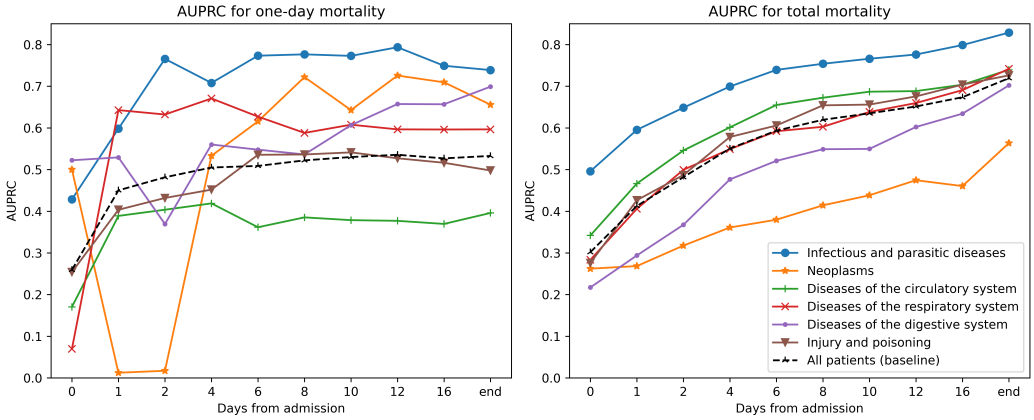

Figure 5.10: ICD-9 specific AUPRC for one-day and total mortality

5.5.3 Age & ICD-9 performance

To differentiate the analysis of the performance of the model, the AUPRC for *infectious and parasitic diseases*, *neoplasms*, and *diseases of the circulatory system* for the six age cohorts introduced in section 5.5.1 is calculated. This is done independently for the one-day mortality and total mortality tasks.
By conducting this two-dimensional investigation, the resulting patient cohorts become smaller. Thus, prediction time points for the one-day mortality task with no positive labels exist, making the AUPRC an unsuitable metric. Instead, each cohort's accuracy in combination with the corresponding mortality rate is computed to analyze the one-day mortality task (see Figure 5.11). The model predicts a 100% accuracy for patients under 40 years of age for *infectious and parasitic diseases* and *neoplasms* at every prediction time point, despite changes in mortality over time. For *neoplasms*, mortalities for all age groups change rarely and accuracies remain consistently high. The mortality rate for *diseases of the circulatory system* consistently increases in the two oldest age cohorts (70-80, >80), while the corresponding accuracy decreases. For the other age cohorts, mortality remains relatively constant and accuracy

remains consistently high. It can be concluded that the model has difficulty detecting patients who expire within one day, if in the respective cohort, patients expire consistently. However, the model achieves superior results for patients under 40 with *infectious and parasitic diseases* and *neoplasms*. This cohorts would be a suitable clinical use case for the model to predict mortality in one day.

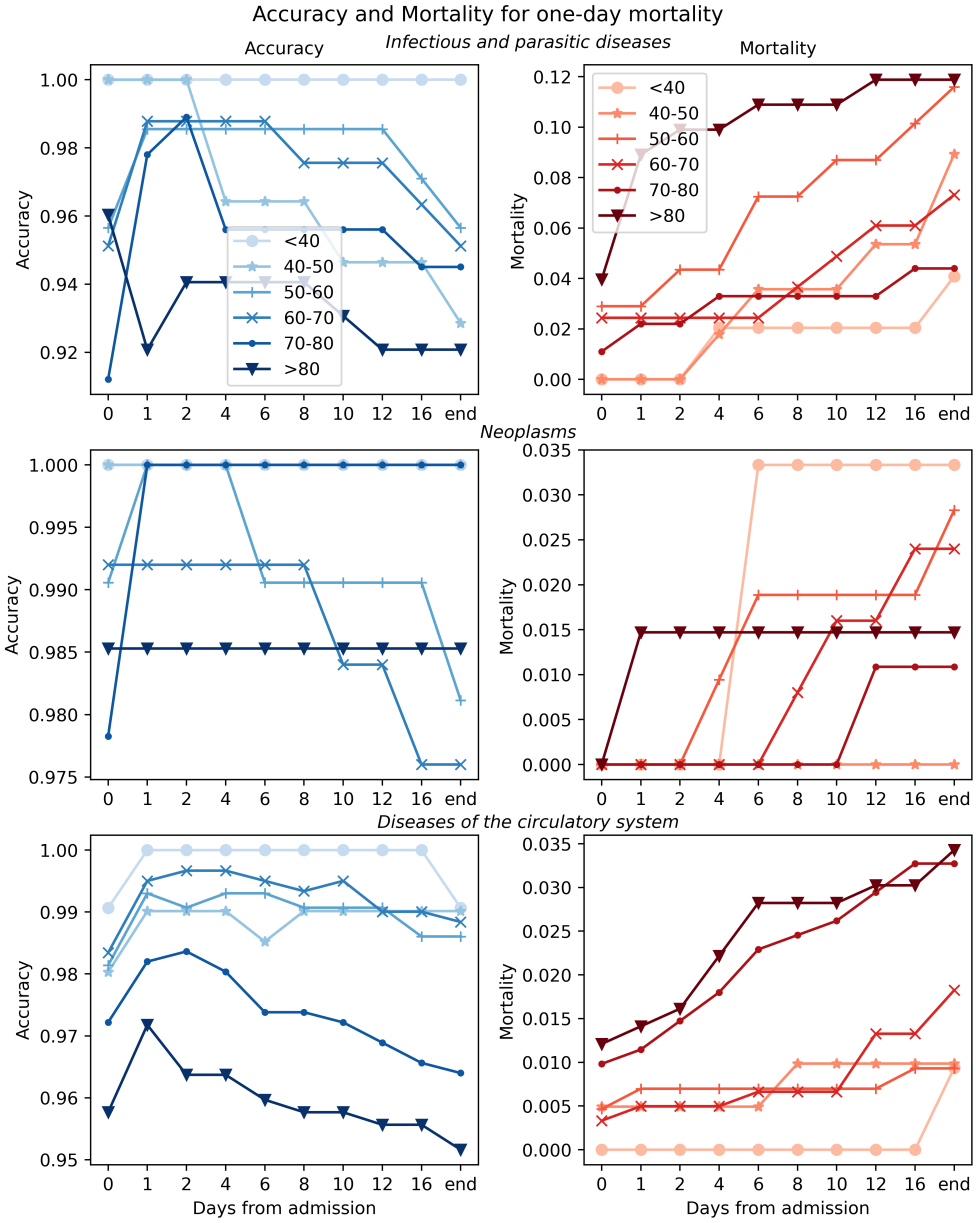

Figure 5.11: Accuracy and mortality for one-day mortality

In order to analyze the total mortality, the AUPRC is used. To investigate

whether the mortality rate of specific cohorts correlates with the AUPRC, it is shown to the right of each AUPRC curve (Figures 5.12, 5.13, 5.14). Compared to the other two ICD-9 groups, there is a greater variance in the mortality rate between different age groups for *infectious and parasitic diseases*. However, the performance at later prediction time points is fairly similar. A comparison of mortality rates and AUPRC in the three ICD-9 groups reveals no apparent correlation between the order of mortality rates and the order of the AUPRC for different age groups. For example, people between the ages of 70 and 80 have the second highest mortality rate for *infectious and parasitic diseases* and *diseases of the circulatory system*, but exhibit the least favorable performance for *infectious and parasitic diseases* and the second-best performance for *diseases of the circulatory system*. It can be concluded that the performance of the model is not solely dependent on the mortality rate of a specific age group. Rather, the model yields better results for different cohorts.
The model demonstrates superior performance for patients over the age of 80 with *infections and parasitic diseases*, as well as for patients under the age of 40 with *neoplasms*. The performance of the model is particularly robust from the time of admission, making this cohort an excellent clinical use throughout the entire patient's stay. Although the model performs poorly on the first two days for patients under 40 years of age with *diseases of the circulatory system*, the performance for the following days is notably high, reaching a consistently high level from day six onward, with an AUPRC of approximately 0.9. This cohort would be a suitable use case for the later prediction time points.

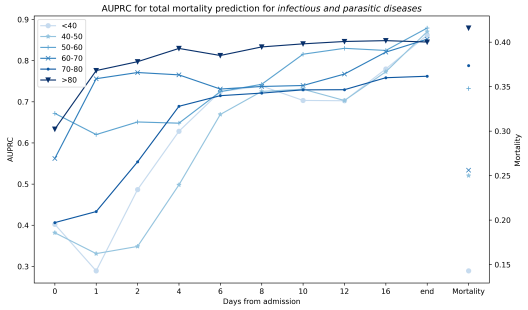

Figure 5.12: Age-specific AUPRC for total mortality for *infectious and parasitic diseases*

Although the AUPRC is similar across different age cohorts at later prediction time points, there are significant differences in the AUPRC for certain age groups at earlier prediction time points. For example, for *infectious and parasitic diseases* on day one, the AUPRC is 0.28 for patients under 40 years of age, while 0.77 for patients over 80 years of age. The confusion matrices for these two predictions are shown in Figure 5.15 to investigate this model behavior further. For both age groups, most of the predicted negatives are negative. The difference lies in the number of false positives, resulting in a

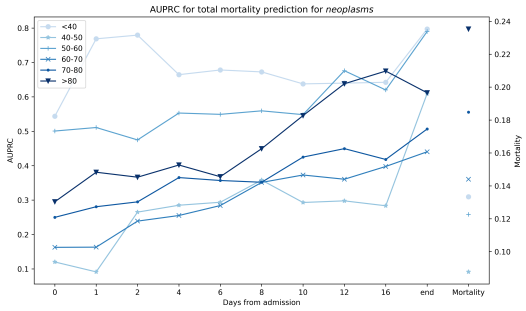

Figure 5.13: Age-specific AUPRC for total mortality for *neoplasms*

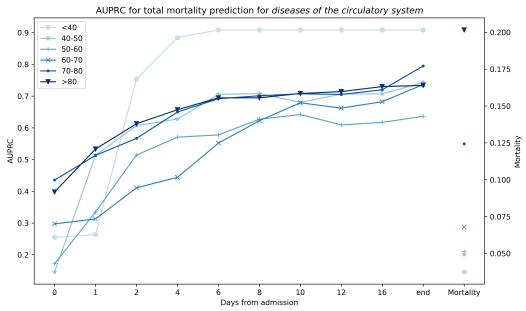

Figure 5.14: Age-specific AUPRC for total mortality for *diseases of the circulatory system*

low precision value. On the day of admission, the precision value for patients over 80 was 0.58, while for patients under 40, it was 0.26. Over time, the number of false positives decreased for patients below 40 until it reached only two on the last day (right side of Figure 5.15). This resulted in an increase in precision and, consequently, an elevated AUPRC. The model demonstrates a

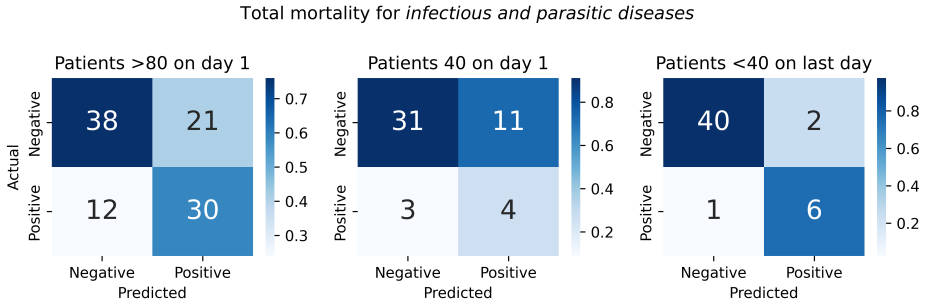

Figure 5.15: Confusion matrices for *infectious and parasitic diseases*

comparable pattern of behavior for neoplasms on the day of admission. The AUPRC for patients under 40 is 0.76, while for patients between the ages of 40 and 50, it is 0.09. For patients below the age of 40, 66% of the predicted positives are true positives, while for patients between 40 and 50 years of age,

the proportion of true positives is 20% on the first day but 42% on the last day (Figure 5.16). It can be concluded that the discrepancy in performance

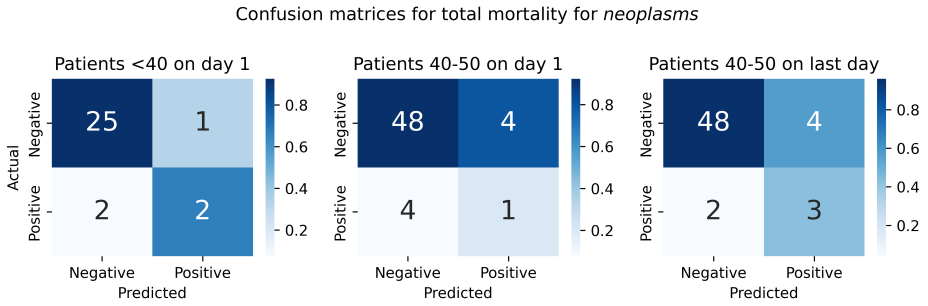

Figure 5.16: Confusion matrices for *neoplasms*

between different age groups at the beginning of the stays is attributable to the high false positive rate for the cohort with poorer performance. The model is biased towards predicting that more patients will expire than is the case. This result is consistent with the findings presented in Section 5.3, which indicates that the initial lower performance during stays is also due to the model's tendency to predict patients as likely to expire, even if they ultimately survive.

Chapter 6

Conclusion and Future Work

This work presents the development of machine learning models and a training methodology for online multi-horizon clinical decision support. The models utilize only the unstructured textual content and corresponding temporal information from EHRs. The results demonstrate that there is untapped potential for achieving this goal.

The architecture is designed based on hierarchical transformers that use the late fusion approach to integrate temporal information after computing feature embeddings for each note. This approach leverages the pre-trained Clinical Longformer for the notes, significantly reducing computational costs during training. The model is trained using appropriate masking to only use notes until a given time point. This technique enables transformer-based models to operate effectively in an online regime, allowing for predictions to be made at any point during an ICU stay based solely on the available information up to that moment.

Patient mortality is the chosen prediction task. Additionally, the task has been extended to include predictions at multiple rolling time horizons, such as one-day, three-day, five-day, ten-day and total mortality. This adds practical value to estimating mortality risk by specifying the time frame and providing a measure of urgency, thus offering more actionable input to clinicians. Various model architectures were designed to use a multi-task objective to jointly predict mortality for all time horizons in one model instead of separate models for each time horizon. All models perform worse for short-term targets compared to middle and long-term targets, which corresponds to the natural difficulty of the respective tasks. The results show that multitasking significantly benefits the more complex short-term prediction time horizon. The most successful outcomes were achieved by leveraging the relationship between long-term and short-term tasks. This was accomplished by training the model to learn a shared patient representation for all time horizons and making the longer-term patient status conditional on the shorter-term patient status. It was observed that the farther away the end of the stay is, the harder

it is to predict the outcome. However, the model can utilize additional information provided during a patient's stay and correct its predictions over time. As a result, the performance using all notes is similar for different stay lengths. Therefore, the models do indeed make online mortality predictions.
Finally, the thesis examines the performance of the model for different groups of patients. The results show that the model performs better for patients under the age of 40 with *infectious and parasitic diseases* and *neoplasms* for short-term predictions, and for patients over 80 years of age with *infections and parasitic diseases*. Additionally, the model performs well for patients under 40 years of age with *diseases of the circulatory system* and *neoplasms* for long-term predictions. These cohorts represent potential clinical use cases for the model. This thesis makes significant contributions to developing online clinical decision support models that take textual and temporal input. These potential clinical use cases demonstrate the practical applications of such models. These models have significant potential as a valuable tool for clinicians in clinical use cases, such as prioritizing the allocation of scarce resources. The optimization of scarce resources can enhance patient care without increasing costs, ultimately leading to improved health care for all.
As a suggestion for future work, the model could be tested on a dataset from a different domain. This is because the designed framework can be universally applied to all online predictions using a growing observation window with multiple time horizons for irregularly sampled data points, regardless of the modality. It is important to investigate whether the design choices made for the MIMIC-III dataset are also applicable to datasets from different domains. Furthermore, the effect of updating the weights of the Clinical Longformer during training on task performance would be worth investigating. The Clinical Longformer could be pre-trained using the Condenser architecture, which has demonstrated significant improvements over standard language models in various text retrieval and similarity tasks [17]. Additionally, the model could be trained on a different medical dataset with the same task to analyze if the poor performance of the model during the initial days of a patient's stay is due to the limitations of the MIMIC-III dataset, which may not be large enough or comprehensive enough due to the high number of missing notes. This could be particularly evident in the case of transformer-based models as developed in this thesis.

Bibliography

[1] Aczon, M., Ledbetter, D., Ho, L., Gunny, A., Flynn, A., Williams, J., and Wetzel, R. Dynamic mortality risk predictions in pediatric critical care using recurrent neural networks. *arXiv preprint arXiv:1701.06675* (2017).

[2] Alsentzer, E., Murphy, J. R., Boag, W., Weng, W.-H., Jin, D., Naumann, T., and McDermott, M. Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323* (2019).

[3] Aras, H., Türker, R., Geiss, D., Milbradt, M., and Sack, H. Get your hands dirty: Evaluating word2vec models for patent data. In *SEMANTiCS (Posters & Demos)* (2018).

[4] Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., and Schmid, C. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 6836–6846.

[5] Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[6] Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* (2020).

[7] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *European conference on computer vision* (2020), Springer, pp. 213–229.

[8] Cartwright, D. J. Icd-9-cm to icd-10-cm codes: what? why? how?, 2013.

[9] Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y. Recurrent neural networks for multivariate time series with missing values. *Scientific reports 8*, 1 (2018), 6085.

[10] Crawshaw, M. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796* (2020).

[11] Dai, J., He, K., and Sun, J. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 3150–3158.

[12] Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems 35* (2022), 16344–16359.

[13] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[14] Ding, J., Ma, S., Dong, L., Zhang, X., Huang, S., Wang, W., Zheng, N., and Wei, F. Longnet: Scaling transformers to 1,000,000,000 tokens, 2023.

[15] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[16] Duan, J., Jiang, H., and Yu, Y. Mhlat: Multi-hop label-wise attention model for automatic icd coding. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2023), IEEE, pp. 1–5.

[17] Gao, L., and Callan, J. Condenser: a pre-training architecture for dense retrieval. *arXiv preprint arXiv:2104.08253* (2021).

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

[19] Grnarova, P., Schmidt, F., Hyland, S. L., and Eickhoff, C. Neural document embeddings for intensive care patient mortality prediction. *arXiv preprint arXiv:1612.00467* (2016).

[20] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks, 2017.

[21] Harutyunyan, H., Khachatrian, H., Kale, D. C., Ver Steeg, G., and Galstyan, A. Multitask learning and benchmarking with clinical time series data. *Scientific data 6*, 1 (2019), 96.

[22] Hashir, M., and Sawhney, R. Towards unstructured mortality prediction with free-text clinical notes. *Journal of biomedical informatics 108* (2020), 103489.

[23] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[24] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors, 2012.

[25] Huang, K., Altosaar, J., and Ranganath, R. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342* (2019).

[26] Johnson, A. E., Pollard, T. J., and Mark, R. G. Reproducibility in critical care: a mortality prediction case study. In *Machine learning for healthcare conference* (2017), PMLR, pp. 361–376.

[27] Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., and Mark, R. G. Mimic-iii, a freely accessible critical care database. *Scientific data 3*, 1 (2016), 1–9.

[28] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization, 2017.

[29] Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles, 2017.

[30] Lehman, E., and Johnson, A. Clinical-t5: Large language models built using mimic clinical text. *PhysioNet* (2023).

[31] Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems 32* (2019).

[32] Li, Y., Wehbe, R. M., Ahmad, F. S., Wang, H., and Luo, Y. Clinical-longformer and clinical-bigbird: Transformers for long clinical sequences. *arXiv preprint arXiv:2201.11838* (2022).

[33] Lin, T., Wang, Y., Liu, X., and Qiu, X. A survey of transformers. *AI Open* (2022).

[34] Liu, L., Perez-Concha, O., Nguyen, A., Bennett, V., and Jorm, L. Hierarchical label-wise attention transformer model for explainable icd coding. *Journal of biomedical informatics 133* (2022), 104161.

[35] Lyu, X., Hueser, M., Hyland, S. L., Zerveas, G., and Raetsch, G. Improving clinical predictions through unsupervised time series representation learning, 2018.

[36] Mao, C., Yao, L., and Luo, Y. Aki-bert: a pre-trained clinical language model for early prediction of acute kidney injury. *arXiv preprint arXiv:2205.03695* (2022).

[37] Meyer, A., Zverinski, D., Pfahringer, B., Kempfert, J., Kuehne, T., Sündermann, S. H., Stamm, C., Hofmann, T., Falk, V., and Eickhoff, C. Machine learning for real-time prediction of complications in critical care: a retrospective study. *The Lancet Respiratory Medicine 6*, 12 (2018), 905–914.

[38] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems 26* (2013).

[39] Mucsányi, B., Kirchhof, M., Nguyen, E., Rubinstein, A., and Oh, S. J. Trustworthy machine learning, 2023.

[40] Naeini, M. P., Cooper, G., and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence* (2015), vol. 29.

[41] Neimark, D., Bar, O., Zohar, M., and Asselmann, D. Video transformer network. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 3163–3172.

[42] Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. Image transformer. In *International conference on machine learning* (2018), PMLR, pp. 4055–4064.

[43] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library, 2019.

[44] Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), pp. 1532–1543.

[45] Popel, M., and Bojar, O. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics 110*, 1 (Apr. 2018), 43–70.

[46] Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences 63*, 10 (2020), 1872–1897.

[47] Rajkomar, A., Oren, E., Chen, K., Dai, A. M., Hajaj, N., Hardt, M., Liu, P. J., Liu, X., Marcus, J., Sun, M., et al. Scalable and accurate deep learning with electronic health records. *NPJ digital medicine 1*, 1 (2018), 1–10.

[48] Si, Y., and Roberts, K. Deep patient representation of clinical notes via multi-task learning for mortality prediction. *AMIA Summits on Translational Science Proceedings 2019* (2019), 779.

[49] Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489* (2017).

[50] Sushil, M., Šuster, S., Luyckx, K., and Daelemans, W. Patient representation learning and interpretable evaluation using clinical notes. *Journal of biomedical informatics 84* (2018), 103–113.

[51] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023.

[52] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems 30* (2017).

[53] Wang, H., Gao, C., Dantona, C., Hull, B., and Sun, J. Drg-llama: tuning llama model to predict diagnosis-related group for hospitalized patients. *npj Digital Medicine 7*, 1 (2024), 16.

[54] Wang, X., Luo, J., Wang, J., Yin, Z., Cui, S., Zhong, Y., Wang, Y., and Ma, F. Hierarchical pretraining on multimodal electronic health records. *arXiv preprint arXiv:2310.07871* (2023).

[55] Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., and Sun, L. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125* (2022).

[56] Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems 34* (2021), 22419–22430.

[57] Xiong, W., Oğuz, B., Gupta, A., Chen, X., Liskovich, D., Levy, O., Yih, W.-t., and Mehdad, Y. Simple local attentions remain competitive for long-context tasks. *arXiv preprint arXiv:2112.07210* (2021).

[58] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems 32* (2019).

[59] Zerveas, G., Schmidt, F., and Eickhoff, C. Neural nlp methods for online clinical decision support.

[60] Zhang, Y., Chen, Q., Yang, Z., Lin, H., and Lu, Z. Biowordvec, improving biomedical word embeddings with subword information and mesh. *Scientific data 6*, 1 (2019), 52.

[61] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence* (2021), vol. 35, pp. 11106–11115.

[62] Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning* (2022), PMLR, pp. 27268–27286.

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum Unterschrift