UNIVERSITY OF CAMBRIDGE

Representation Learning for Patients in the Intensive Care Unit

Emma Rocheteau

Churchill College

This dissertation is submitted in December 2022 for the degree of *Doctor of Philosophy* in Computer Science and Technology

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. It is not substantially the same as any that I have submitted, or am concurrently submitting, for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my dissertation has already been submitted, or is being concurrently submitted, for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. This dissertation does not exceed the prescribed limit of 60 000 words.

Emma Rocheteau
December, 2022

Abstract

Representation Learning for Patients in the Intensive Care Unit

Emma Rocheteau

The past decade has seen accelerating interest in Artificial Intelligence (AI) in Healthcare. Data is now being generated in the form of Electronic Health Records at a scale previously unimaginable. Not only does this create opportunities for the application of AI, but it also drives innovation in the machine learning sphere. This is because health problems can present unique challenges not encountered in other domains, and clinical decision making itself can provide ingenious approaches inspiring new learning methods.

The work in this thesis sits in the space between medicine and machine learning and has contributions to both domains. The broad theme is *representation learning for the patient in intensive care*. The eventual aim is to promote better outcomes for patients and improve the efficiency of the healthcare system. I focus in particular on predicting patient deaths and estimated dates of discharge, because they lie at the heart of the resource allocation problem in hospitals. The efficient management of hospital beds is more important than ever in the wake of staff retention crises, post-pandemic budgets and ageing populations.

Specifically, in Chapter 3, I use clinical knowledge of the medical *time series* (namely that they are periodic signals with particular systematic biases) to improve upon the state-of-the-art in length of stay prediction (with additional investigations into mortality prediction). In Chapter 4, I am again inspired by knowledge of the clinical decision making process to propose a method using graph neural networks to leverage data from similar patients when predicting outcomes, providing important context for the predictions and interpretability opportunities. In Chapter 5, I delve further into the representation space, exploring the effect of auxiliary tasks on the performance of patient outcome models for mechanically ventilated patients. I then cluster the learned representations with the aim of discovering hidden patient phenotypes. The vision is ultimately to create robust and holistic patient representations which are suitable for deployment in the real-world.

Acknowledgements

Undertaking this PhD has been the most challenging endeavour of my academic life. I feel privileged and emotional to be writing these words knowing I would not be here without the support of the extraordinary people below. I will never be able to give back what they did for me, but I want them to know how much it means.

First of all is my wonderful supervisor **Pietro Liò**. It's fair to say that he is unlike any other professor I have ever met! I will be forever grateful for the risk that he took in me – a student with no prior background in Computer Science. But it was at the end of my first year when I realised the true depth and extent of his support. Those close to me will know the events that I am referring to here. All I can say is that he may be the busiest man in the department, but I know that he would stop at nothing if I were ever in need.

Next is **Petar Veličković**, who selflessly adopted a de-facto supervisory role during my first year, patiently showing me how to train my very first LSTM in Keras, and instilling a passion for research which will remain in me forever. Later on, the support became more personal as we developed a close friendship. I am beyond proud of what he has achieved and of the person he has become, and I will always be inspired by his work and his kindness. We will be lifelong friends.

Thirdly, I must say a heartfelt thanks to **Stephanie Hyland**, who entered my PhD journey at a time when I had very little confidence in my research abilities. This was a real turning point in my PhD, as she was able to turn my half-baked research ideas into competent science. Our work together became my first publication and it remains my proudest work. I hope that we will collaborate again in the future.

Next is **Sophie Xhonneux**, who gave her time and energy on weekends, evenings and sometimes in the middle of the night to help with both academic and personal matters. Be it a bug in my code, a new research idea, proofreading, or to lend an ear to whatever was bothering me at the time. She was a constant source of support despite going through a very tumultuous period in her own life and I hope she knows how much I appreciate that.

Finally, is **Ari Ercole**, who picked me up more times than I can count during the final months of this PhD. It's no secret that I was dealing with crippling self doubt and burn out during the writing process. The one thing that kept me on track through the tears was our weekly meetings, when he would patiently reassure me *again* that I was good enough to be awarded this degree.

On the academic side, I must also mention my close collaborators **Catherine Tong** and **Ioana Bica**, who are phenomenally intelligent researchers from whom I learned so much. **Rudolf Cardinal** was an enlightened source of advice and support during my first year. Discussions with **Charlotte Summers** and **Alex Campbell** were instrumental in forming some of my early research ideas. Fellow members of the lab, including **Kamilė Stankevičiūtė**, **Jacob Deasy**, **Helena Andrés-Terré**, **Tiago Azevedo**, **Duo Wang**, **Benjamin Day**, **Paul Scherer**, **Cristian Bodnar**, and **Cătălina Cangea**, created an environment in which it was easy to get excited about research. I would also like to express my gratitude to my PhD viva examiners, **Paul Elbers** and **Cecilia Mascolo**, for their insightful comments and suggestions for future work. **Lise Gough** offered friendly and reassuring guidance on the administrative aspects of my work, and I always left her office feeling calmer than when I entered. During the final months of my PhD, I received exceptional pastoral care from **Jason Ali**, my clinical director of studies at Churchill, who went above and beyond his duties. Finally, I want to acknowledge the **MB/PhD programme** and my funders, the **Armstrong Fund** and the **Frank Edward Elmore Fund**, whose financial support was invaluable in making this work possible.

On the personal side, I must mention my loving **parents**, whose unyielding belief in me has been invaluable throughout my academic journey, despite my unconventional choices. My grandparents **Nana** and **Pépé**, who offered to proofread everything no matter how far outside their own interests and understanding! My sister **Colette**, who shared in the joy of my successes and comforted me after every paper rejection, reminding me of my capabilities and my worth. My best friend **Sarah** provided the perfect escape during challenging moments, thanks to her unwavering loyalty and thoughtful planning of activities. Finally, **Seyon**, the love of my life, who is always here to remind me what is important in life. Thank you for being devotedly by my side through every deadline and stressful presentation. This PhD feels as much yours as it is mine.

Contents

Glossary

ABP Arterial Blood Pressure.

AI Artificial Intelligence; the theory and development of computer systems that are able to perform tasks normally requiring human intelligence [54].

ALT Alanine Transaminase; formerly known as Serum Glutamic-Pyruvic Transaminase (SGPT). It is a biomarker of liver damage.

APACHE-IV A popular risk scoring model which is evaluated only once after the first 24 hours of the patient's stay [136].

ARDS Adult Respiratory Distress Syndrome; when the lungs become severely inflamed due to an infection or injury and makes breathing difficult.

AST Aspartate Aminotransferase; formerly known as Serum Glutamic-Oxaloacetic Transaminase (SGOT). It is a biomarker of liver damage.

AUPRC Area Under the Precision Recall Curve; especially useful when there is significant imbalance in the test set e.g. only a few positive samples and many more negative samples.

AUROC Area Under the Receiver Operating Characteristic curve; plots the true positive rate against the false positive rate at various threshold settings; useful for evaluating classification algorithms.

BERT Bidirectional Encoder Representations from Transformers [25].

BiLSTM Bidirectional LSTM [40].

BPTT Backpropagation Through Time [91].

BUN Blood Urea Nitrogen; a biomarker of kidney function.

CCS Clinical Classifications Software [27].

CI Confidence Interval.

CNN Convolutional Neural Networks.

CRP C Reactive Protein.

CVP Central Venous Pressure; the pressure in the thoracic vena cava near the right atrium.

ECG Electrocardigram; a test for the heart's rhythm and electrical activity.

EHR Electronic Health Record; contains a patient's medical notes, diagnoses, medications, allergies, radiology images, laboratory tests etc.

eICU A multi-center ICU database with high granularity data for over 200,000 admissions to ICUs monitored by eICU Programs across the United States.

FC Fully-Connected; a neural network that connects every neuron in one layer to every neuron in the next layer.

FiO$_2$ Fraction of inspired Oxygen.

GAT Graph Attention Network; a neural network architecture that operates on graph-structured data, leveraging masked self-attentional layers to address the shortcomings of prior methods based on graph convolutions or their approximations..

GCN Graph Convolutional Networks [58].

GNN Graph Neural Network; a neural network architecture that operates on graph-structured data.

GPT Generative Pre-trained Transformer [78].

GraphSAGE SAmple and aggreGatE [45]; an approach for graph structured data.

IBW Ideal Body Weight; often calculated based on age, height and gender. It can be used to estimate optimal drug dosages, and also physiological variables such as Tidal Volume.

ICD International Classification of Diseases; international standard diagnostic tool for epidemiology, health management and clinical purposes [129].

ICU Intensive Care Unit; department of the hospital typically filled with very sick patients who require life support.

LoS Length of Stay.

LSTM Long Short-Term Memory; a type of recurrent neural network that summarises time series data [50].

MAD Mean Absolute Deviation.

MAPE Mean Absolute Percentage Error.

MIMIC Medical Information Mart for Intensive Care; a large, single-center database comprising information relating to patients admitted to critical care units at a large tertiary care hospital in the United States.

ML Machine Learning; a subset of AI techniques that allow machines to learn automatically from past data without explicit programming.

MPNN Message Passing Neural Networks [33].

MSE Mean Squared Error.

MSLE Mean Squared Logarithmic Error.

MV Mandatory Ventilation; the mandatory ventilation settings used in this disseration are defined in Table C.6.

NAFLD Non-Alcoholic Fatty Liver Disease.

NHS National Health Service; the publicly funded healthcare system in the UK.

NLP Natural Language Processing.

OPCS Office of Population Censuses and Surveys [76].

PaO_2 The partial pressure of oxygen in arterial blood.

PCOS Polycystic Ovary Syndrome.

PEEP Positive End-Expiratory Pressure; designed to prevent the alveoli from collapsing on expiration.

PTT Partial Thromboplastin Time; also known as the activated Partial Thromboplastin Time (aPTT). It measures how long it takes for a clot to form in a blood sample.

RCT Randomised Controlled Trial; where subjects are randomly assigned to either an experimental group (receiving the intervention that is being tested), or the control group (receiving the conventional or a placebo treatment).

ReLU Rectified Linear Unit; a type of activation function used in neural networks.

RGB A colour model in which the red, green, and blue primary colours of light are added together in various ways to reproduce a broad array of colours.

RNN Recurrent Neural Network.

SAH Sub-Arachnoid Haemorrhage; a type of stroke associated with particularly high mortality in the ICU.

SaO_2 Arterial Oxygen Saturation.

TCN Temporal Convolution Networks.

TPC Temporal Pointwise Convolution; a neural network made up of pointwise and temporal convolution [95].

TV Tidal Volume; the amount of air that moves in or out of the lungs with each respiratory cycle.

VD Ventilation Duration.

WBC White Blood Cell Count.

CHAPTER 1

Introduction

Over the past decade, we have seen accelerating interest in Artificial Intelligence (AI) for healthcare. This is reflected in the exponential increase in published research papers [133], soaring investment into AI startups [102], and year-on-year increases in regulatory approvals for AI software devices [6]. This is an encouraging prospect at a time when many healthcare systems including the UK's National Health Service (NHS) are struggling with unprecedented demand on services. Acutely, some of the stress is due to the lasting impact of the COVID-19 pandemic, however, ageing populations and the continual development of new treatments will ensure that this trend is likely to continue. Looking to the success in other industries, there is cautious optimism that AI could be part of a long-term solution to improve the efficiency and personalisation of care, alleviating some of the ever-increasing burden on services.

1.1 AI, Machine Learning and Deep Learning

Although there is no universally accepted definition, Joiner [54] describes AI as "the theory and development of computer systems that are able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages". As 'human intelligence' is subjective, the resulting field of AI is dynamic and diverse. Machine Learning (ML) is a more specific term, referring to a subset of AI techniques that allow machines to learn automatically from data without explicit programming. However, the learning process may only be partially automated because the data often requires some degree of feature engineering[1] before it is given to the algorithm (Figure 1.1).

Deep learning is a subset of machine learning that focuses on *end-to-end* systems

[1]Feature engineering is the process of using domain knowledge to extract features (characteristics, properties, attributes) from raw data.

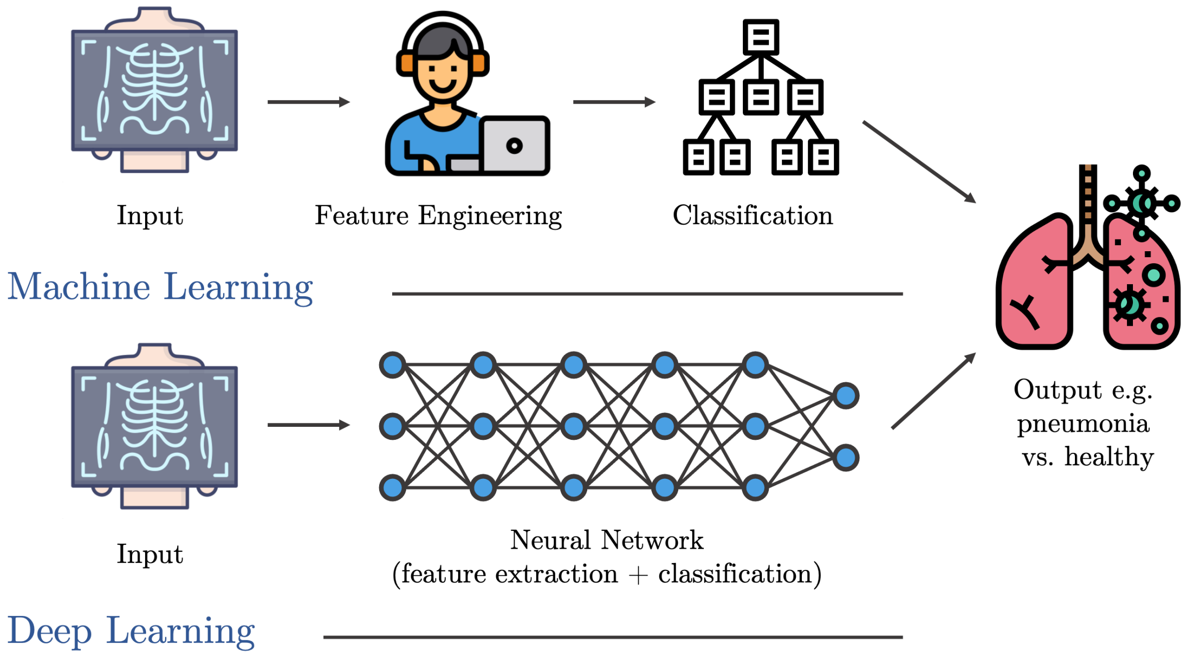

Input
Feature Engineering
Classification
Machine Learning
Output e.g. pneumonia vs. healthy
Input
Neural Network (feature extraction + classification)
Deep Learning

Figure 1.1: A traditional machine learning vs deep learning approach to pneumonia classification from X-rays. In this example, the feature engineering process might create low-level features, such as the contrast or exposure of the X-ray, or high-level features such as the size of the heart and lungs or percentage consolidation of the lung.

designed to learn from raw data without manual feature engineering. It is synonymous with a particular type of machine learning model called a neural network (explained in detail in Section 2.1). Deep learning has shown incredible promise over the past decade; for example, in identifying objects in images, transcribing speech into text, matching news items or products with users' interests, and selecting relevant search results [61]. When applied to healthcare, I highlight three of its particular strengths:

Scalability to large data sets Once a workflow is established, the incremental cost of adding data is small, making the system cheaply scalable and easy to keep up to date as new data becomes available.

Capacity to supersede human performance on specialised tasks Deep learning is well suited to solving highly specific tasks with quantifiable performance metrics. It exploits reliable patterns in patient data without getting fatigued.

Automation Deep learning tools may have the biggest impact in situations where there is significant stress on hospital resources. It has the potential to increase the capacity of a service by easing the workload demands for highly trained specialists, translating to real differences in patient outcomes. For example, a reliable speech to text documentation system for medical notes could increase the number of appointments per clinic.

1.2 Learning Representations of the Patient

In order to make intelligent predictions in a healthcare setting, we usually need to learn a *representation* of the patient. By this we mean finding a low-dimensional transformation of the original data, such that it could serve as a useful input to a simple predictor or classifier model. In other words, learning representations is all about finding *structure* in the data. Identifying ways in which the data can be meaningfully separated, isolating invariant properties from noise, or creating interpretable high dimensional objects are all examples of representation learning problems.

Before deep learning came to the fore, much of the effort in designing machine learning algorithms would go towards pre-processing pipelines and feature engineering in order to extract meaningful representations. This was especially true in medicine, where domain knowledge is absolutely necessary to make sense of the data. As highlighted in the previous section, a key strength of deep learning is its capability for automatic learning from raw data. To do this successfully, the model must have enough capacity to represent complex functions, but also sufficient data to learn the representation. Currently, we are limited by both the quantity and the quality of Electronic Health Record (EHR) data. It can be sparse, incomplete, irregularly sampled, and riddled with errors. When faced with such challenges, learning a good representation relies on careful design of the network such that it can learn efficiently in spite of these factors. Note that this is a little different to the approach taken by OpenAI in the GPT model series [78] or by various CNNs trained on ImageNet [24], where high quality data is cheap. Vast and densely-connected models are very powerful in these scenarios, but in the context of limited medical data we need to be more selective with the size and connectivity of the network. That is to say, the current challenge when designing deep representations for patients is not so much feature engineering, but rather *network engineering.*

First of all, let us consider what makes a representation better than another. What are the desirable qualities? Even more importantly, how can we design learning objectives to encourage the models to learn 'good' representations of the patient? I will address the first question by outlining some general principles [5]:

Smoothness Similar inputs should produce similar outputs i.e. if $x \approx y$ then $f(x) \approx f(y)$. If the function represented by the network is smooth, then the outputs will be more robust to noise in the input.

Expressivity A reasonably sized learned representation should be able to capture a huge number of possible input configurations. For example, there are many possible manifestations of the same diagnosis which need to be mapped to the same concept.

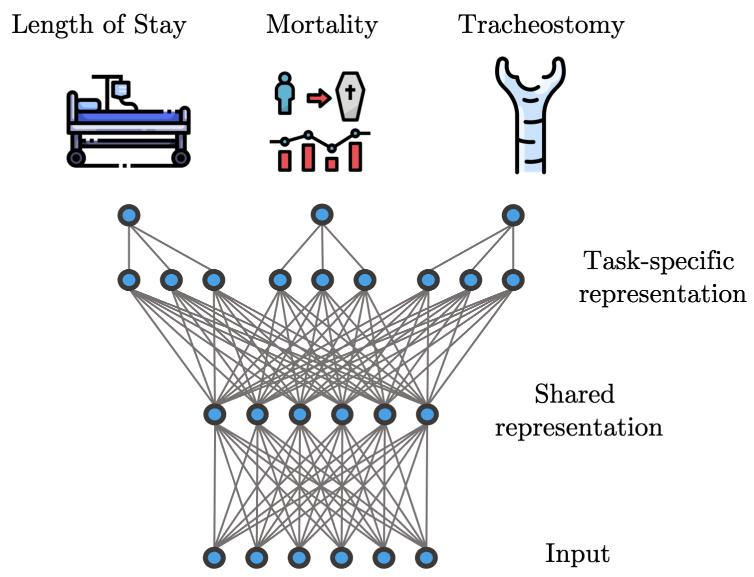


Figure 1.2: A multitask example of patient representations, where input features are used to predict length of stay, mortality and whether the patient has had a tracheostomy. The first layer creates a shared representation of the input, which allows the model to exploit factors which are shared between tasks e.g. the health status of various organs. Then there are task-specific representations, which keep only the components which are relevant for that task e.g. respiratory factors for the tracheostomy task. Since the representations overlap, the statistical strength can be shared as the model learns, which aids generalisation.

Sharing explanatory factors With inputs X and a target Y, a subset of factors which explain the distribution of X, can explain much of Y, given X. Therefore representations that are useful for $P(X)$ tend to be useful for learning $P(Y|X)$. This is also true when there are multiple tasks i.e. $P(Y|X, \text{task})$ will be explained by factors which are shared with other tasks. This explains why semi-supervised, multitask learning, transfer learning and domain adaption can all lead to more robust representations, because they allow the sharing of statistical power between them. This concept is shown in Figure 1.2.

Hierarchy Concepts that are useful for describing the world can be defined in terms of other concepts. Initially we compute 'low-level' factors, which could be basic signal processing factors, but these eventually combine to form 'high-level' abstract concepts. Note that we also see this organisation in sensory processing in the brain.

Manifolds The probability mass of learned representations will become dense in certain areas which represent different examples of the same concept and sparse in others, where no general concept exists. A good representation will therefore tend to naturally cluster itself into a low dimensional space.

Simple dependencies In a good high-level representation, the factors are related to each other through simple, often linear dependencies. This can be tested by assessing the performance of a simple predictor on top of a learned representation.

Deep learning is ideally suited to learning representations. Not only does it have the capacity to model non-linearities and feature interactions, its organisation into sequential layers provides a natural opportunity to construct hierarchies and share information between tasks. Another important property of deep networks is that the number of paths through the network scales exponentially with the depth. This means there are many ways in which disparate inputs can map to a similar output. This is the expressivity principle.

The second question that I posed, about how to encourage models to learn useful representations of the patient, will be addressed in detail throughout this dissertation. I am strongly guided by the foundational principles outlined above, but in the context of healthcare I will argue that we can go further to take advantage of what we already know in clinical medicine. For example, by mimicking aspects of traditional clinical reasoning, or by leveraging existing knowledge of the data structure, we might be able to engineer models to improve upon the state-of-the-art.

1.3 Research Questions

Having provided an introduction to representation learning and surveyed the landscape of challenges that we face, I will now formulate an overall theme and three research questions that I seek to answer within this dissertation. I have provided a visual summary in Figure 1.3. The consistent theme is:

Can we improve our representation of the patient using clinical knowledge about the structure of EHR data in the ICU?

In the scope of this dissertation, I cannot hope to analyse every possible method of exploiting clinical knowledge to improve representations of the patient. Instead, I aim to show that the approach has been useful in three separate pieces of work. These research questions are outlined below:

Q1. *Can we use our knowledge of medical time series to design a specialised convolutional model to improve the performance of patient outcome prediction models?*

Q2. *Can we leverage data from similar patients to provide additional context to time series models when making predictions about patient outcomes?*

Q3. *Can we use auxiliary prediction tasks to guide the representations to reflect the patient phenotype, trajectory and outcomes, with a view towards uncovering subtypes of disease?*

My research has focused on addressing challenges in healthcare at the *system-level*, as these problems have fewer barriers to implementation, which can lead to greater real-world impact in the short to medium term. In addressing research questions **Q1** and **Q2**, the patient outcome predictions (mortality and length of stay) were intended to aid in hospital bed management rather than clinical decision support. In Section 3.1, the importance of hospital bed management will be further explained. In addressing **Q3**, the time series models were used in an exploratory manner, for example, to discover hidden phenotypes that could inspire future clinical studies or guide the investigation of targeted treatment strategies.

While I have focused on problems in healthcare, it is worth noting that the resulting time series methods could apply to a range of domains beyond healthcare. In fact, by exploiting the properties of time series data, including periodicity, long-term trends, and graph structure, my work can aid in better understanding the underlying patterns and trajectories in diverse fields such as finance markets, energy consumption, social media, and transportation.

1.4 Thesis Outline

My dissertation follows the structure detailed below. Each contribution chapter involves developing a new deep learning technique that is useful for extracting a particular patient representation in the ICU.

- **Chapter 2: Theoretical Foundations.** Firstly, I review the necessary theoretical background that underlies the research presented in the contribution chapters. In particular, I cover recurrent, convolutional and graph neural networks, with reference to important related works.

- **Chapter 3: Temporal Pointwise Convolution.** In this chapter, I focus on **Q1**, which is concerned with improving the representation of medical *time series* for improved performance when predicting patient outcomes. Specifically, I predicted the mortality risk and remaining length of stay of patients in the ICU by designing a new model which combined both temporal and pointwise convolution. I used temporal convolution to extract trends across the time dimension whereas pointwise convolution specialises in extracting inter-feature relationships. Both of these are useful when assessing the health status of a patient for the purpose of predicting outcomes. I was able to improve on the state-of-the-art models. My long term vision is that automatic mortality and length of stay prediction will enable sophisticated bed management strategies to streamline the patient journey through ICU, reducing costs and hospital acquired infections.

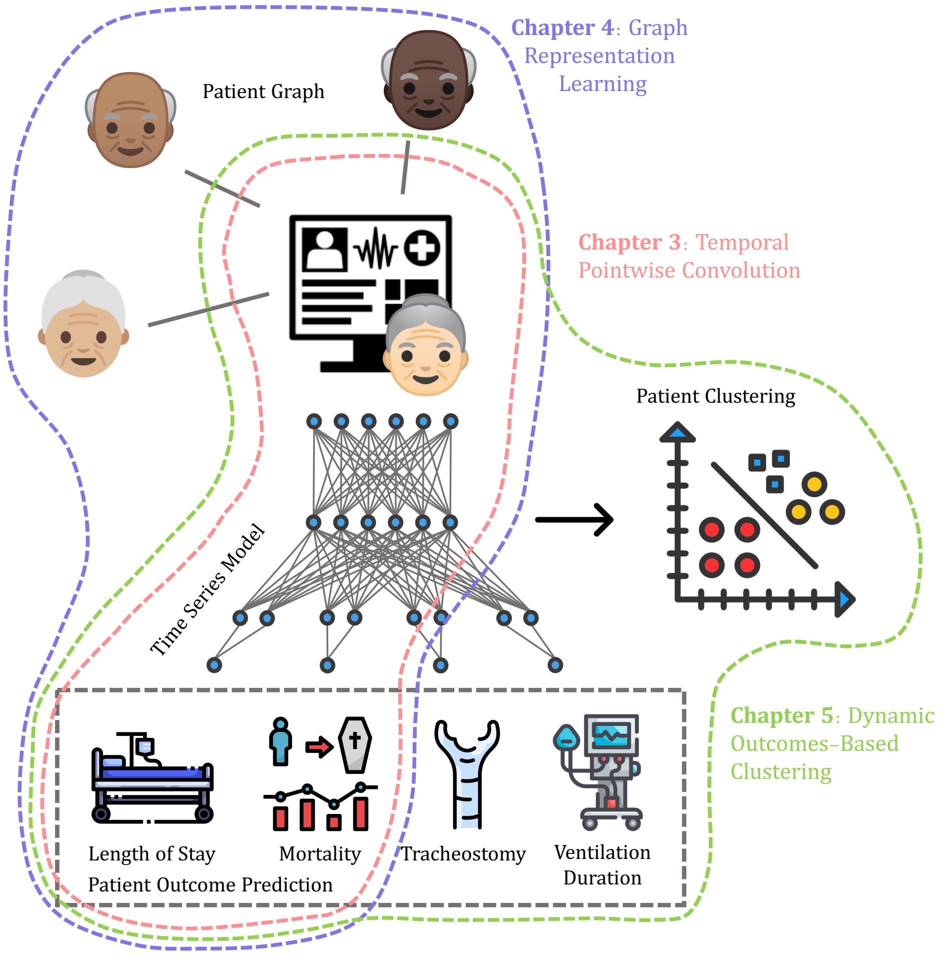

Chapter 4: Graph Representation Learning
Patient Graph
Chapter 3: Temporal Pointwise Convolution
Patient Clustering
Time Series Model
Chapter 5: Dynamic Outcomes-Based Clustering
Length of Stay
Mortality
Tracheostomy
Ventilation Duration
Patient Outcome Prediction

Figure 1.3: A visual abstract of this dissertation. Chapters 3 and 4 propose new methodologies to exploit particular data types in the EHR. Chapter 5 clusters patient trajectories in order to find consistent phenotypes in patients undergoing mechanical ventilation.

- **Chapter 4: Graph Representation Learning.** In this chapter, I concentrate on the representation of *sparse data* in the EHR as outlined in **Q2**, specifically information such as diagnosis codes, medications and procedures. I used the diagnoses as a proof of concept to build a 'patient graph' of related patients. We then used this patient graph along with a graph neural network e.g. Graph Attention Network (GAT) [123] to predict the mortality risk and length of stay, as in Chapter 3. We found that representing patients within a neighbourhood or 'context' of similar patients led to slightly improved prediction performance. To my knowledge, this method of representing the diagnoses had not been done in prior work. Our approach could be easily extended to medications and treatments – indeed any form of sparse data that yields a useful neighbourhood of similar patients.

- **Chapter 5: Dynamic Outcomes-Based Clustering.** In my final contribution chapter, aimed at **Q3**, I delve further into patient representations with the goal of uncovering hidden phenotypic subtypes in the trajectories of patients who underwent mechanical ventilation. I encouraged the learned representations (and therefore the resulting clusters) to reflect the patient *phenotype*, *trajectory* and *outcomes.* I used a mixture of supervised and unsupervised techniques to achieve this, and I was able to make use of my previous Temporal Pointwise Convolution architecture developed in Chapter 3 to improve the representations. I also employed various visualisation techniques to learn more about the structure of the representation space.

- **Chapter 6: Conclusion and Future Directions.** Finally, I summarise the contributions presented in the thesis and discuss future directions for each.

1.5 List of Publications

The research efforts presented in this thesis have led to peer-reviewed publications, which are listed below (in order of contribution chapters 3, 4, 5):

[95] **Rocheteau, E.**, Liò, P., Hyland, S. (2021). Temporal pointwise convolutional networks for length of stay prediction in the intensive care unit. *Proceedings of the Conference on Health, Inference, and Learning, CHIL'21. Association for Computing Machinery.* `https://dl.acm.org/doi/10.1145/3450439.3451860`.

A previous version of the work was selected for an oral presentation at the *Machine Learning for Health (ML4H) Workshop at NeurIPS 2020* and again as a spotlight talk at the *Healthcare Systems, Population Health, and the Role of Health-Tech (HSYS) Workshop at ICML 2020* [94]. The talk can be seen by clicking on this link.

[119] Tong, C.*, **Rocheteau, E.***, Veličković, P., Lane, N., Liò, P. (2022). Predicting Patient Outcomes with Graph Representation Learning. *AI for Disease Surveillance and Pandemic Intelligence: Intelligent Disease Detection in Action. W3PHAI 2021. Studies in Computational Intelligence, Springer.* `https://doi.org/10.1007/978-3-030-93080-6_20`.

I presented this work as a spotlight talk at the *Health Intelligence (W3PHAI) Workshop at AAAI 2021* where we were awarded 2nd Runner-Up for Best Short Paper Award. The talk can be seen by clicking on this link. The work was also presented by Catherine at the *Deep Learning on Graphs Workshop at AAAI 2021.*

[96] **Rocheteau, E.**, Bica, I., Liò, P., Ercole, A. (2022). Dynamic Outcomes-Based Clustering of Disease Trajectory in Mechanically Ventilated Patients. *Learning from Time Series for Health Workshop at NeurIPS 2022.* `https://openreview.net/pdf?id=S7FEB6rwc5R`.

I also presented this work as a spotlight talk at the *AI for Social Good (AI4SG)*

Workshop and the *Health Intelligence (W3PHAI) Workshop at AAAI 2023*, and at the latter I was awarded Best Paper. The talk can be seen by clicking on this link. Finally, I presented as a poster at the *Representation Learning for Responsible Human-Centric AI (R2HCAI) Workshop at AAAI 2023.*

I have additionally published the following articles during my PhD years, which do not relate directly to the work in this dissertation:

[92] **Rocheteau, E.** (2022). On the role of artificial intelligence in psychiatry. *The British Journal of Psychiatry.* `https://doi.org/10.1192/bjp.2022.132`.

This work won the Eliot Slater Prize in Psychiatry from the Department of Psychiatry in Cambridge.

[23] **Rocheteau, E.**, Deasy, J., Kohler, K., Stubbs, D., Barbiero, P., Liò, P., Ercole, A. (2020). Rapid Design and Implementation of a Data-Driven Forecast of ICU Strain from COVID-19 for Early Surge Planning in England. *Intensive Care Medicine Experimental, 8(2):000267, 73.*

I was selected for an oral presentation at the 33rd Annual Congress, ESICM LIVES 2020, this can be viewed here. Note that the full length preprint can be found at `https://www.medrxiv.org/content/10.1101/2020.03.19.20039057`.

[93] **Rocheteau, E.***, Kim, D.* (2020). Deep Transfer Learning for Automated Diagnosis of Skin Lesions from Photographs. *Machine Learning for Mobile Health (ML4MH) Workshop at NeurIPS 2020.* `https://arxiv.org/abs/2011.04475`.

I also have the following peer reviewed abstracts accepted for presentations at conferences:

- **Rocheteau, E.**, Deasy, J., Roggeveen, J.F., Ercole, A. (2020). ICUnity: A software tool to harmonise the MIMIC-III and AmsterdamUMCdb databases. *Machine Learning for Healthcare Conference (MLHC) 2020.* `https://www.mlforhc.org/2020accepted-papers`.

  The talk can be viewed by clicking on this link, and the code can be accessed here.

- **Rocheteau, E.**, Liò, P. (2018) Predicting outcomes in psychiatric disorders using automated reinforcement learning analysis of Electronic Health Records. *2nd Human Brain Project Student Conference, Ljubljana, Slovenia.*

  I won the Best Poster Prize among 49 submissions.

Whenever possible, I have endeavoured to make the source code for my work publicly available on my GitHub profile: `https://github.com/EmmaRocheteau`.

CHAPTER 2

Background

The original inspiration for deep learning came from neuroscience [47]. In the 1980s, early AI researchers knew that the brain comprised networks of neurons propagating signals encoded as action potentials. As these biological neural networks were known to underpin human intelligence, they hypothesised that they could induce intelligent reasoning if they replicated the basic structure of these networks.

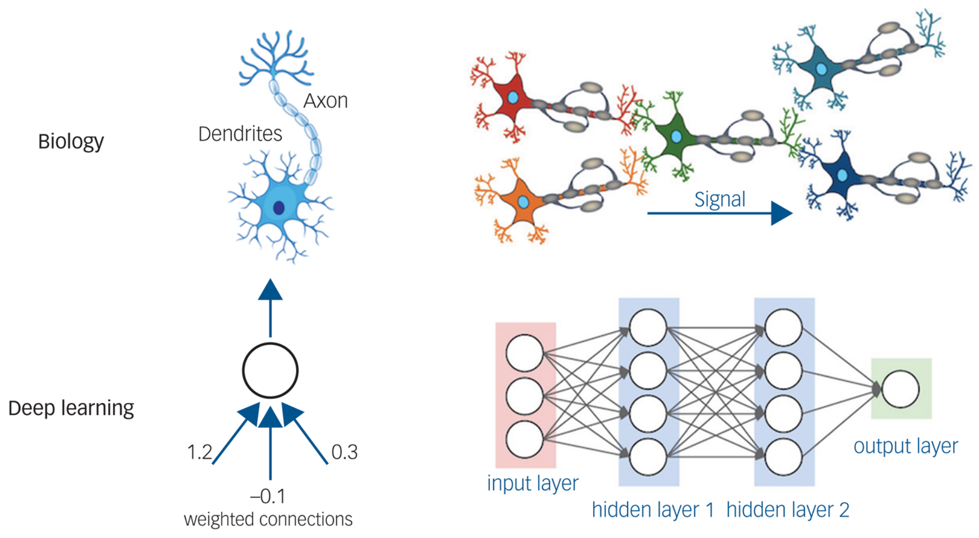


Figure 2.1: A comparison of biological and artificial neural networks. The basic structure consists of layers of neurons (shown as circles) that are connected together by axons (represented by arrows with associated 'weights' which indicate the strength of the connection between those neurons). Modified from the original with permission from Laura Dubreuil Vall.

Therefore, the discovery of biological neural networks laid the foundations for deep learning. The core concept that neurons are connected together in layers is preserved (Figure 2.1) with some modifications to improve computational efficiency or performance. Just like in the brain, the architecture and connectivity of the neurons can be specifically designed to extract particular representation from a given input. A good example of this occurring in biology are the simple cells in the primary visual cortex, which are specialised for extracting lines and edges from images.

Deep learning works by trial and error. If we imagine that we want to classify patients who have died vs. those who survived, we need examples of both outcomes in the training data. Before training, the AI will not know how to distinguish the patients. However, each time the AI gets the classification wrong, the 'weights' between the artificial neurons will be updated such that the network is less likely to return the same mistake again. To do this we use an algorithm called backpropagation in conjunction with an optimisation method such as gradient descent [99].

2.1 Fully-Connected Neural Networks

Neural networks are theoretically are capable of approximating any bounded continuous differentiable function, given a sufficient number of labelled (input, output) variables [19]. Each neuron computes a weighted sum of its inputs from the previous layer, potentially applying a nonlinear transformation such as logistic, tanh, or most commonly Rectified Linear Unit (ReLU). At the most basic level, the operation of a single perceptron (artificial neuron) can be written as:

$$y = \sigma\left(b + \sum_{i=1}^{n} w_i x_i\right) \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the input vector, $\mathbf{w} \in \mathbb{R}^n$ is a weight vector, b is a bias value, σ is a non-linear activation function, and y is the output value. We can expand this to represent a fully-connected neural network layer:

$$\mathbf{y} = \sigma\Big(\mathbf{W}\mathbf{x} + \mathbf{b}\Big) \tag{2.2}$$

where $\mathbf{W} \in \mathbb{R}^{m \times n}$ is a weight matrix, $\mathbf{b} \in \mathbb{R}^m$ is a bias vector, and $\mathbf{y} \in \mathbb{R}^m$ is the output vector. Neural networks that have one or more 'hidden' layers between the input and output layers are referred to as 'deep'. For example a two-layered perceptron (also shown diagrammatically in Figure 2.2) can be written as:

$$\mathbf{y} = \sigma_2\Big(\mathbf{W}_2\sigma_1\Big(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\Big) + \mathbf{b}_2\Big) \tag{2.3}$$

Networks with greater depth can execute more instructions in sequence [37], which can help represent complexity. However, larger networks are generally more prone to *over-fitting* during training. This can often be partially offset using regularisation techniques such as dropout[1] [110] and batch normalisation[2] [52].

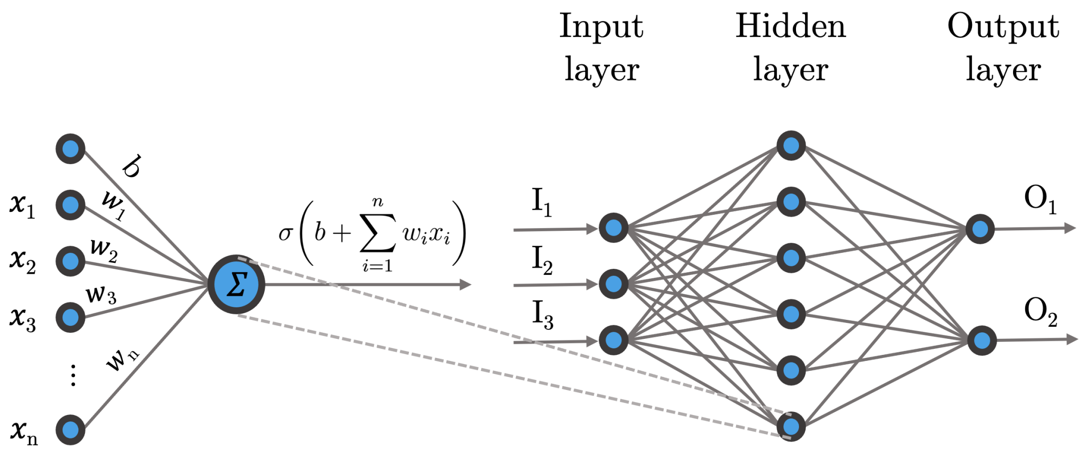


Figure 2.2: A two-layered perceptron.

The layers depicted here are fully-connected, that is, every neuron in the preceding layer is connected to every neuron in the subsequent one. Usually, we only use fully-connected networks when we are dealing with unstructured data. This is because if something about the structure, then it can be exploited for better performance. In the following sections, I will present some of the ways in which we can specialise deep networks to process particular inputs. Note that I implement and use every architecture mentioned in this section in this dissertation, mostly as baselines to compare to my novel architectures.

2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are used for processing time series data. If we briefly consider the structure of sequential data, usually the following are true:

- There are a fixed set of features for each time step, t.
- The length of the series can be arbitrary.
- The time step between each time point is regular and fixed.
- It some cases, the data that comes at timestep t can only be explained by the data that comes before it in the sequence $\mathbf{x}_{1:t}$ (note this is not true for speech).

Recurrent neural networks can summarise time series data into a vector of consistent size (Figure 2.3). RNNs process the input one timestep at a time, meaning that the input

[1]Dropout is a regularisation strategy which effectively trains an ensemble of sub-networks by removing non-output units randomly from the original network.

[2]Batch normalisation performs the normalisation for each training mini-batch, to accelerate training by reducing internal covariate shift i.e. the change of parameters of previous layers will change each layer's input's distribution.

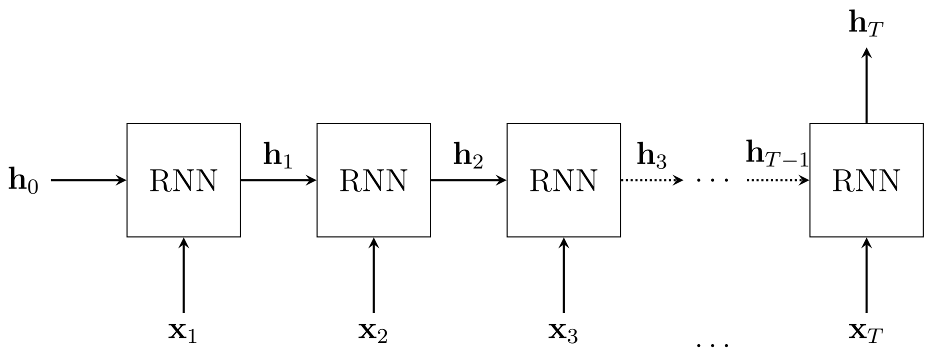


Figure 2.3: Unfolded RNN. The vector $\mathbf{h}_t$ is computed iteratively across all timesteps. The final hidden state $\mathbf{h}_T$ can then be used as appropriate e.g. it can be fed into a fully-connected network to get a final classification or prediction.

can be of any length, however it also means that they have a depth equal to the number of steps in the time series. Each RNN cell shares the same weights, and these are updated together using backpropagation through time (BPTT) [91].

A significant problem with the simple RNN is *vanishing gradients* – this means that timesteps that occur early in the sequence are more likely to be 'forgotten' than recent timesteps. This can be particularly true when the input is very long and the RNN is very deep.

2.2.1 Long Short-Term Memory

Long Short-Term Memory networks (LSTMs) [50] are a specific type of RNN that attempts to counter the vanishing gradient problem. The specific components of the LSTM are found within the cells that process each timestep. A single LSTM cell is defined as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i) \tag{2.4}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f) \tag{2.5}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o) \tag{2.6}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh\left(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c\right) \tag{2.7}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{2.8}$$

In these equations, $\mathbf{x}_t$ is the input and $\mathbf{h}_t$ is the output of the LSTM cell. $\mathbf{W}_*$, $\mathbf{U}_*$ and $\mathbf{b}_*$ correspond to learnable parameters, i, f, o and c refer to the input gate, forget gate, output gate and new features respectively (see Figure 2.4), $\odot$ corresponds to element-wise multiplication, tanh is the hyperbolic tangent function and σ is the sigmoid (logistic) function.

Figure 2.4 shows a diagram of the various gates and structure of an LSTM cell. The input gate, $\mathbf{W}_i$ (Equation 2.4) decides which candidate values will be updated in the cell

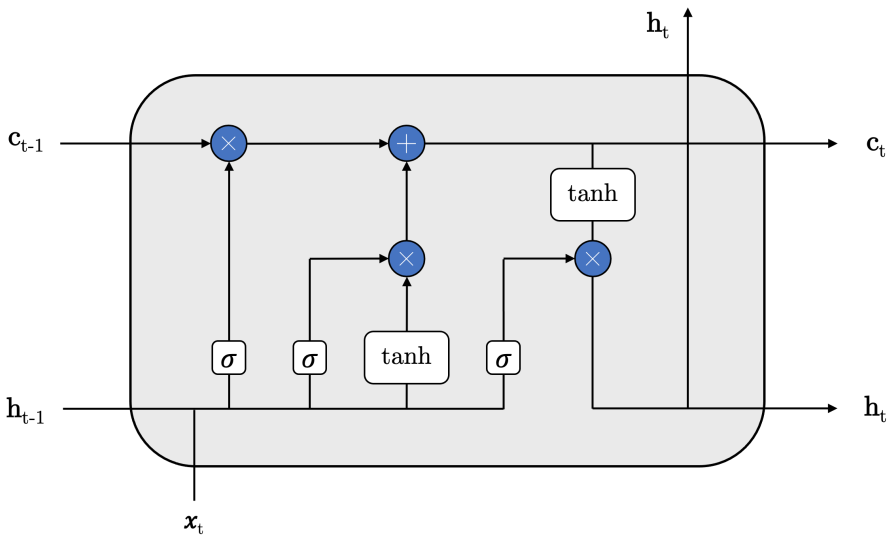


Figure 2.4: An LSTM cell. The forget gate appears first on the left, then the input gate is in the middle, and the output gate is found on the right.

state. The forget gate, $\mathbf{W}_f$ (Equation 2.5) decides which values will be remembered from the previous cell state. The matrix $\mathbf{W}_c$ (Equation 2.7) calculates a new set of candidate values that could be added to the state. The output gate, $\mathbf{W}_o$ (Equation 2.6) decides which components of the new cell state are allowed to exit the cell.

The power of the LSTM comes from its ability to learn what to remember and what to forget. They have effectively surpassed the performance of simple RNNs [50].

Bi-directional LSTM When the causality assumption is not true of a particular input, it can be useful to process the data in both the forward and backward directions [40]. The final hidden states from each direction are typically concatenated to form the output.

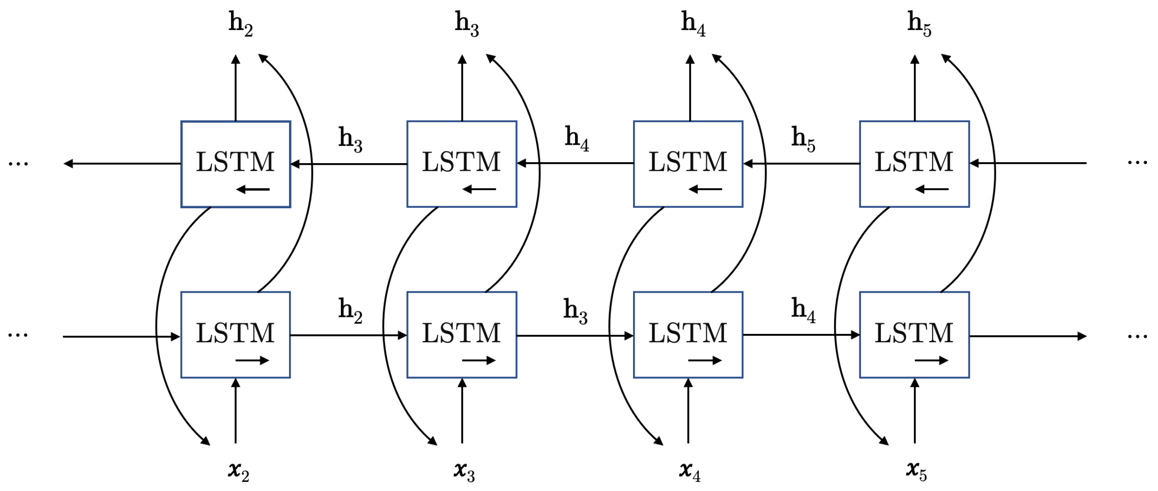


Figure 2.5: A bidirectional LSTM (BiLSTM).

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are typically used for processing image data, however they are not limited to this (for instance we use them for temporal data in this dissertation). For image data, the following are usually true:

- The shape takes the form $h \times w \times d$, where h is the height, w is the width and d is the depth, or number of channels. For example in RGB images, d will be 3.
- Each pixel has a fixed number of neighbours which represent fixed relative displacements.
- Neighbouring pixels are much more likely to be related than pixels in different parts of the image (local structure).
- Objects in the image may be of different sizes.

Grid-like structures are exploited by CNNs. In particular, the spatial structure in visual data is taken into account by processing neighbouring pixels with *kernels* – miniature grids which slide over the image but maintain the same set of parameters. The vast degree of parameter sharing in CNNs makes them very parameter efficient and less likely to overfit.

A single convolutional operator for a 3D image tensor can be defined as follows[3]:

$$(\mathbf{I} * \mathbf{K})_{ijk} = \sum_{a=1}^{n} \sum_{b=1}^{m} \sum_{c=1}^{d} I_{i+a,j+b,k+c} K_{a,b,c} \tag{2.9}$$

where $\mathbf{I} \in \mathbb{R}^{h \times w \times d}$ is the image tensor and $\mathbf{K} \in \mathbb{R}^{n \times m \times d}$ is the kernel tensor. This is usually more intuitive to understand in a diagram format (Figure 2.6). A single channel operator (the case where $d = 1$) is shown for simplicity.

The kernel slides over all possible locations in the image, to record the sum of the elementwise products. We can see that the convolutional operator exploits the fixed structure between neighbouring pixels, as these are reliably placed within the kernel.

If the input data is images, it is usually necessary to downsample the data with pooling layers. These *summarise* convolutional outputs into smaller representations, before the next convolutional layer is applied. We will not discuss pooling layers in detail because we do not use image data in this dissertation. However, the principle of summarising data and subsequently stacking convolutional layers *is* relevant for our purposes. In particular as we move deeper through a CNN, we obtain increasingly high-level representations of the input and the receptive field size[4] of the kernels becomes larger.

[3]Strictly speaking, the operator shown is *cross-correlation*, not convolution, but the difference is unimportant; in cross-correlation, the kernel slides across the image without being flipped.

[4]By 'receptive field' we mean the area within the input image which can influence the output of a kernel placed at a specific location.

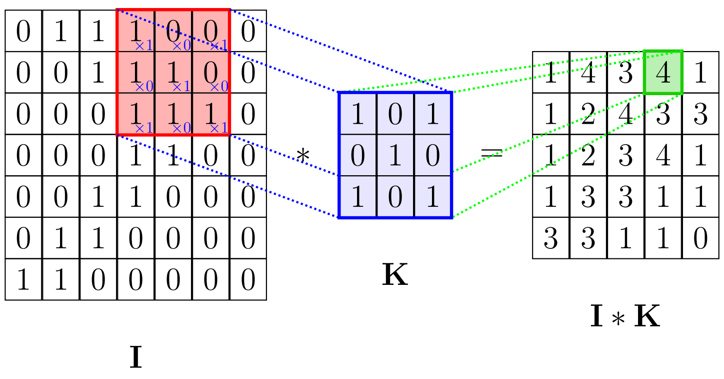


Figure 2.6: A convolution operator applied to a single channel image ($d = 1$), adapted with permission from Veličković [122].

2.4 Graph Neural Networks

So far we have covered ways in which we can alter the connectivity of neural networks to incorporate inductive biases about time series and grid-like data. Graph Neural Networks (GNNs) are designed for graph structured data. Unlike grid-structured data, the neighbourhood of nodes in a graph can have an irregular pattern. There are fewer definitive things to say about graphs, however the following are usually true:

- There is a finite set of nodes with a set of edges between them.
- The edges can be undirected or directed.
- Each node may have a different number of neighbours.
- The edges may have values associated with them e.g. they may have a distance to represent that the nodes are not equally spaced even if they are linked with an edge.
- The total number of nodes may be mutable in some scenarios.

Evidently, we need a fairly flexible architecture to exploit this data structure. The aim is to distil the high-dimensional information about a node's neighbourhood into a dense vector embedding. These node embeddings can then be fed to downstream machine learning systems and aid in tasks such as node classification, clustering, and link prediction. In the following sections, I introduce four graph methodologies which I will go on to use in Chapter 4.

2.4.1 Graph Convolutional Networks

The Graph Convolutional Network (GCN) [58] was one of the early GNNs to generalise ideas from CNNs to the graph domain. If we assume a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ contains a set of

N nodes $\mathbf{v}_i \in \mathcal{V}$ with edges $(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{E}$. From this we can obtain the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ and degree matrix $\mathbf{D}_{ii} = \sum_j^N A_{ij}$.

The l-th GCN layer follows the propagation rule:

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^l\mathbf{W}^l\right) \tag{2.10}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ is the adjacency matrix with self loops inserted, $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$, $\mathbf{H}^l \in \mathbb{R}^{N \times d^l}$ are intermediate node features from layer l, where d^l is the number of features, and $\mathbf{W}^l \in \mathbb{R}^{d^l \times d^{(l+1)}}$ is a shared set of learnable parameters that are applied node-wise to obtain $d^{(l+1)}$ higher level features.

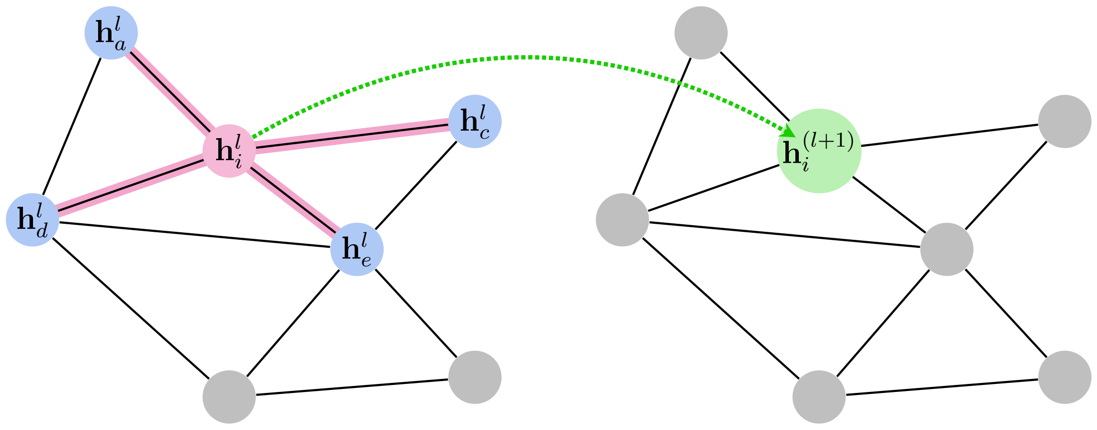


Figure 2.7: A GCN layer.

A graph convolutional layer generates a higher-level representation of a node i by leveraging the data from its neighbourhood, $\mathcal{N}_i$ (Figure 2.7). When written node-wise, the rule follows:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_j|}}\mathbf{W}^l\mathbf{h}_j^l\right) \tag{2.11}$$

where $\mathcal{N}_i$ includes the i-th node, and $\mathbf{h}_j^l \in \mathbb{R}^{d^l}$ represents the node features in layer l.

Analysing this rule, we can see that the node is updated based on its current features and the features of its neighbours. The self-loops allow the node's own features to be preserved across layers. Crudely, the GCN can be compared to *averaging* within the neighbourhood in each layer. This means that it works best when the edges represent a uniform degree of *similarity* between nodes.

2.4.2 Graph Attention Networks

The Graph Attention Network (GAT) [123] is broadly similar to the GCN except that it uses self-attention (Section 2.5) to determine how much each node should be weighted when recombining the features within the neighbourhood. A GAT layer with a single attentional head can be written as follows:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}^l\mathbf{h}_j^l\right) \tag{2.12}$$

where:

$$\alpha_{ij} = \text{softmax}_j\Big(\sigma'\big(\mathbf{a}^\top[\mathbf{W}^l\mathbf{h}_i^l \parallel \mathbf{W}^l\mathbf{h}_j^l]\big)\Big) \tag{2.13}$$

where $\mathbf{a}^\top$ is a transposed vector of attentional weights $\mathbf{a} \in \mathbb{R}^{2d^l}$, $\parallel$ represents the concatenation operation, and σ' is a LeakyReLU [66]. Again the neighbourhood $\mathcal{N}_i$ includes the self node. When multiple attentional heads are used, the output from each head is concatenated. This is shown diagrammatically in Figure 2.8.

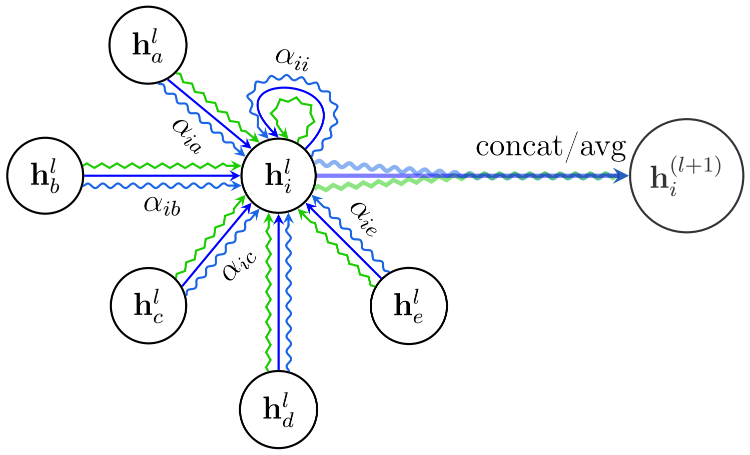


Figure 2.8: A GAT layer with three attentional heads (represented with blue, purple, green), modified from the original work with permission [123].

The GAT can model more complex relationships within neighbourhoods than the GCN, because the attentional weights can draw the focus onto particular nodes, and even apply negative weightings to certain nodes. Effectively the GAT has the power to apply *different importances* to nodes in the same neighbourhood.

2.4.3 GraphSAGE

The sampling method from GraphSAGE (SAmple and aggreGatE) [45] is a method for computing node representations in an *inductive* manner. It operates by sampling a fixed-size neighbourhood (uniform with replacement) of each node and aggregating e.g. by taking the mean, the element-wise minimum/maximum of the feature vectors, or feeding them through a neural network such as LSTM. We show GraphSAGE-mean as an example:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i^*} \frac{1}{|\mathcal{N}_i^*|}\mathbf{W}^l\mathbf{h}_j^l\right) \tag{2.14}$$

where $\mathcal{N}_i^*$ is a sampled neighbourhood of fixed size, including the self node.

2.4.4 Message Passing Neural Networks

Message Passing Neural Networks (MPNNs) [33] refer to a generic set of GNNs which may allow for the passing of 'edge feature' information as well as node features. We can define an example as the following:

$$\mathbf{h}_i^{(l+1)} = \sigma\Big(\gamma^l\Big(\mathbf{h}_i^l, \mathbf{m}_i^{(l+1)}\Big)\Big) \tag{2.15}$$

where:

$$\mathbf{m}_i^{(l+1)} = \sigma\Bigg(\sum_{j\in\mathcal{N}_i} \phi^l\Big(\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{e}_{ij}\Big)\Bigg) \tag{2.16}$$

where $\{\mathbf{e}_{ij} \in \mathbb{R}^k | A_{ij} = 1\}$ is a collection of k-dimensional edge features, ϕ^l is the message encoder network and γ^l performs the node updates. Note that I have shown the sum aggregator in Equation 2.16, but this could any could be any permutation-invariant aggregator function. MPNNs are an incredibly flexible formulation for GNNs – in fact, both GCN and GAT can be reformulated as particular instances of MPNNs. In theory, they can have greater modelling capacity than the other GNN formulations mentioned here.

2.5 Self-Attention

The use of self-attention is not limited to a particular data type, and it will appear in different contexts in this dissertation. For example it has already appeared in Section 2.4.2 (Equation 2.13) when describing the GAT model. In simple terms, it describes a method whereby the inputs ('self') are allowed to interact with one another, to discover which are the important parts of the input ('attention').

Figure 2.9 shows a diagram of self-attention. The inputs are considered to be an unordered set $\mathcal{E} = \{\mathbf{e}_1^l, \mathbf{e}_2^l \ldots, \mathbf{e}_n^l\}$. Self-attention will return a set of corresponding outputs whereby each output step can consider all input steps:

$$\mathbf{e}_i^{(l+1)} = \sum_j^N \alpha_{ij} f(\mathbf{e}_j^l) \tag{2.17}$$

where f is a single layer neural network and α_{ij} is the attentional coefficient.

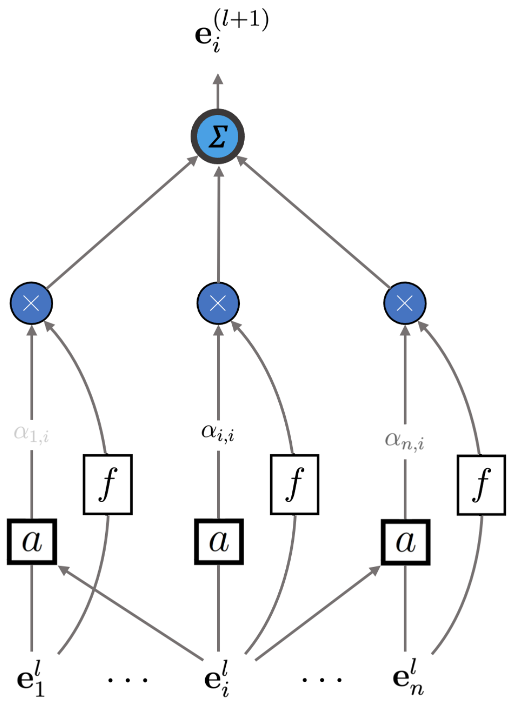


Figure 2.9: A self-attentional layer, where a represents the attention function, and f is a simple neural network.

2.5.1 Transformer

The Transformer [121] is an encoder-decoder model originally intended for Natural Language Processing (NLP), although now it has many other uses. It relies solely on the use of self-attention, where the representation of a sequence (or sentence) is computed by relating different words in the same sequence. Its sole reliance on attention addresses many of the pitfalls of RNN-based models for time series – most importantly that they cannot be parallelised due to their sequential inputs. This suddenly made it possible to train huge models on equally enormous language data sets.

The self-attention components of the Transformer (Figure 2.10) take their input as three separate tensors - *queries* $\mathbf{Q}$, *keys* $\mathbf{K}$ and *values* $\mathbf{V}$. The concept is analogous to retrieval systems in computing. For example, when you search for results on google, the search engine will map the *query* (text in the search bar) to a set of *keys* (title, keywords etc.) associated with candidate web pages in their data set (*values*). They will then present the best matched web pages. In the original Transformer, a 'multi-head attention' block (Figure 2.10) performs the following operations:

$$\psi(\mathbf{Q}, \mathbf{V}, \mathbf{K}) = \left(\Big\Vert_{i=1}^{h} \mathbf{a}_i \right) \mathbf{W}^O \tag{2.18}$$

where ψ represents the multi-head attention function, $||$ denotes concatenation i.e. $||_{i=1}^{A} \mathbf{a}_i =$

$\mathbf{a}_1 \parallel \ldots \parallel \mathbf{a}_A$, $\mathbf{Q} \in \mathbb{R}^{m \times d_{model}}$ represents the queries, $\mathbf{K} \in \mathbb{R}^{n \times d_{model}}$ is the keys, $\mathbf{V} \in \mathbb{R}^{n \times d_{model}}$ is the values, and $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{model}}$ is the output transformation, h (the number of attention heads), d_v, d_k and d_{model} are all hyperparameters, and $\mathbf{a}_i$ is the output of a single attention head, defined as:

$$\mathbf{a}_i = a\Big(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V\Big) \tag{2.19}$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{model} \times d_k}$, and $\mathbf{W}_i^V \in \mathbb{R}^{d_{model} \times d_v}$ are transformations to the queries, keys and values respectively, and a is the attention function, defined as:

$$a(\mathbf{Q}', \mathbf{V}', \mathbf{K}') = \mathrm{softmax}\left(\frac{\mathbf{Q}'\mathbf{K}'^{\top}}{\sqrt{d_k}}\right)\mathbf{V}' \tag{2.20}$$

where $\mathbf{Q}' = \mathbf{Q}\mathbf{W}_i^Q \in \mathbb{R}^{m \times d_k}$, $\mathbf{K}' = \mathbf{K}\mathbf{W}_i^K \in \mathbb{R}^{n \times d_k}$ and $\mathbf{V}' = \mathbf{V}\mathbf{W}_i^V \in \mathbb{R}^{n \times d_k}$.

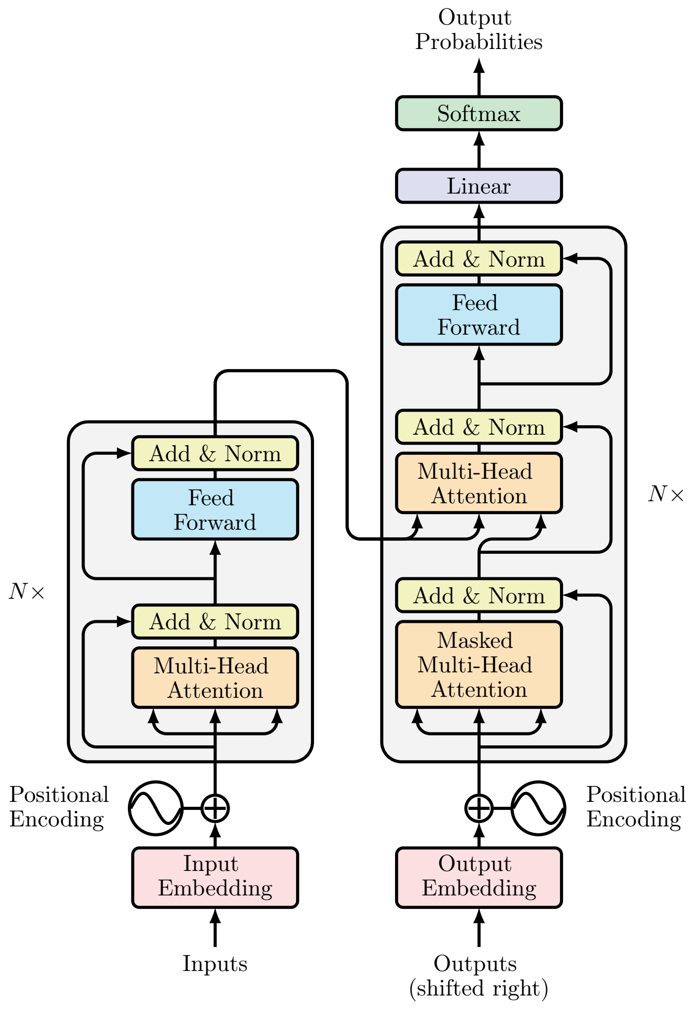


Figure 2.10: The Transformer architecture, taken from the original paper 'Attention is all you need' [121]. The 'Multi-Head Attention' blocks are defined in Equation 2.18. 'Norm' refers to Layer Normalisation [3].

CHAPTER 3

Temporal Pointwise Convolution

This chapter presents my work on a new deep learning model for *time series* representation learning based on the combination of temporal convolution and pointwise (1x1) convolution. I designed this to solve the problem of length of stay prediction on the eICU and MIMIC-IV critical care data sets. The model – which I refer to as Temporal Pointwise Convolution (TPC) – is specifically designed to mitigate common challenges with Electronic Health Records, such as skewness, irregular sampling and missing data.

I started this work following a research internship at Microsoft Research Cambridge in 2019, where I was supervised by Stephanie Hyland. During the internship, we had been experimenting with ways to improve the performance of length of stay prediction models. The work was still in its early stages by the end of the summer, so we agreed to continue collaborating after the internship was over. After my return to the Computer Lab, I was talking to Alex Campbell, a fellow PhD student who was working on convolutional neural networks for functional MRI data, when I had a seed of an idea which evolved into the TPC model presented in this chapter. All aspects of this work are my own, and I had bi-weekly supervision from Stephanie Hyland during this time. My work was published as a conference paper at ACM Conference on Health, Inference, and Learning (CHIL) 2021, under the title "Temporal Pointwise Convolutional Networks for Length of Stay Prediction in the Intensive Care Unit" [95]. A shorter, 4-page extended abstract was accepted as a spotlight talk at the Machine Learning for Health (ML4H) Workshop at NeurIPS 2020, and at the Healthcare Systems, Population Health, and the Role of Health-Tech (HSYS) Workshop at ICML 2020 [94].

3.1 Length of Stay Prediction

In-patient length of stay (LoS) explains approximately 85-90% of inter-patient variation in hospital costs in the United States [88]. Extended length of stay is associated with increased risk of contracting hospital acquired infections [48] and mortality [60]. Hospital bed planning can help to mitigate these risks and improve patient experiences [8]. This is particularly important in the intensive care unit (ICU), which has the highest operational costs in the hospital [20] and a limited supply of specialist staff and resources. Such a system would rely on data from the Electronic Health Record (EHR) system, which contains patient data such as medical histories, diagnoses, medications, medical imaging, laboratory tests and clinical notes.

At present, discharge date estimates are done manually by clinicians, but these rapidly become out-of-date and can be unreliable (for example Mak et al. [67] found that the average error made by clinicians was 3.82 days). Automated systems drawing on the electronic health record (EHR) have the potential to improve forecasting accuracy using state-of-the-art models that can be updated in light of new data. This has efficiency benefits in reducing the administrative burden on clinicians, and the improved accuracy may enable more sophisticated planning strategies e.g. scheduling high-risk elective surgeries on days with more availability [32].

3.2 Key Contributions

In this work, I simulated real-time predictions in retrospective data by updating the patients' remaining ICU length of stay prediction at hourly intervals during their stay using the preceding data from the EHR (similar to Harutyunyan et al. [46]). When designing both the architecture and pre-processing, I focused on mitigating the effects of non-random missingness due to irregular sampling, sparsity, outliers, skew, and other common biases in EHR data. My key contributions are:

1. A new model – Temporal Pointwise Convolution (TPC) – which combines:

    - Temporal convolutional layers [55, 120], which capture causal dependencies across the time domain.
    - Pointwise convolutional layers [64], which compute higher level features from interactions in the feature domain.

    My model significantly outperformed the commonly used Long-Short Term Memory (LSTM) network [50] and the Transformer [121] by margins of 18-68%.

2. I make a case for using the mean-squared logarithmic error (MSLE) loss function to train LoS models. The raw labels have a significant positive skew, whereas the log(LoS) approximates to a Gaussian distribution (Figure 3.1). Using MSLE rather than mean-squared error (MSE), penalises *proportional* rather than absolute errors and stabilises training.

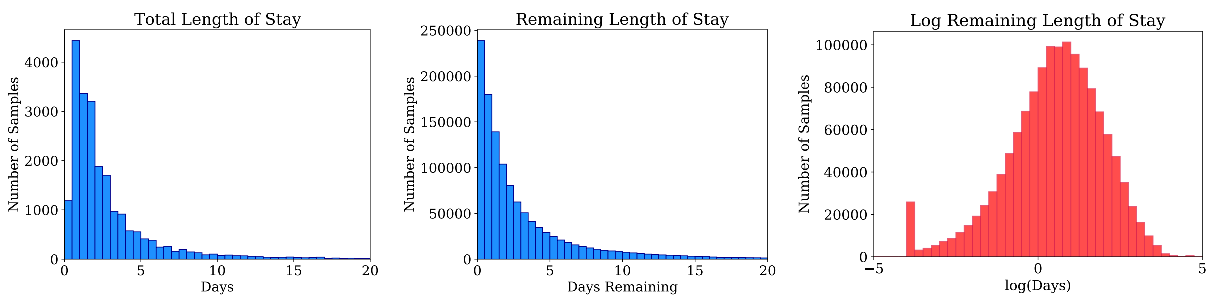


Figure 3.1: Total LoS, remaining LoS, and log(remaining LoS) distributions in the eICU data set. The remaining LoS has a significant positive skew, with mean and median values of 3.47 and 1.67 days respectively. Note that the log(remaining LoS) has a distribution much closer to that of a Gaussian. The spike at -3.87 log(days) is an artefact of the data pre-processing (the patient will never appear to have less than 30 minutes remaining in their stay). The skew in the remaining LoS in MIMIC-IV (not shown) is even more pronounced (5.70 and 2.70 days).

3. By adding in-hospital mortality as a side-task, I demonstrated further performance gains in the multitask setting.

4. I performed several investigations to improve my understanding of the model, including: an extensive ablation study of the model architecture, a post-hoc analysis of feature importances with integrated gradients [112], and a visualisation to show the model reliability as a function of the time since admission and the predicted remaining LoS.

Additionally, I developed a data processing pipeline for the eICU [83] and MIMIC-IV [53] databases that is designed to i) mitigate some of the impact of sparsity (for the diagnoses) and missing data (for time series) in the EHR and ii) extract a wide variety of features semi-automatically such that the approach is generalisable to other EHR databases. My code is available at: `https://github.com/EmmaRocheteau/TPC-LoS-prediction`.

3.3 Related Work

Despite its importance, LoS prediction has received less attention than mortality prediction. This could be due to its difficulty; LoS depends heavily on operational factors and there is considerable positive skew in its distribution (see Figure 3.1). While it has been addressed as a regression problem (optimised using the mean-squared error (MSE) [86, 106]), it is often simplified into binary classification (short vs. long stay) [36, 75, 87], or as a multi-class task [46]. This simplification limits its usefulness as you can only plan over a pre-set timescale, so I choose to focus on the more challenging regression variant.

Traditional machine learning techniques have not performed as well as deep learning on LoS prediction because they often require assumptions about the data, such as linearity, normality, or stationarity which do not hold true [12, 46, 86, 87]. Medical time series data is high-dimensional and non-linear, making it difficult for traditional machine learning methods to capture the complex temporal patterns in the data.

Owing to the centrality of time series in the EHR, the most popular deep learning model for LoS prediction has been the LSTM [46, 87, 106]. This reflects the prominence of LSTMs in other clinical prediction tasks such as predicting in-hospital adverse events including cardiac arrest [118] and acute kidney injury [117], forecasting diagnoses, medications and interventions [13, 65, 113], missing-data imputation [10], and mortality prediction [12, 46, 107]. More recently, the Transformer model [121] been shown to marginally outperform the LSTM on LoS [109] (and it continues to dominate in many other domains [73]). Therefore, the LSTM and the Transformer were chosen as key baselines.

Temporal convolution models have previously been applied to the task of early disease detection using longitudinal lab tests [77, 89, 90], yielding similar results to the LSTM. I highlight two main differences in my work: I introduced a set of pointwise convolutions in parallel, and the temporal convolution filters do not share their parameters between features, allowing the model to optimise processing in spite of heterogeneity in the temporal characteristics. I demonstrated via ablation studies how these design choices contribute substantial improvements to the patient state representation, yielding state-of-the-art results on LoS prediction.

3.4 Methods

3.4.1 Model Overview

I designed my model to extract both temporal trends and inter-feature relationships in order to capture the patient's clinical state. To explain the reasoning, let us consider a patient who is experiencing slowly worsening respiratory symptoms but is otherwise stable. As this patient is unlikely to be weaned from their ventilator in the near future, a clinician might anticipate a long remaining LoS, but how do they come to this conclusion? Intuitively, one of the factors they are evaluating is the trajectory of the patient e.g. they may ask themselves "Is the respiratory rate getting better or deteriorating?". However, they can obtain a better indication of lung function by combining certain features e.g. the PaO_2/FiO_2 ratio, and then looking at how *these* vary over time. A model should therefore be adept at extracting and combining both intra-feature temporal statistics and inter-feature relationships.

Formally, my task was to predict the remaining LoS at regular time points $y_1, \ldots, y_T \in \mathbb{R}_{>0}$ in the patient's ICU stay, up to the discharge time T, using D diagnoses ($\mathbf{d} \in \mathbb{R}^{D\times 1}$), S static features ($\mathbf{s} \in \mathbb{R}^{S\times 1}$), and time series ($\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathbb{R}^{F\times 2}$). In the time series, there are two 'channels' per time series feature for every time point t: F feature values

($\mathbf{x}'_t \in \mathbb{R}^{F\times 1}$), and F corresponding decay indicators ($\mathbf{x}''_t \in \mathbb{R}^{F\times 1}$). The decay indicators tell the model how recently the observation $\mathbf{x}'_t$ was recorded. They are described in detail in Section 3.5.3.3. As we pass through the layers of my model, I repeatedly extract trends and inter-feature relationships using a novel combination of techniques.

3.4.2 Temporal Convolution

Temporal Convolution Networks (TCNs) [55, 120] are a subclass of convolutional neural networks [31] that convolve over the time dimension. They operate on two key principles: the output is the same length as the input, and there can be no leakage of data from the future. I used stacked TCNs to extract *temporal trends* in my data. Unlike most implementations including [90], I *did not share weights across features* i.e. weight sharing is only across time (like in Xception [15]). This is because the features differ sufficiently in their temporal characteristics to warrant specialised processing.

I defined the temporal convolution operation for the i_{th} feature in the n_{th} layer as

$$(f^{n,i} * \mathbf{h}^{n,i})(t) = \sum_{j=1}^{k} f^{n,i}[j]\ \mathbf{h}^{n,i}_{t-d(j-1)} \tag{3.1}$$

where $\mathbf{h}^{n,i}_{1:t} \in \mathbb{R}^{C^n \times t}$ represents the temporal input to layer n until time point t, which contains C^n channels per feature[1]. The convolutional filter $f^{n,i} : \{1, \ldots, k\} \to \mathbb{R}^{Y\times C^n}$ is a tensor of $Y \times C^n \times k$ parameters per feature. It maps C^n input channels into Y output channels while examining k timesteps. The output is therefore $(f^{n,i} * \mathbf{h}^{n,i})(t)^\top \in \mathbb{R}^{1\times Y}$. The dilation factor, d, and kernel size, k, together determine the temporal receptive field or 'timespan' of the filter: $d(k-1)+1$ hours for a single layer. To ensure that the output is always length T, I added left-sided padding of size $d(k-1)$ before every temporal convolution (not shown in Equation 3.1). The $t - d(j-1)$ term ensures that I only look backwards in time. The receptive field can be increased by stacking multiple TCNs (as in Wavenet [120] and ByteNet [55]). I increased the dilation by 1 with each layer i.e. $d = n$.

I concatenated the temporal convolution outputs for each feature, i as follows

$$\underbrace{(f^n * \overbrace{\mathbf{h}^n}^{\text{Temp. In.(1)}})(t)}_{\text{Temp. Out.(2)}} = \Big\Vert_{i=1}^{R^n} (f^{n,i} * \mathbf{h}^{n,i})(t)^\top \tag{3.2}$$

I used $||$ to denote concatenation i.e. $||_{i=1}^{A}\, \mathbf{a}^i = \mathbf{a}^1 \parallel \ldots \parallel \mathbf{a}^A$. In my case, the output dimensions are $R^n \times Y$, where R^n is the number of temporal input features. Throughout this section I label terms with numbers (1), (2) etc. corresponding to objects in Figure 3.4. I recommend following this alongside the equations.

[1]In the first layer, the input $\mathbf{h}^{n,i}_{1:t}$ is the original data $\mathbf{x}^{n,i}_{1:t} \in \mathbb{R}^{2\times t}$, so $C^1 = 2$.

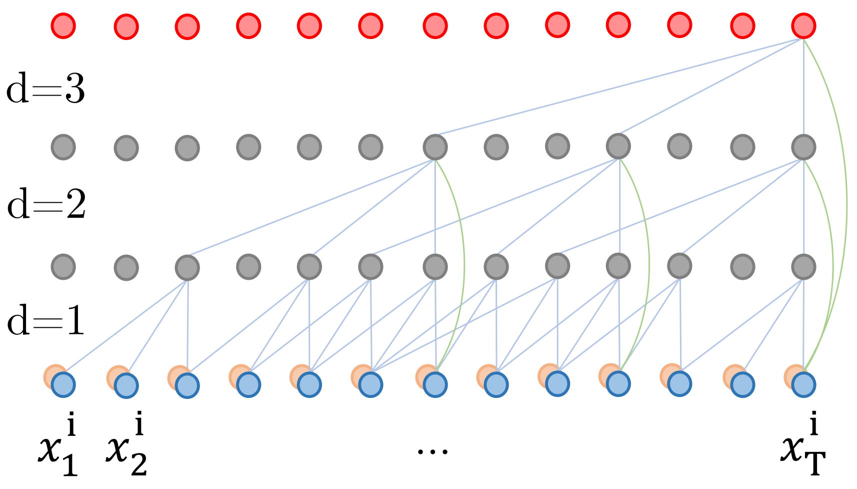


Figure 3.2: Temporal convolution with skip connections (green lines). Each time series, i (blue dots) and their decay indicators (orange dots) are processed with independent parameters.

3.4.3 Pointwise Convolution

Pointwise convolution [64], also referred to as 1×1 convolution, is typically used to reduce the channel dimension when processing images [114]. It can be conceptualised as a fully connected layer, applied separately to each time point (shown diagrammatically in Figure 3.3). As in temporal convolution, the weights are shared across all time points; however, there is no *information transfer* across time. Instead, information is shared across the *features* to obtain Z interaction features[2], $\mathbf{p}_t^n = (\flat(\mathbf{h}_t^n) \parallel \mathbf{s} \parallel \mathbf{x''}_t) \in \mathbb{R}^{P^n \times 1}$, where $P^n = (R^n \times C^n) + F + S$, and $\flat : A^{d_1 \times d_2 \ldots \times d_n} \rightarrow A^{(d_1 \cdot d_2 \ldots \cdot d_n) \times 1}$ is the flatten operation. I defined the pointwise convolution operation in the n_{th} layer as

$$\underbrace{(g^n * \overbrace{\mathbf{p}^n}^{\text{Point. In.(4)}})}_{\text{Point. Out.(5)}}(t) = \sum_{i=1}^{P^n} g^n[i] p_t^{n,i} \tag{3.3}$$

where $g^n : \{1, \ldots, P^n\} \rightarrow \mathbb{R}^{Z \times 1}$ is the pointwise filter, and the resulting convolution produces Z output channels, so $(g^n * \mathbf{p}^n)(t) \in \mathbb{R}^{Z \times 1}$.

3.4.4 Skip Connections

I propagated skip connections [49] to allow each layer to see the original data and the pointwise outputs from previous layers. This helps the network to cope with sparsely sampled data. For example, suppose a particular blood test is taken once per day. In order not to lose temporal resolution, I forward-filled these data (Section 3.5) and convolved with increasingly dilated temporal filters until I found the appropriate width

[2] I used a wider set of features for pointwise convolution, including static features $\mathbf{s}$ and decay indicators $\mathbf{x''}$ i.e. $\mathbf{p}_t^n = (\flat(\mathbf{h}_t^n) \parallel \mathbf{s} \parallel \mathbf{x''}_t) \in \mathbb{R}^{P^n \times 1}$.

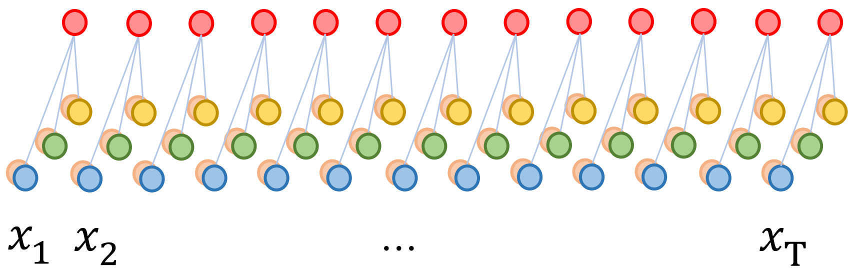


Figure 3.3: Pointwise convolution. There is no information sharing across time, only across features (blue, green, yellow dots).

to capture a useful trend. However, if the smaller filters in previous layers (which did not see any useful trend) have polluted the original data by re-weighting, learning will be harder. Therefore, skip connections provide a consistent anchor to the input. They are concatenated (like in DenseNet [51], and are arranged in the shared-source connection formation [128]) as illustrated in Figure 3.2. The skip connections expand the feature dimension, $R^n = F + Z(n-1)$, to accommodate the pointwise outputs, and also the channel dimension to fit the original data, $C^n = Y + 1$. This is best visualised in Figure 3.4.

3.4.5 Temporal Pointwise Convolution

My model – which I refer to as Temporal Pointwise Convolution (TPC) – combines temporal and pointwise convolution in parallel. Firstly, the temporal output is combined with the skip connections to form $\mathbf{r}_t^n$ (Step 3 in Figure 3.4).

$$\underbrace{\mathbf{r}_t^n}_{(3)} = \underbrace{(f^n * \mathbf{h}_t^n)}_{\text{Temp. Out.(2)}} \parallel \underbrace{\mathbf{x}'_t \parallel \left[\Big\Vert_{n'=1}^{n-1} (g^{n'} * \mathbf{p}_t^{n'}) \right]}_{\text{Skip Connections}} \tag{3.4}$$

$\mathbf{r}_t^n$ is then concatenated with the pointwise output after it has been broadcast $Y+1$ times. I can therefore define the n_{th} TPC layer as

$$\underbrace{\mathbf{h}_t^{(n+1)}}_{\text{TPC Out.(6)}} = \sigma\left(\underbrace{\mathbf{r}_t^n}_{(3)} \parallel \left[\Big\Vert_{i=1}^{Y+1} \underbrace{(g^n * \mathbf{p}_t^n)}_{\text{Point. Out.(5)}} \right] \right) \tag{3.5}$$

where σ represents the ReLU activation function. The full model has N TPC layers stacked sequentially. After N layers, the output $\mathbf{h}_t^N$ is combined with S static features $\mathbf{s} \in \mathbb{R}^{S \times 1}$, and a diagnosis embedding $\mathbf{d}^* \in \mathbb{R}^{D^* \times 1}$ (the output of a diagnosis encoder which takes in D raw diagnoses). Then, two further pointwise convolutions are applied to obtain

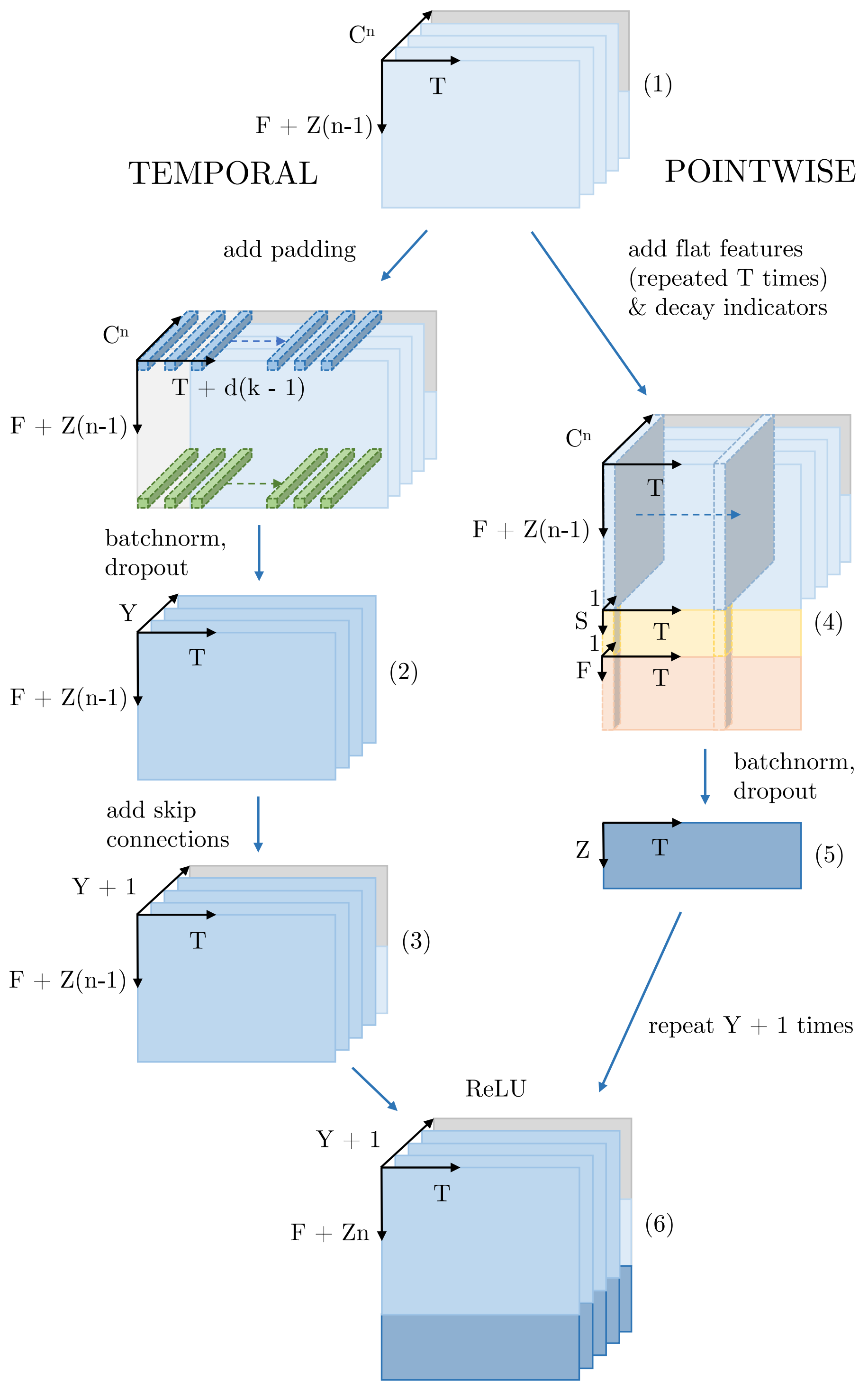


Figure 3.4: The n_{th} TPC layer. F is the number of time series features. T is the length of the time series. C^{n} is the number of temporal convolutional channels per feature in the *previous* TPC layer (in the first layer C^{n} is 1). Z_{n-1} is the cumulative number of pointwise outputs from *all previous* TPC layers (in the first layer Z_{n-1} is 0). Y and Z are the number of temporal channels per feature and pointwise outputs respectively in the *current* TPC layer. Left-sided padding (off-white) is added to the temporal side before each feature is processed independently (indicated by the differently coloured filters). d is the temporal dilation, k is the kernel size. On the pointwise side, S static features (yellow) and F decay indicators (orange) (see Section 3.5.3.3) are added before each convolution. Skip connections containing F original features (grey) plus Z_{n-1} pointwise outputs (light blue) are added. I have ignored the batch dimension for clarity.

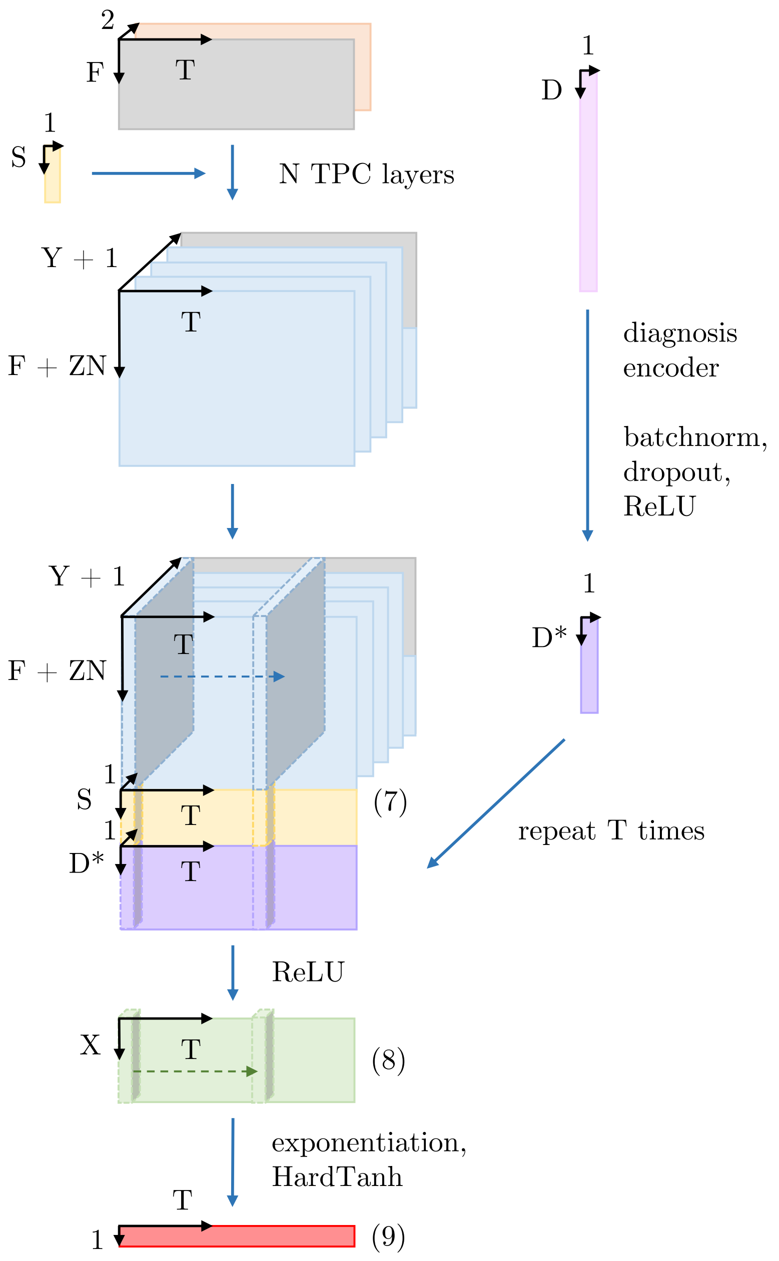


Figure 3.5: Overview of the model. The original time series, $\mathbf{x}'$ (grey) and the decay indicators, $\mathbf{x}''$ (orange) (Section 3.5.3.3) are processed by N TPC layers before being concatenated with a diagnosis embedding $\mathbf{d}^*$ (purple) and static features $\mathbf{s}$ (yellow) along the feature axis. A two-layer pointwise convolution is applied to obtain the final predictions $\hat{\mathbf{y}}$ (red). If a baseline model were used instead of TPC, the time series output dimensions would be M x T, where M is the LSTM hidden size or d_{model} in the Transformer (this is in place of the light blue and grey output in the TPC model).

the final predictions. Formally, these final steps (shown in Figure 3.5) can be written as

$$\underbrace{\hat{y}_t}_{(9)} = \text{HardTanh}\Bigg(\exp\Bigg(g'' * \underbrace{\sigma\Bigg(g' * \overbrace{\Big[\flat(\mathbf{h}_t^N) \parallel \mathbf{s} \parallel \mathbf{d}^*\Big]}^{\text{Final Combined In.(7)}}\Bigg)}_{\text{Penultimate Point. Out.(8)}}\Bigg)\Bigg) \tag{3.6}$$

where $B = R^N \times (Y + 1) + S + D^*$ and the final pointwise filters are $g' : \{1, \ldots, B\} \rightarrow \mathbb{R}^{X\times1}$ and $g'' : \{1, \ldots, X\} \rightarrow \mathbb{R}^{1\times1}$. Note that if a baseline model were to be used instead of TPC, the output dimensions would be H x 1 instead of B x 1, where H is the LSTM hidden size or d_{model} in the Transformer. I apply an exponential function to allow the upstream model to predict log(LoS) instead of LoS. I hypothesised that this could help to circumvent a common issue seen in previous models (e.g. Harutyunyan et al. [46], as they struggle to produce predictions over the full dynamic range of length of stays). Finally, I apply a HardTanh function [43] to clip any predictions that are smaller than 30 minutes or larger than 100 days, which protects against inflated MSLE loss values.

$$\text{HardTanh}(x) = \begin{cases} 100, & \text{if } x > 100, \\ \frac{1}{48}, & \text{if } x < \frac{1}{48}, \\ x, & \text{otherwise.} \end{cases} \tag{3.7}$$

I used batch normalisation [52] and dropout [110] throughout to regularise the model.

3.4.6 Loss Function

The remaining LoS has a positive skew (shown in Figure 3.1) which makes the prediction task more challenging. I addressed this by replacing the commonly-used mean squared error (MSE) loss with mean squared *log* error (MSLE).

$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T}(\log(\hat{y}_t) - \log(y_t))^2 \tag{3.8}$$

MSLE penalises *proportional* errors, which is more reasonable when considering an error of e.g. 5 days in the context of a 2-day stay vs. a 30-day stay. The difference can be seen in Figure 3.6. For bed management purposes it is particularly important not to harshly penalise over-predictions – the model will become overly cautious and regress its predictions towards the mean. This is counter-productive because long stay patients have a disproportionate effect on bed occupancy.

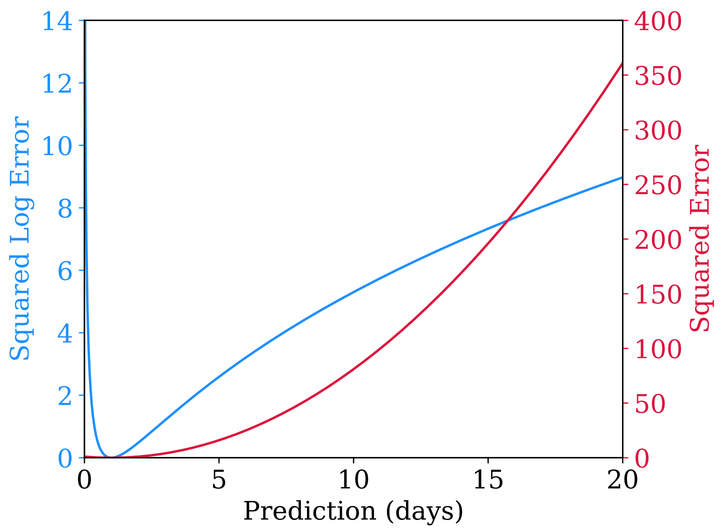


Figure 3.6: The behaviour of squared logarithmic error (blue) and squared error (red) functions when the true LoS is 1 day.

3.5 Data

3.5.1 eICU Database

I used the eICU Collaborative Research Database [83], a multi-centre data set collated from 208 care centres in the United States, available through PhysioNet [35]. It comprises 200,859 patient unit encounters for 139,367 unique patients admitted to ICUs between 2014 and 2015.

I selected all adult patients (>18 years) with an ICU LoS of at least 5 hours and at least one recorded observation, resulting in 118,535 unique patients and 146,671 ICU stays. I selected 87 time series from the following tables: *lab*, *nursecharting*, *respiratorycharting*, *vitalperiodic* and *vitalaperiodic*. To be included, variables had to be present in at least 12.5% of patient stays, or 25% for *lab* variables. The increased cut off for lab variables was due to rarely ordered tests also having fewer time points per patient. Including more variables than needed – especially if they have no trend information – makes the models computationally inefficient as they have to scale to the size of the input.

I extracted diagnoses from the *pasthistory*, *admissiondx* and *diagnoses* tables, and the full set of 17 static features from the *patient*, *apachepatientresult* and *hospital* tables (see Tables 3.2 and A.9 for the full list of features). All pre-processing decisions were made with input from two ICU physicians.

3.5.2 MIMIC-IV Database

I verified my results on a second data set, the Medical Information Mart for Intensive Care (MIMIC-IV v0.4) database [53], a de-identified and publicly available EHR data set from the Beth Israel Deaconess Medical Center containing 69,619 ICU stays from 50,048 patients admitted between 2008 and 2019.

I used the same cohort selection criteria as in eICU to select 69,609 ICU stays from 50,042 patients. I followed the same feature selection process to obtain a short list of 172 time series from the *chartevents* and *labevents*. I manually removed 71 of these from *chartevents* as per expert advice because the variable did not vary over time, or because the distribution was not found to provide useful discrimination between patients (see Table A.10 for the final list of features).

I extracted 12 static features from the *icustays*, *admissions*, *patients* and *chartevents* tables (Table 3.3). I did not extract diagnoses from MIMIC-IV because they are not associated with reliable timestamps.

Table 3.1: Cohort summaries.

	eICU	**MIMIC-IV**
Number of patients	118,535	50,042
Train	82,973	35,028
Validation	17,781	7,507
Test	17,781	7,507
Number of stays	146,671	69,609
Train	102,749	48,848
Validation	22,033	10,497
Test	21,889	10,264
Gender (% male)	54.1%	55.8%
Age in years (mean)	63.1	64.7
LoS in days (mean)	3.01	3.98
LoS in days (median)	1.82	2.06
Remaining LoS (mean)	3.47	5.70
Remaining LoS (median)	1.67	2.70
In-hospital mortality	9.25%	11.4%
Number of input features	104	113
Time series	87	101
Static	17	12

3.5.3 Feature Pre-Processing

3.5.3.1 Static Features

I selected 17 static features from eICU (Table 3.2) and 12 from MIMIC-IV (Table 3.3). Discrete and continuous variables were scaled to the interval [-1, 1], using the 5th and 95th percentiles as the boundaries, and absolute cut offs were placed at [-4, 4]. This was to protect against large or erroneous inputs, while avoiding assumptions about the variable distributions. Binary variables were coded as 1 and 0. Categorical variables were converted to one-hot encodings.

Table 3.2: eICU static features. Age >89, Null Height and Null Weight were added as indicator variables to indicate when the age was more than 89 but has been capped, and when the height or weight were missing and have been imputed with the mean value.

Feature	**Type**	**Source Table**
Gender	Binary	*patient*
Age	Discrete	*patient*
Hour of Admission	Discrete	*patient*
Height	Continuous	*patient*
Weight	Continuous	*patient*
Ethnicity	Categorical	*patient*
Unit Type	Categorical	*patient*
Unit Admit Source	Categorical	*patient*
Unit Visit Number	Categorical	*patient*
Unit Stay Type	Categorical	*patient*
Num Beds Category	Categorical	*hospital*
Region	Categorical	*hospital*
Teaching Status	Binary	*hospital*
Physician Speciality	Categorical	*apachepatientresult*
Age >89	Binary	
Null Height	Binary	
Null Weight	Binary	

3.5.3.2 Diagnoses

Here I only describe pre-processing for eICU since MIMIC-IV did not contain coded diagnoses with appropriate timestamps.

Like many EHRs, diagnosis coding in eICU is hierarchical. At the lowest level they can be quite specific e.g. "neurologic | disorders of vasculature | stroke | hemorrhagic stroke | subarachnoid hemorrhage | with vasospasm". To maintain the hierarchical structure within a flat vector, I assigned separate features to each hierarchical level and use binary encoding. This produces a vector of size 4,436 with an average sparsity of 99.5% (only 0.5% of the data is positive). I applied a 1% prevalence cut-off on all these features to reduce the size of the vector to 293 and the average sparsity to 93.3%. If a disease does not make the cut-off for inclusion, it is still included via any parent classes that do make the cut-off (in the above example I recorded everything up to 'hemorrhagic stroke'). I only included diagnoses that were recorded before the 5th hour in the ICU, to avoid leakage from the future.

Many diagnostic and interventional coding systems are hierarchical in nature e.g. International Classification of Diseases (ICD) [129], Clinical Classifications Software

Table 3.3: MIMIC-IV static features. Age was calculated from the 'intime' field in the *icustays* table and 'anchor year' in the *patients* table.

Feature	**Type**	**Source Table**
Gender	Binary	*patients*
Ethnicity	Categorical	*admissions*
Admission Location	Categorical	*admissions*
Insurance Type	Categorical	*admissions*
First Careunit	Categorical	*icustays*
Hour of Admission	Discrete	*icustays*
Admission Height	Continuous	*chartevents*
Admission Weight	Continuous	*chartevents*
Eyes	Discrete	*chartevents*
Motor	Discrete	*chartevents*
Verbal	Discrete	*chartevents*
Age	Discrete	

(CCS) [27], SNOMED CT [22], and Office of Population Censuses and Surveys (OPCS) Classification of Interventions and Procedures [76], so this technique is generalisable to other coding systems present in EHRs.

3.5.3.3 Time Series

For each admission, I extracted 87 time-varying features from eICU (Table A.9) and 101 from MIMIC-IV (Table A.10) for each hour of the ICU visit, and up to 24 hours before the ICU visit. The variables were processed in the same manner as the static features. In general, the *lab* variables tend to be sparsely and irregularly sampled (this can be seen in Figure 3.7). To help the model cope with this missing data, I re-sampled according to one-hour intervals and forward-filled the data over the gaps. Note that this is more realistic than interpolation as the clinician would only have the most recent value. After forward-filling was complete, any data recorded before the ICU admission was removed.

Decay Indicators With the forward-filling method alone, the model would not know whether a particular data point was a true observation or whether the data had been imputed. This is important because the sampling itself may be informative, for example a deteriorating patient may have more frequent investigations. To mitigate for this, I added 'decay indicators' to specify where the data had been imputed, and if it had, how long it had been since the measurement was taken. The decay was calculated as 0.75^j, where j is the time since the last recording. This is similar in spirit to the masking used by Che et al. [12].

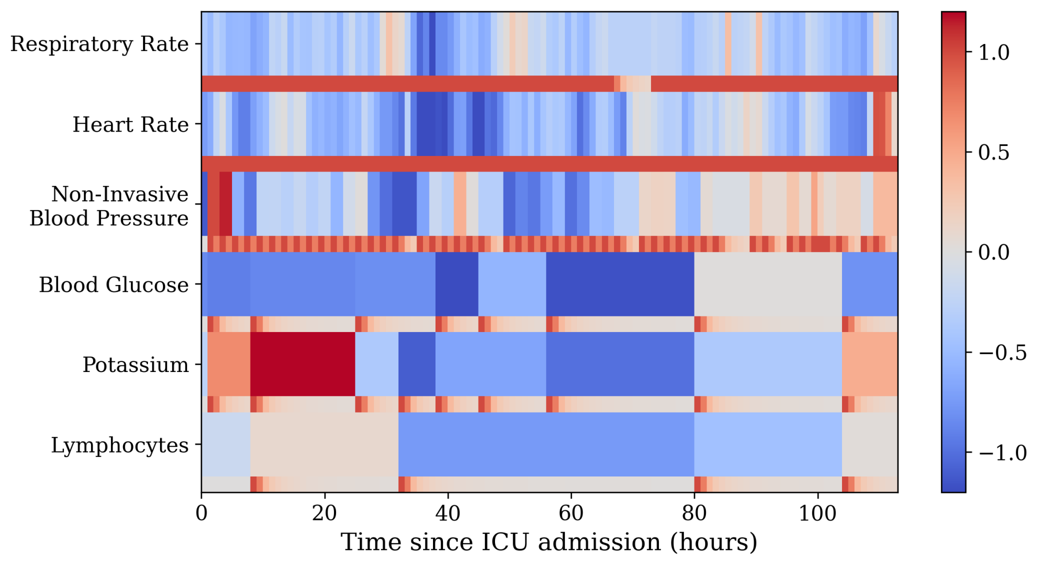


Figure 3.7: Example data from a patient in eICU (after pre-processing). The colour scale indicates the value of the feature, and the narrow bars show the corresponding decay indicators. Blood glucose, potassium and lymphocytes are from the *lab* table and are sparsely sampled. Non-invasive blood pressure is manually recorded by the nurse every 2 hours, while respiratory rate and heart rate are vital signs that are automatically logged.

3.6 Experiments

In this section, I describe the prediction tasks, baseline models and evaluation metrics. As in Harutyunyan et al. [46] the training and test data was fixed upfront – the patients were divided such that 70% were used for training, 15% for validation, and 15% for testing.

3.6.1 Prediction Tasks

3.6.1.1 Remaining Length of Stay

I assigned a remaining LoS target to each hour of the stay, beginning at 5 hours and ending when the patient dies or is discharged. I trained the models to make a prediction every hour of the stay. I only included the first 14 days of any patient's stay to protect against very long batches which would slow down training. This cut-off applies to <5% of patient stays, but it does *not* affect their maximum remaining LoS values.

3.6.1.2 In-Hospital Mortality

I also tested the performance of the models on mortality prediction. Unlike LoS, these labels remain static throughout the patient stay. I used the same training procedure as the LoS task i.e. one prediction each hour. However, to reflect the approach taken by Purushothama et al. [86] and Harutyunyan et al. [46], I only report the mortality performance once per patient (at 24 hours into the stay). This means that the cohort represented in the mortality metrics in Table 3.7 is smaller (16,239 of 21,889 test stays in eICU and 8,320 of 10,264 test stays in MIMIC-IV).

3.6.1.3 Multitask

Previous work has found merit in a multitask approach to patient outcome prediction [46, 106]. I investigated whether I would see a similar benefit in the TPC model. When combining the LoS and mortality losses, I applied a relative weighting to the mortality loss – dictated by a parameter α (which was treated as a hyperparameter). Further information on the hyperparameter search and implementation details is in Appendix A.1.

3.6.2 Baselines

I included the following baselines in my experiments:

'Mean' and 'Median' models (LoS only) These always predict 3.47 and 1.67 days respectively for eICU and 5.70 and 2.70 days for MIMIC-IV (these correspond to the mean and median of the training data). This is to benchmark the level of performance which is achievable 'for free' just by predicting in a reasonable range, and to provide points of reference when setting performance expectations for each data set.

APACHE-IV values (eICU only) These are generated by a risk assessment scoring model which is evaluated only once per patient at 24 hours [136]. Therefore it cannot be compared directly, but I include it *only as a point of reference* for a widely used clinical model. APACHE-IV is only present in the eICU data set.

Standard LSTM My standard LSTM is similar to Harutyunyan et al. [46].

Channel-wise LSTM (CW LSTM) Again similar to Harutyunyan et al. [46], this consists of a set of independent LSTMs that process each feature separately before concatenation (note the similarity with the independent temporal convolutions in the TPC model).

Transformer This model takes advantage of multi-head self-attention. Like the TPC model, it is not constrained to progress one timestep at a time; however, unlike TPC, it is not able to scale its receptive fields or process features independently.

3.6.3 Evaluation Metrics

3.6.3.1 Length of Stay

I reported on 6 LoS metrics: mean absolute deviation (MAD), mean absolute percentage error (MAPE), mean squared error (MSE), mean squared log error (MSLE), coefficient of determination (R^2) and Cohen Kappa Score [16].

I modified the MAPE metric slightly so that very small true LoS values do not produce unbounded MAPE values. I placed a 4 hour lower bound on the divisor i.e.

$$\text{Absolute Percentage Error} = \left| \frac{y_{true} - y_{pred}}{\max\left(y_{true}, \frac{4}{24}\right)} \right| * 100$$

The 4 hour bound produces a compromise between approximating the true metric (the alteration only applies to the cases where $y_{true} < 4$ hours), whilst protecting against wildly magnified errors that would ultimately make the MAPE metric meaningless.

Note that the MAD and MAPE metrics are improved by centering predictions on the median. Whereas MSE and R^2 are bettered by centering predictions around the mean and hence are more affected by the skew. MSLE is arguably the best metric for this task as it is robust to skew and takes all the data into account, indeed, it is the loss function in most experiments, but is less readily-interpretable than some of the other measures. Cohen's linear weighted Kappa Score [16] is intended for ordered classification tasks rather than regression, but it can effectively mitigate for skew if the bins are chosen well. It has previously provided useful insights in Harutyunyan et al. [46], so I use the same LoS bins: 0-1, 1-2, 2-3, 3-4, 4-5, 5-6, 6-7, 7-8, 8-14, and 14+ days. As a classification measure, it will treat everything falling within the same classification bin as equal, so it is fundamentally a coarser measure than the other metrics.

To illustrate the importance of using multiple metrics, consider that the 'Mean' and 'Median' models are in some sense equally poor – neither has learned anything meaningful for our purposes. Nevertheless, the median model is able to better exploit the MAD, MAPE and MSLE metrics, and the mean model fares better with MSE, but the Kappa score betrays them both. A good LoS model will perform well across all of the metrics.

3.6.3.2 In-Hospital Mortality

In the mortality and multitask experiments I reported the Area Under the Receiver Operating Characteristic curve (AUROC) and the Area Under the Precision Recall Curve (AUPRC).

Table 3.4: Performance of the TPC model compared to baseline models. The loss function in all experiments is MSLE. For the first four metrics, lower is better. The error margins are 95% confidence intervals (CIs) calculated over 10 runs. These are not present for the mean, median and APACHE-IV models because they are deterministic. The best results are highlighted in blue. If the result is statistically significant on a t-test then it is indicated with stars (*$p<0.05$, **$p<0.001$). MAD: mean absolute deviation; MAPE: mean absolute percentage error; MSE: mean squared error; MSLE: mean squared logarithmic error; R^2: coefficient of determination, Kappa: Cohen Kappa Score [16]. †Note that the APACHE-IV results (only present in the eICU data set) cannot be compared directly to the other models (explained in Section 3.6.2).

Data	Model	MAD	MAPE	MSE	MSLE	R^2	Kappa
eICU	Mean	3.21	395.7	29.5	2.87	0.00	0.00
	Median	2.76	184.4	32.6	2.15	-0.11	0.00
	APACHE-IV†	2.54	182.1	16.6†	1.10	-0.01	0.20
	LSTM	2.39±0.00	118.2±1.1	26.9±0.1	1.47±0.01	0.09±0.00	0.28±0.00
	CW LSTM	2.37±0.00	114.5±0.4	26.6±0.1	1.43±0.00	0.10±0.00	0.30±0.00
	Transformer	2.36±0.00	114.1±0.6	26.7±0.1	1.43±0.00	0.09±0.00	0.30±0.00
	TPC	**1.78±0.02****	**63.5±4.3****	**21.7±0.5****	**0.70±0.03****	**0.27±0.02****	**0.58±0.01****
MIMIC-IV	Mean	5.24	474.9	77.7	2.80	0.00	0.00
	Median	4.60	216.8	86.8	2.09	-0.12	0.00
	LSTM	3.68±0.02	107.2±3.1	65.7±0.7	1.26±0.01	0.15±0.01	0.43±0.01
	CW LSTM	3.68±0.02	107.0±1.8	66.4±0.6	1.23±0.01	0.15±0.01	0.43±0.00
	Transformer	3.62±0.02	113.8±1.8	63.4±0.5	1.21±0.01	0.18±0.01	0.45±0.00
	TPC	**2.39±0.03****	**47.6±1.4****	**46.3±1.3****	**0.39±0.02****	**0.40±0.02****	**0.78±0.01****

3.7 Results

In this section, I will present analyses of the model in several ways. Firstly, I report the overall performance and make comparisons against a set of baselines. Next, I examine the role of the loss function. Finally, I performed a set of ablation studies to find out which components of the model architecture contribute the most to its success.

3.7.1 TPC Performance on Length of Stay

The TPC model outperforms all of the baseline models on every metric on both data sets (Table 3.4) – particularly those that are more robust to skewness: MAPE, MSLE and Kappa. Discounting APACHE-IV, the best performing *baseline* across both data sets is the Transformer (although the channel-wise LSTM (CW LSTM) is similar on eICU). This is consistent with Harutyunyan et al. [46] (for CW LSTM) and Song et al. [109] (for Transformers), who found small improvements over standard LSTMs.

Performance differences between eICU and MIMIC-IV Although the pattern of results is remarkably similar between eICU and MIMIC-IV, there are notable differences in the magnitudes of the metrics. These differences can be attributed to their LoS distributions – the positive skew is more severe in MIMIC-IV (Table 3.1). This skew has a

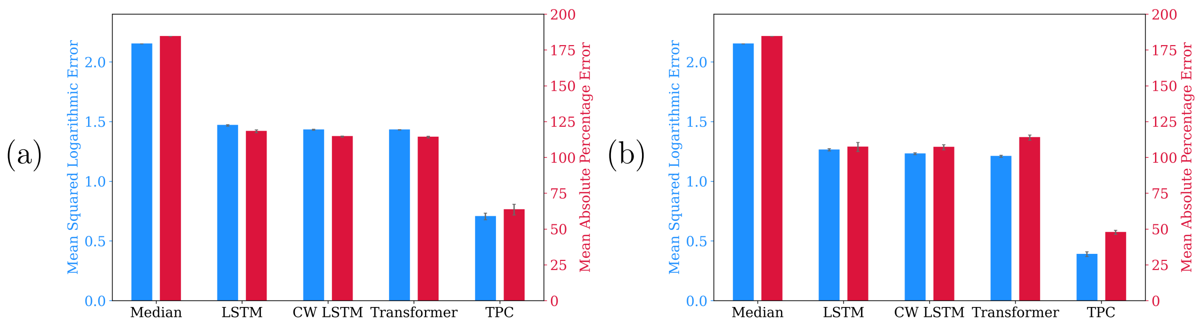


Figure 3.8: A comparison of a subset of results from Table 3.4 showing the MSLE in blue and the MAPE in red. (a) compares the eICU results and (b) shows MIMIC-IV.

disproportionate impact on the *absolute* error, which is captured in the MSE and MAD metrics. Interestingly, the Kappa score is higher in MIMIC-IV because the model can assign the longest stay patients to the >8 day bin, whereas eICU has more medium stay patients in the 3-8 day range which need to be precisely placed. The most comparable results are the MSLE and MAPE metrics, both of which penalise the *proportional* error, making them more robust to shifts in the LoS distribution.

3.7.2 Ablation Studies

Table 3.5: Ablation studies of the TPC model (performed on the eICU data set). Unless otherwise specified, the loss function is MSLE. The first subtable compares the effect of the loss function on the TPC model (see Table A.6 in the Appendix for the MSE results of LSTM, CW LSTM and Transformer). The second shows various TPC ablation studies. Results that are not significantly different from the best result are highlighted in light blue. The TPC (MSLE) result has been repeated in each subtable for ease of comparison. WS: weight sharing; "no skip": no skip connections; "no diag.": no diagnoses, "no decay": no decay indicators.

Model	**MAD**	**MAPE**	**MSE**	**MSLE**	R^2	**Kappa**
TPC (MSLE)	**1.78±0.02****	**63.5±4.3****	21.7±0.5	**0.70±0.03****	**0.27±0.02**	**0.58±0.01****
TPC (MSE)	2.21±0.02	154.3±10.1	**21.6±0.2**	1.80±0.10	**0.27±0.01**	0.47±0.01
TPC	1.78±0.02	**63.5±3.8***	21.8±0.5	**0.71±0.03***	0.26±0.02	0.58±0.01
Point. only	2.68±0.15	137.8±16.4	29.8±2.9	1.60±0.03	-0.01±0.10	0.38±0.01
Temp. only	1.91±0.01	71.2±1.1	23.1±0.2	0.86±0.01	0.22±0.01	0.52±0.01
Temp. only (WS)	2.34±0.01	116.0±1.2	26.5±0.2	1.40±0.01	0.10±0.01	0.31±0.00
TPC (no skip)	1.93±0.01	73.9±1.9	23.0±0.2	0.89±0.01	0.22±0.01	0.51±0.01
TPC (no diag.)	**1.77±0.02**	65.6±4.1	**21.5±0.5**	**0.71±0.03***	**0.27±0.02**	**0.59±0.01**
TPC (no decay)	1.84±0.01	64.5±3.0	22.5±0.3	0.77±0.02	0.24±0.01	0.56±0.01
Point. (no decay)	2.90±0.18	179.1±17.4	34.2±4.6	1.80±0.05	-0.16±0.16	0.33±0.00

To understand the impact of each design choice for the TPC model, I studied the performance under different ablations on the eICU data set. The results of these ablations are reported in Table 3.5 and shown visually in Figure 3.9.

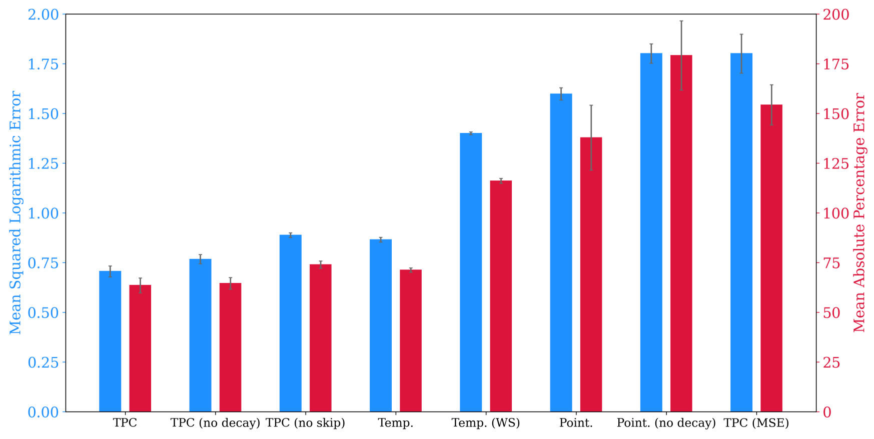


Figure 3.9: A comparison of a subset of results from Table 3.5 (an ablation study on various components of the TPC model). MSLE is shown in blue and MAPE in red.

3.7.2.1 MSLE Loss Function

The first two rows of Table 3.5 show that using the MSLE (rather than MSE) loss function leads to significant improvements in the TPC model, with large performance gains in MAD, MAPE, MSLE and Kappa, while conceding little in terms of MSE and R^2. The MSE results for the other models are in Appendix Table A.6; they show a similar pattern to the TPC model.

3.7.2.2 Model Architecture

The second subtable shows that the temporal-only model is superior to the pointwise-only model, but neither reaches the performance of the TPC model. The temporal-only model performs much better than its weight-sharing variant, which demonstrates the importance of having independent parameters per feature. Note that the temporal-only model with weight sharing is the most similar to the approach taken by Razavian et al. [90], and the results are comparable to the LSTM which is consistent with the results presented in the paper. Removing the skip connections reduces performance by 5-25%. Together the ablation studies demonstrate that the superior performance of the TPC model is the culmination of multiple design decisions.

3.7.2.3 Data

I also tested the models without the diagnoses or decay indicators. Perhaps surprisingly, I found that the exclusion of diagnoses does not seem to harm the model. This could be because the relevant diagnoses for predicting LoS e.g. Acute Respiratory Distress Syndrome (ARDS), are discernible from the time series alone e.g. PaO_2, FiO_2, PEEP etc. The decay indicators contribute a small (but statistically significant) benefit. Their contribution is more obvious in the pointwise-only model where all of the metrics see improvements of 5-23%. This difference is expected since they might reveal some of the temporal structure to the pointwise model e.g. reveal links between up-to-date observations and patient deterioration.

Table 3.6: Performance of the TPC model and its baselines when only some of the time series are included (the static features and diagnoses are still included). The indicator '(labs)' means that only the laboratory tests were included, '(other)' refers to everything except labs: vital signs, nurse observations and machine logged variables. The metric acronyms, colour scheme and confidence interval calculations are described in Table 3.4. The percentage impairment when compared to the complete data set is shown in grey underneath the absolute values. They are calculated with respect to the best value for the metric: 0 for MAD, MAPE, MSE and MSLE, and 1 for R^2 and Kappa. A large percentage impairment means that the model does much better with complete data i.e. it has a high 'percentage gain' from the combination of both data types compared to the ablation case.

Model	**MAD**	**MAPE**	**MSE**	**MSLE**	R^2	**Kappa**
LSTM	**2.39±0.00**	**118.2±1.1**	**26.9±0.1**	**1.47±0.01**	**0.09±0.00**	**0.28±0.00**
LSTM (labs)	2.43±0.00	123.8±1.2	27.3±0.1	1.57±0.00	0.08±0.00	0.27±0.00
	(-1.7%)	(-4.7%)	(-1.5%)	(-6.8%)	(-1.1%)	(-1.4%)
LSTM (other)	2.41±0.00	120.2±0.7	27.3±0.1	1.49±0.00	0.07±0.00	0.27±0.00
	(-0.8%)	(-1.7%)	(-1.5%)	(-1.4%)	(-2.2%)	(-1.4%)
CW LSTM	**2.37±0.00**	**114.5±0.4**	**26.6±0.1**	**1.43±0.00**	**0.10±0.00**	**0.30±0.00**
CW LSTM (labs)	2.42±0.00	124.4±0.7	27.0±0.1	1.57±0.00	0.08±0.00	0.28±0.00
	(-2.1%)	(-8.6%)	(-1.5%)	(-9.8%)	(-2.2%)	(-2.9%)
CW LSTM (other)	2.41±0.00	120.6±0.8	27.1±0.1	1.51±0.00	0.08±0.00	0.29±0.00
	(-1.7%)	(-5.3%)	(-1.9%)	(-5.6%)	(-2.2%)	(-1.4%)
Transformer	**2.36±0.00**	**114.1±0.6**	**26.7±0.1**	**1.43±0.00**	**0.09±0.00**	**0.30±0.00**
Transformer (labs)	2.42±0.00	121.0±0.7	27.3±0.1	1.56±0.00	0.07±0.00	0.27±0.00
	(-2.5%)	(-6.0%)	(-2.2%)	(-9.1%)	(-2.2%)	(-4.3%)
Transformer (other)	2.40±0.00	118.3±0.6	27.3±0.1	1.50±0.00	0.07±0.00	0.27±0.00
	(-1.7%)	(-3.7%)	(-2.2%)	(-4.9%)	(-2.2%)	(-4.3%)
TPC	**1.78±0.02**	**63.5±4.3**	**21.7±0.5**	**0.70±0.03**	**0.27±0.02**	**0.58±0.01**
TPC (labs)	1.85±0.01	72.0±2.2	22.5±0.2	0.81±0.01	0.24±0.01	0.55±0.00
	(-3.9%)	(-13.4%)	(-3.7%)	(-15.7%)	(-4.1%)	(-7.1%)
TPC (other)	**1.81±0.02**	**68.5±4.7**	**21.8±0.3**	0.77±0.03	**0.26±0.01**	**0.57±0.01**
	(-1.7%)	(-7.9%)	(-0.5%)	(-10.0%)	(-1.4%)	(-2.4%)

Finally I performed ablations on the type of time series variable that I included: laboratory tests only (labs), which are infrequently sampled, and all other variables (other) which include vital signs, nurse observations, and automatically recorded variables (e.g. from ventilator machines). This shows how well each model can cope with time series exhibiting different periodicity and sampling frequencies. The results are shown in Table 3.6 and Figure 3.10. The TPC model has the largest percentage gain when the labs and other variables are combined. This suggests that the TPC model is best able to exploit EHR time series with different temporal properties.

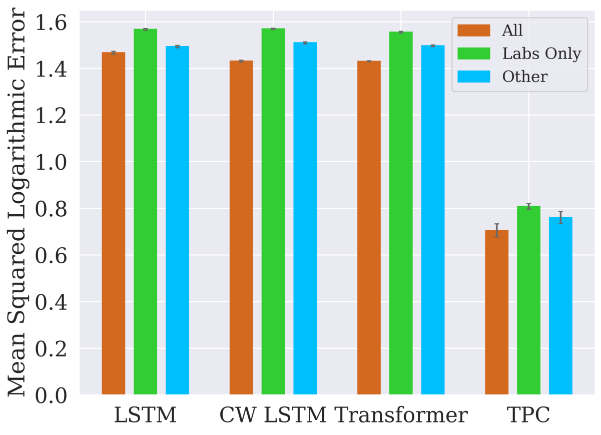


Figure 3.10: A comparison of a subset of MSLE results from Table 3.6.

When examining the results for LSTM and CW LSTM in more detail, we can see that the CW LSTM only has an advantage when the model has to process different types of time series simultaneously. This supports the hypothesis that the CW LSTM is better able to cope when there are varying frequencies in the data, as it can tailor the processing to each. When the inter-feature variability is small (the same type of time series) they perform similarly.

It is unsurprising that the Transformer does better than the LSTM when combining data types, as it can directly skip over large gaps in time to extract a trend in lab values, while simultaneously attending to recent time points for the processing of other variables.

The TPC is the most successful model; its inherent periodic structure helps it to extract useful information from all of the variables. The CW LSTM and Transformer do not have this in their architectures, making the derivation more obscure. The importance of periodicity is discussed in more detail in Section 3.9.

3.7.3 Mortality and Multitask Performance

I investigated adding in-patient mortality as a side-task to improve LoS prediction. Table 3.7 shows the TPC performance both on *single-task* mortality prediction, as well as the multi-task setting. I observed first that TPC achieves good performance on mortality alone. Comparing the impact on LoS forecasting in the multi-task setting, I saw significant improvements on every metric. Multi-task performance for all baselines is reported in Tables A.7 and A.8 in the Appendix, where the multitask training confers a more modest benefit.

Table 3.7: Performance of the TPC model in the multitask setting. I compared the performance of each model on individual tasks (mortality only on the first line; LoS only on the second) to the multitask setting (third line). The performance of the baseline models are in Tables A.7 and A.8.

	In-Hospital Mortality		Length of Stay					
Data	**AUROC**	**AUPRC**	**MAD**	**MAPE**	**MSE**	**MSLE**	R^2	**Kappa**
eICU	0.864±0.001	0.508±0.005	–	–	–	–	–	–
	–	–	1.78±0.02	63.5±3.8	21.8±0.5	0.71±0.03	0.26±0.02	0.58±0.01
	0.865±0.002	**0.523±0.006****	**1.55±0.01****	**46.4±2.6****	**18.7±0.2****	**0.40±0.02****	**0.37±0.01****	**0.70±0.00****
MIMIC-IV	0.905±0.001	0.691±0.006	–	–	–	–	–	–
	–	–	2.39±0.03	47.6±1.4	46.3±1.3	0.39±0.02	0.40±0.02	0.78±0.01
	0.918±0.002**	**0.713±0.007****	**2.28±0.07***	**32.4±1.2****	**42.0±1.2****	**0.19±0.00****	**0.46±0.02****	**0.85±0.00****

3.8 Further Analyses

In this section, I further explored the performance and behaviour of the TPC model for LoS prediction on the eICU data set. I tested its capacity to exploit smaller data sets, explore which features it uses, and provide a visualisation of the reliability of the model. Finally, I simulated the potential use of the model for bed planning.

3.8.1 Training Data Size

The TPC model consistently outperforms the baselines when the training data is small, but I noticed even greater advantage for big data. I tested the TPC, LSTM, CW LSTM, and Transformer models with 6.25%, 12.5%, 25%, 50%, and 100% of the eICU training data. TPC maintains the best test performance on all data sizes. Figure 3.11 shows the effect on MSLE (the full results for all metrics are included in Table A.5).

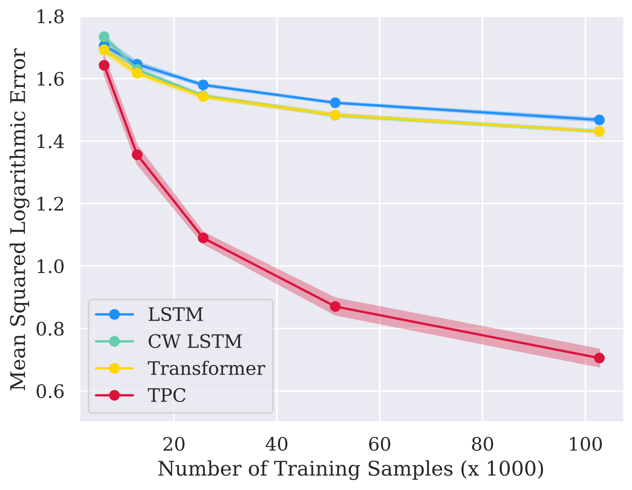


Figure 3.11: The effect of changing the training data size on the LSTM, CW LSTM, Transformer, and TPC model performance on the eICU data. Only the MSLE is shown for clarity, however the other metrics are shown in Table A.5. Note that the performance of the CW LSTM and Transformer models are so similar that the curves are superimposed.

3.8.2 Feature Importance

I used the integrated gradients method [112] to calculate feature attributions for the LoS estimates in the eICU data set. This method computes the importance scores ϕ_i^{IG} by accumulating gradients interpolated between a baseline input $\mathbf{b}$ (intended to represent the absence of data) and the current input $\mathbf{x}$:

$$\phi_i^{IG}(\psi, \mathbf{x}, \mathbf{b}) = \overbrace{(\mathbf{x}_i - \mathbf{b}_i)}^{\text{diff. from baseline}} \times \int_{\alpha=0}^{1} \overbrace{\frac{\delta\psi(\mathbf{b} + \alpha(\mathbf{x} - \mathbf{b}))}{\delta \mathbf{x}_i}}^{\text{acc. local grad.}} d\alpha \tag{3.9}$$

where the TPC model is represented as ψ. I used the mean feature values as my baseline input vector. I took the absolute attribution values when a single LoS prediction is made for each patient at 24 hours. I aggregated by taking the mean along the time dimension and then the patient dimension to obtain Figure 3.12. The background and intuition behind the method is explained clearly by Sturmfels et al. [111].

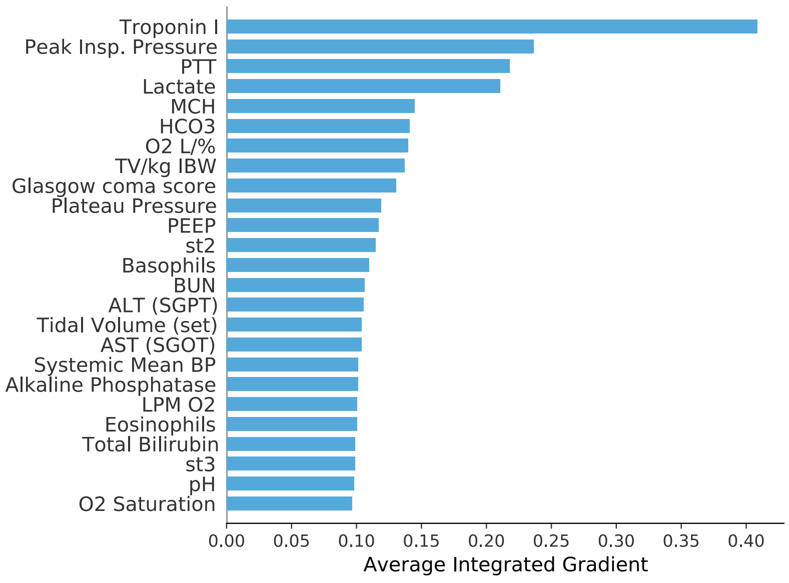


Figure 3.12: Top 25 most important features to the TPC model in the eICU data set.

Analysing Figure 3.12, we can see that the top features are all strong indicators of organ failure: troponin I is a specific biomarker of myocardial infarction; peak inspiratory pressure, O2 L/%, TV/kg IBW, plateau pressure, PEEP and tidal volume indicate mechanical ventilation (on account of respiratory failure); PTT, ALT (SGPT), AST (SGOT) and alkaline phosphatase suggest liver disease; and high BUN and bilirubin levels point towards kidney failure. Additionally we see infection markers such as lactate, basophils and eosinophils which could indicate sepsis. Both multi-organ failure and sepsis are known causes of extended LoS in the ICU [9].

3.8.3 Evaluation by Use-Case

I have reported aggregate performance metrics indicating strong performance of the TPC model for overall LoS forecasting. In this section, I provide further evaluations tailored to two potential users – an individual ICU clinician, and a bed manager for the unit.

3.8.3.1 Individual-Level Reliability

Although aggregate measures of performance are typically reported, these can mask underlying variability in model performance. Such variability can undermine trust or result in unsafe applications [105]. In this section, I think of a clinician who wishes to interpret the prediction of the system for an *individual* patient. I broke down the aggregate performance metrics based on factors which will be readily-available at the time of the prediction. Specifically, I visualised the MAPE (chosen for its interpretability) as a function of the time since admission and the *predicted* remaining LoS.

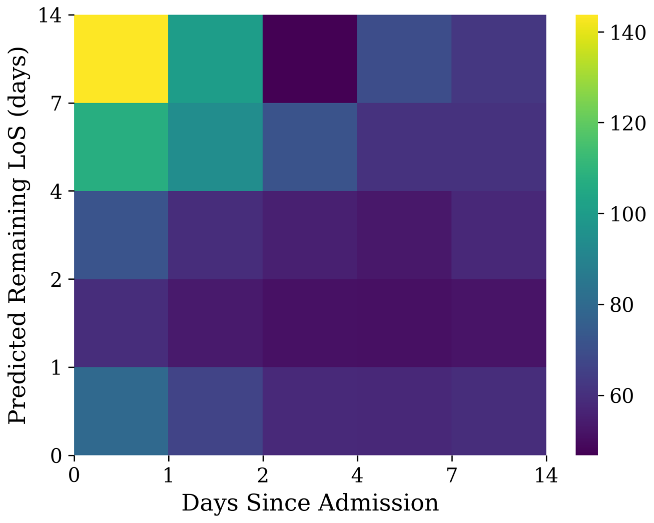


Figure 3.13: Mean absolute percentage error as a function of days since admission and predicted remaining LoS on the eICU data set.

Figure 3.13 shows an example for the TPC model on eICU. We can see that high predicted remaining LoS on the *first* day of a patient's stay can be quite unreliable, with performance rapidly improving over time. Additional investigation revealed these initial predictions to be *under*-predictions, indicating that it is challenging to accurately forecast *very long* LoS for patients on their first day. The long tail of LoS in the data set reflects the abundance of short-stay patients. The model therefore seems to wait for 1-2 days of data to justify a long LoS prediction. The system can therefore be equipped with instructions indicating that a high predicted remaining LoS on the *first* day should not be acted upon. This could complement information provided on a model card [71, 105].

3.8.3.2 ICU-Level Bed Management

From the perspective of a bed manager, *aggregate* performance of the model is important: an over-prediction for one patient could be offset by an under-prediction for another, resulting in the same net bed availability. To investigate this, I performed a simulation study. I ran 500 ICU simulations by randomly selecting 16 examples from the eICU test set to form a 'virtual cohort'. The number 16 was chosen because US hospitals have, on average, 24 ICU beds [125] with an occupancy rate of 68% [44]. Figure 3.14 shows the number of patients remaining in the ICU (of the selected cohort; I do not visualise incoming ICU admissions) using their true remaining LoS (blue). I computed the error (red) between the predictions (green) and true values. The model is well calibrated when predicting patients who are going to stay for at least 1 day. After this, the model tends to under-predict the occupancy by approximately 0.8 patients, corresponding to a small bias towards under-estimating the remaining LoS.

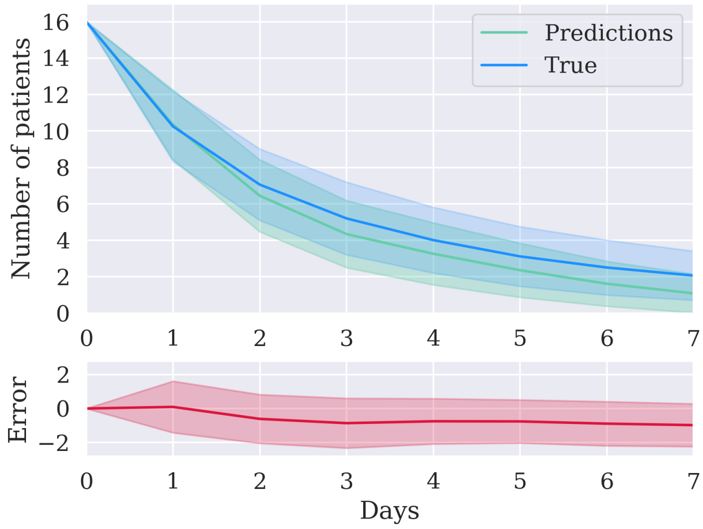


Figure 3.14: ICU simulation. I show the number of patients remaining in the ICU over time from an initial cohort of 16 random eICU patients from 500 simulations. The shaded regions show the standard deviation across the runs. 'Error' is calculated from 'True' minus 'Predictions'.

3.9 Discussion

I have shown that the TPC model outperforms all baseline models in all task settings (LoS, mortality or multitask) on both the eICU and MIMIC-IV data sets. To explain the success of TPC, I start by examining the parallel architectures in the TPC model. Each component has been designed to extract different information: trends from the temporal convolutions and inter-feature relationships from the pointwise convolutions. The eICU ablation studies reveal that the temporal element is more important, but I stress that their contributions are complementary since the best performance is achieved when they are used together.

Next, I highlight that the temporal-only model far outperforms its most direct comparison, the CW LSTM, on all metrics. Theoretically, they are well matched because they both have feature-specific parameters but are restricted from learning cross-feature interactions. To begin to explain this, we can consider how the information flows through the model. The temporal-only model can directly step across large time gaps, whereas the CW LSTM is forced to progress one timestep at a time. This gives the CW LSTM the harder task of remembering information across a noisy EHR with distracting signals of varying frequency. In addition, the temporal-only model can tune its receptive fields for improved processing of each feature thanks to the skip connections (which are not present in the CW LSTM).

Even the hyperparameter search results (Appendix A.1) for the TPC model are interesting, because the best model was found to have 9 layers and a kernel size of 4. This means that the temporal convolutions in the final layer are learning relationships over a receptive field size of at least 28 hours[3]. This is long enough for a single convolution to span over one day to extract useful trends directly from lab results.

The difference in performance between the temporal-only model with and without weight sharing provides strong evidence that assigning independent parameters to each feature is important. Some EHR time series are irregularly and sparsely sampled, and can exhibit considerable variability in the temporal frequencies within the underlying data (evident in Figure 3.7). This presents a challenge for any model, especially if it is constrained to learn one set of parameters to suit all features. The relative success of the CW LSTM over the standard LSTM when processing *disparate* time series – but not similar – also lends weight to this theory.

However, the assignment of independent parameters to each feature does not explain all the successes of TPC e.g. the TPC model can process disparate time series and gain more marginal performance than the CW LSTM (Table 3.6). We need to consider that *periodicity* is a key property of EHR data – this is true in both the sampling patterns and in the underlying biology e.g. medication schedules, sleep cycles, meals etc. The temporal component of the TPC model is the only architecture with an inherent periodic structure (from the stacked temporal filters) which makes it much easier to learn EHR trends. By comparison, a single attention head in the Transformer model does not look at time points a fixed distance apart, but can take an arbitrary form. This is a strength for natural language processing, given the variety of sentence structures possible, but it does not help

[3]The receptive field for a single layer can be calculated with $d(k-1)+1$, where d is the dilation and k is the kernel size. For the final TPC layer this is $9(4-1)+1 = 28$ hours, since $d = n$. In reality, the receptive field can capture much longer than 28 hours because the temporal convolutions are stacked on top of convolutions that can also look backwards in time and so on.

the Transformer to process EHRs.

Additionally, I have shown that the TPC model outperforms baselines on in-hospital mortality both as a standalone task and in combination with LoS. The performance on both mortality and LoS is significantly better in the multitask setting (this is consistent with past works [46, 106]) because multitask learning helps to regularise the model and reduce the chance of overfitting [98]. Adding further tasks may be a valid strategy to improve LoS performance.

Finally, I reiterate that using MSLE loss instead of MSE greatly mitigates for positive skew in the LoS task, and this benefit is not model-specific (all of the baselines perform better with MSLE – see Table A.6). This demonstrates that careful consideration of the task – as well as the data and model – is an important step towards building useful tools in healthcare.

3.10 Summary

I proposed and evaluated a new deep learning architecture, which I call 'Temporal Pointwise Convolution' (TPC). TPC combines temporal convolutional layers with pointwise convolutions to extract temporal and inter-feature information. I have shown that the TPC model is well-equipped to analyse EHR time series containing missingness, differing frequencies and sparse sampling. I believe that the following four aspects contribute the most to its success:

1. The combination of two complementary architectures that are able to extract different features, both of which are important.

2. The ability to step over large time gaps.

3. The capacity to specialise processing to each feature (including the freedom to select the receptive field size for each).

4. The rigid spacing of the temporal filters, making it easy to derive trends.

From a clinical perspective, I have contributed to the advancement of LoS prediction models, a prerequisite for automated bed management tools. Improving the practice of bed management promises cost reduction [44] and better resource allocation [69] worldwide. From a computational perspective, I have provided key insights for retrospective EHR studies, particularly where LSTMs are the currently model of choice. In the broader context of machine learning for healthcare I have demonstrated that careful consideration of the complexities of health data is necessary to gain state-of-the-art performance.

CHAPTER 4

Graph Representation Learning

This chapter presents my work on using graph representations to help exploit *sparse data* in the EHR. In the earlier phases of my PhD, I noticed that work on predicting patient outcomes in the ICU focuses heavily on the physiological time series and basic demographics (and I did the same in my last chapter). This is probably because these data types already have well-established architectures for extracting good representations e.g. the LSTM. However, in the context of EHRs, it could mean that models are missing out on rich sources of additional information such as diagnoses and medications, which are usually sparse. When I *did* see sparse data used, they were usually heavily condensed and fused in the final stages of a model. Therefore these models were not given a chance to use the diagnosis information to help process the time series and vice versa. Nor would they be able to learn from rarer disease patterns because there were too few training examples.

Therefore, I began to think about how to solve the problem of exploiting sparse data in the EHR. I initially discussed many of these ideas with Petar Veličković, who was keen to be involved in the project. I had settled on a preliminary version of this idea – using graph representation learning for sparse data processing in the EHR – by the time I attended NeurIPS in 2018. During that conference, Petar connected me with another PhD student called Catherine Tong, who was studying at Oxford. She decided to come onboard the project and from that point we worked together as joint first co-authors on the work presented in this chapter. We split the work as follows: I was responsible for the data pre-processing (Section 4.5) and the construction of the diagnosis graph (Section 4.4.1), whereas Catherine was responsible for implementing the LSTM-GNN (Section 4.4.2). The experimentation was truly a collaborative effort, meaning that we both conducted experiments and closely discussed the model development and results throughout. This chapter has been adapted from text written by me in the corresponding paper, with the notable exception of Section 4.4.2 which was originally written by Catherine, but I have made modifications for the purposes of this dissertation.

4.1 The Difficulty of Using Sparse Data

The past decade has seen growing interest in patient outcome prediction, particularly in the Intensive Care Unit (ICU). This is following the increased availability of Electronic Health Records (EHRs) and the drive to minimise preventable deaths through the use of early warning systems [72, 108]. Most prior works have focused on a small subset of features in the EHR [82] – namely, the physiological time series data (especially following the publication of pre-processing pipelines e.g. Harutyunyan et al. [46]). This is problematic because the resulting models can miss clinically important information, leading to poorer clinical outcomes [87].

Among the frequently overlooked (but very informative) data are the diagnoses, medications and surgical procedures. They are difficult to use for two reasons:

1. The large number of features makes distinguishing relevant comorbidity patterns combinatorially difficult[1].

2. The model does not have enough data on rare diseases.

The common approach has been to throw away the long tail of the distribution (shown in Figure 4.1) during pre-processing and concatenate the remaining features (often via an encoder network) to the main part of the model at a late stage. Unfortunately this approach is always a compromise between the difficulty of the modelling task and the amount of valuable data that is thrown away.

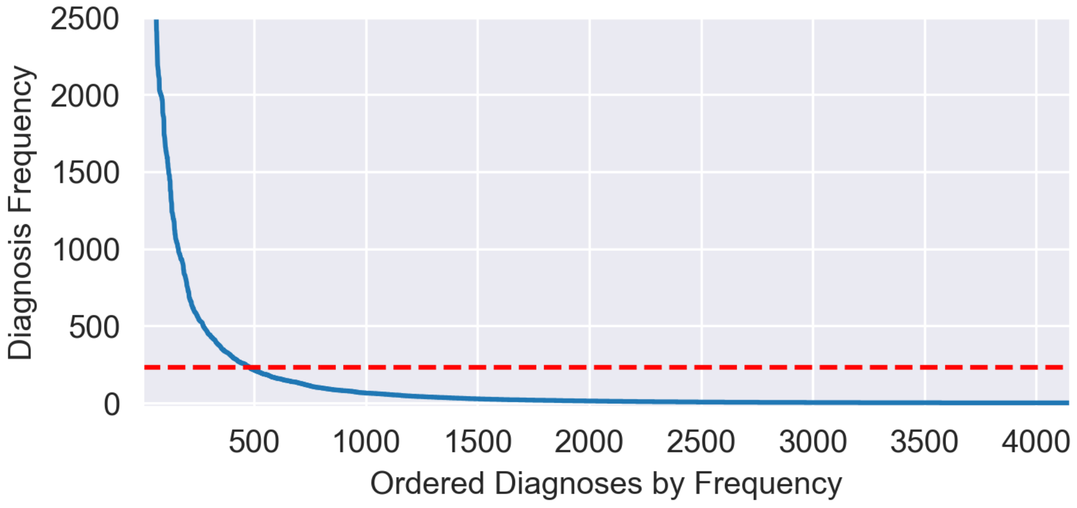


Figure 4.1: The distribution of diagnosis frequency in our data is positively skewed. The mean number of samples per diagnosis is 229 (shown in red) which is not enough for most deep learning models to learn from. Note that the y axis has been truncated (the maximum value is in fact 79,778).

[1]The average patient in eICU has only 9 recorded diagnoses, but there are 4,172 distinct diagnoses in our cohort.

4.2 Key Contributions

I wanted to improve on this approach toward sparse information in the EHR, taking diagnoses as an example. When designing the model, I took inspiration from clinicians, who tend to rely on their past experience of treating similar patients when making clinical judgements. I captured this similarity concept by constructing a *patient graph* where the nodes are patients and the edges express relatedness in diagnoses. We exploited this information with a hybrid architecture consisting of a Long Short-Term Memory (LSTM) [50] network for extracting temporal features, composed with a Graph Neural Network (GNN) [38, 101] for extracting the patient neighbourhood information. This represents a novel application of GNNs in healthcare. We demonstrated that LSTM-GNNs outperform the LSTM-only baseline on the LoS prediction task when using data from the first 24 hours of the ICU stay. More generally, our results indicated that exploiting information from neighbouring patient cases using graph neural networks is a promising research direction, yielding tangible returns in supervised learning performance on Electronic Health Records.

While I focused on diagnosis information, our method can easily be extended to other sparse medical data such as shared medications. Our code can be found at `https://github.com/EmmaRocheteau/eICU-GNN-LSTM`.

4.3 Related Work

Our work is motivated by the following areas of related work:

Graph Neural Networks GNNs are a subclass of neural networks which operate on graph-structured data as input. The general principle is to apply a transformation function to each node representation in the graph, before aggregating information between neighbouring nodes. Different GNNs vary in their node transformation and neighbourhood aggregation functions [130]. In our work, we selected four popular GNNs to model the similarity relationships between patients: Graph Convolutional Networks (GCN) [58], Graph Attention Networks (GAT) [123], GraphSAGE [45], and Message Passing Neural Networks (MPNN) [33]. A full background and explanation of all these GNN architectures can be found in Section 2.4.

Recurrent Neural Networks RNNs (particularly LSTMs) have so far been the most popular model for patient outcome prediction from time series, and they have achieved state-of-the-art results on the MIMIC-III and eICU data sets [46, 87, 106]. We therefore selected an LSTM (very similar to that used in Harutyunyan et al. [46]) to model the time series component. Again, the relevant background for LSTMs can be found in Section 2.2.

Combination Models Combining sequential modelling with GNNs has been explored by several works outside of the healthcare domain in recent years [39, 63, 79, 132]. However, since we assumed the graph to be static in my case (a reasonable assumption since patient diagnoses do not vary much during a typical ICU stay), this calls for a simpler modelling approach than these works.

Graphical Representation of Clinical Data This is a young and exciting research domain [103] that has so far focused on injecting medical knowledge in the form of knowledge graphs, or structuring the EHR itself as a graph, as in Choi et al. [14] (note that these applications do not employ any *inter-patient* data sharing). The only example of a patient graph appearing in the literature before ours was Malone et al. [68] who solved the task of missing data imputation using embedding propagation [26]. This was done as a separate step (i.e. their approach was not end-to-end) before using logistic regression and ridge regression as downstream prediction models.

4.4 Methods

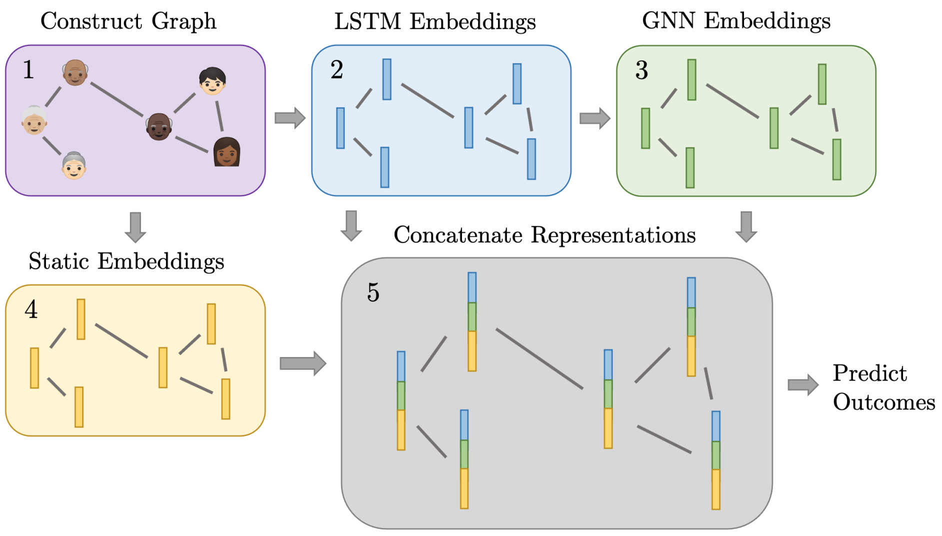


Figure 4.2: Approach Overview. First, I constructed a patient graph, which becomes the input to an end-to-end LSTM-GNN model. Through the LSTM-GNN, each node's temporal features are encoded by a temporal encoder followed by a graph encoder, and the static features are encoded separately. Finally, these are concatenated and passed to a fully-connected layer for prediction.

Figure 4.2 gives an overview of our approach. I started with a set of static and temporal features for each patient. My first step (top-left in Figure 4.2) was to define a patient graph construction, $\mathcal{G}$, where related patients (nodes) are connected by edges. Next, we passed the patient graph as input to the LSTM-GNN, which was trained end-to-end. The LSTM-GNN produced three types of embeddings: the LSTM, GNN and static embeddings, which were concatenated and passed to a fully-connected layer to obtain per-node predictions. In the following, I provide the technical details of both the graph construction and LSTM-GNN training procedures.

4.4.1 Diagnosis Graph Construction

I started by assigning a pairwise similarity score between all patients. First, I transformed the diagnoses into a multi-hot vector for each patient (Section 4.5.2), resulting in a binary diagnosis matrix $\mathcal{D} \in \mathbb{R}^{N \times m}$ where N is the number of patients and m is the number of unique diagnoses. The similarity score $\mathcal{M}_{ij}$ between nodes i and j was defined as

$$\mathcal{M}_{ij} = a \overbrace{\sum_{\mu=1}^{m} \Big(\mathcal{D}_{i\mu}\mathcal{D}_{j\mu}(d_{\mu}^{-1} + c)\Big)}^{\text{Shared Diagnoses}} - \overbrace{\sum_{\mu=1}^{m} \big(\mathcal{D}_{i\mu} + \mathcal{D}_{j\mu}\big)}^{\text{All Diagnoses}} \tag{4.1}$$

where d_{μ} is the frequency of diagnosis μ, and a and c are tunable constants. The first term positively rewards shared diagnoses. It includes an inverse of the frequency term (d_{μ}^{-1}) which increases the weighting of rare diagnoses. The reason for this is because the model may be able to learn the key features of common diseases using the traditional method, but for rare diseases this will be the only representation. The second term penalises the total number of diagnoses – this is to prevent patients with many diagnoses becoming ‘hubs’ of high connectivity, attracting imprecise matches with several non-shared diagnoses. Note that the rationale for these choices are explained in much more detail in the dedicated paragraph at the end of this subsection.

I examined $\mathcal{M}$ under a k-Nearest Neighbour (k-NN) scheme to establish k edges per node. The parameters a, c and k were treated as hyperparameters ($c = 0.001$, $a = 5$ and $k = 3$ in the final model). I observed that as a is increased, the model tends to favour forming ‘hubs’ with common comorbidities. Similarly, when c is increased, it takes the focus away from rare diseases (because c corresponds to a minimum weighting that any disease will always contribute to the similarity score). Note that this method always inserts self-loops into the graph. This will mean that the GNN can always propagate the patient’s own features so that they can be used for the downstream prediction tasks.

I also experimented with alternative graph construction methods, such as applying a score threshold for edges (more akin to Malone et al. [68]), or using BERT [25] (a type of Transformer [121], explained in Section 2.5.1) to encode diagnosis texts prior to calculating the similarity. Empirically I found the method presented in Equation 4.1 to work best through manual inspection of the resultant graph and preliminary experimentation.

Why is it desirable to emphasise rare diagnoses? If we take the example of a classic metabolic syndrome, where multiple diagnoses tend to occur together e.g. type 2 diabetes, hypertension, obesity, non-alcoholic fatty liver disease (NAFLD), polycystic ovary syndrome (PCOS) etc. If all diagnoses were treated equally, patients sharing a metabolic syndrome would automatically match by default, since there are many diagnosis codes in common (Figure 4.3). Any additional rare diagnoses that do not fit the pattern would be ignored:

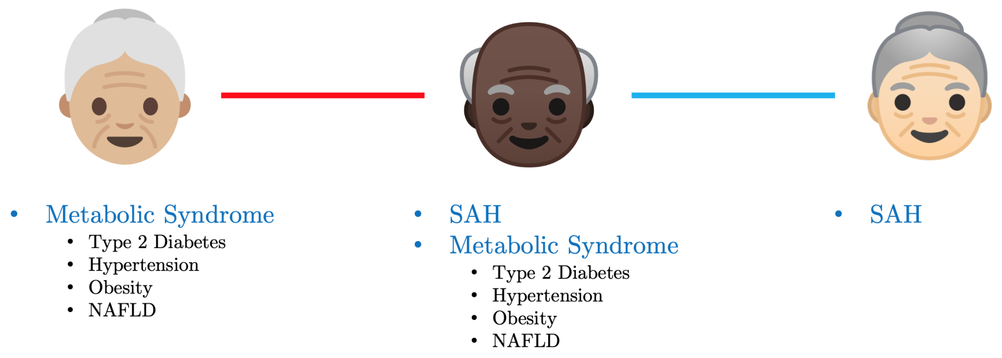
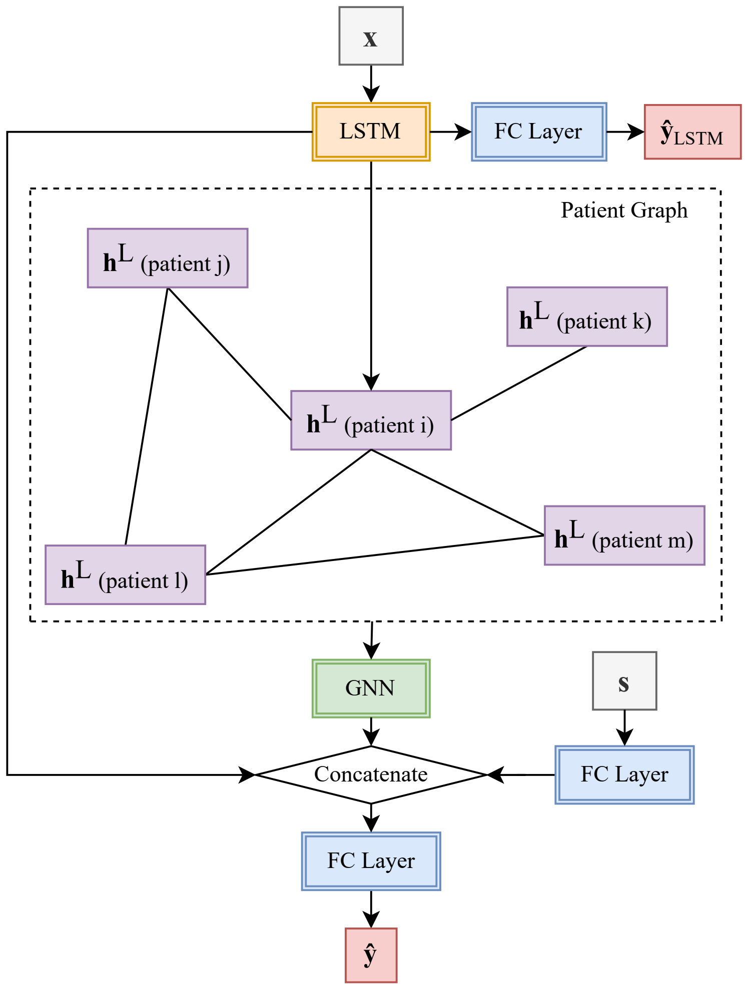


Figure 4.3: A graph which treats diagnosis codes equally irrespective of prevalence would match the two patients who share the metabolic syndrome (red edge). However, in this case the rare diagnosis of sub-arachnoid haemorrhage (SAH) is far more predictive of poor outcomes in the ICU. Therefore, a graph which up-regulates rare diagnoses (blue edge) will not miss these potentially important associations.

This is missing a major opportunity to add value when using the patient graph approach. The metabolic syndrome is sufficiently common that the model could learn the recognise the phenotype and its association to patient outcomes using the time series and static data alone. By contrast this is not possible for the rare diagnoses because the model does not have sufficient training examples to reliably learn a representation. Therefore, the association needs to be presented in a more direct way i.e. by enabling explicit access to other examples of relevant rare diseases via a GNN at the time of prediction.

4.4.2 LSTM-GNNs

Having constructed the patient graph, we framed the patient outcome prediction problem as a node prediction task. We use LSTM-GNN, a hybrid model consisting of temporal and graph encoding components (summarised in Figure 4.4). We assumed the input of LSTM-GNN to be a patient graph $\mathcal{G}$, with each node i having time series $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathbb{R}^{F \times 1}$ and static features $\mathbf{s} \in \mathbb{R}^{S \times 1}$ (this includes a diagnosis vector and other variables e.g. age and gender). F is the number of time series features, S is the number of static features and T is the time to discharge. We can describe a forward pass through the network as follows:



x
LSTM
FC Layer
$\hat{\mathbf{y}}_{\text{LSTM}}$
Patient Graph
$\mathbf{h}^L$ (patient j)
$\mathbf{h}^L$ (patient k)
$\mathbf{h}^L$ (patient i)
$\mathbf{h}^L$ (patient m)
$\mathbf{h}^L$ (patient l)
GNN
s
FC Layer
Concatenate
FC Layer
$\hat{\mathbf{y}}$

Figure 4.4: Our LSTM-GNN architecture. The LSTM extracts temporal features from time series data. These patient-level features are then propagated within local neighbourhoods by the GNN. We concatenated hidden vectors from the temporal, graph and the static features to make the prediction.

The time series $\mathbf{x}$ are first passed through a Bidirectional LSTM (BiLSTM) [40], which outputs a sequence of hidden state vectors $\mathbf{h} \in \mathbb{R}^{M \times T}$ in the forward and reverse directions, where M is the LSTM hidden size. The vectors corresponding to the last timestep are concatenated to produce $\mathbf{h}^L \in \mathbb{R}^{2M \times 1}$ the LSTM temporal embedding for patient i.

Next, the GNN component propagates each node's temporal embedding within its neighbourhood. This function varies between GNNs, however, the aim is always to apply a local smoothing. That is, the features $\mathbf{h}^L$ are re-weighted with the feature vectors in its local neighbourhood to produce the new node representation $\mathbf{h}^N$.

Meanwhile, we passed the static input $\mathbf{s}$ through a fully-connected layer to compute $\mathbf{h}^S$, before concatenating our learnt representations together, $\mathbf{h}^C = (\mathbf{h}^L || \mathbf{h}^N || \mathbf{h}^S)$. Finally, $\mathbf{h}^C$ is passed through a fully-connected layer to obtain a prediction $\hat{\mathbf{y}}$.

We trained the LSTM-GNN in an end-to-end and scalable fashion. To allow for mini-batch training of the LSTM, we adopted a neighbourhood sampling procedure proposed by Hamilton et al. [45]. With each iteration, we uniformly sampled a fixed-size set of neighbours in the graph, which fixes computation and time costs per iteration, thus making the LSTM-GNN scalable to large patient graphs. We took an inductive learning approach. During training, we only sampled nodes and their neighbourhood from the training set. During testing, we sampled neighbours from the entire data set but we only evaluated performance on the test nodes.

Enpirically we noticed that the LSTM performance could degrade with the addition of a GNN. To encourage learning from both components, we defined the loss function as:

$$\mathcal{L} = \mathcal{L}_{\text{LSTM-GNN}} + \alpha \mathcal{L}_{\text{LSTM}} \tag{4.2}$$

where $\mathcal{L}_{\text{LSTM-GNN}}$ is the loss on the full model prediction $\hat{\mathbf{y}}$, $\mathcal{L}_{\text{LSTM}}$ is the loss on the prediction made by the LSTM component $\hat{\mathbf{y}}_{\text{LSTM}}$ (computed by passing $\mathbf{h}^L$ through a distinct fully-connected layer), and α is treated as a hyperparameter. For the mortality task, the loss function was binary crossentropy, whereas for LoS it was the squared logarithmic error as this was found to mitigate for positive skew in the previous chapter (Section 3.7.2.1).

4.5 Data

As in the previous chapter, I used the eICU Collaborative Research Database [83], a multi-centre data set collated from 208 hospitals in the United States (for further details see Section 3.5.1). I selected static features, time series and diagnoses from 89,123 adult patients ($>$18 years) with an ICU LoS of at least 24 hours and at least one recorded observation. If the patient had multiple admissions I selected one at random. This was essential because otherwise the patient could be linked to a neighbour which corresponds to a previous or future self in the graph. The data set was divided at the patient level such that 70%, 15% and 15% were used for training, validation and testing respectively. A summary of the cohort can be found in Table 4.1.

4.5.1 Static Features

I initially extracted 20 static features (shown in Table 4.2). As in Section 3.5.3.1, discrete and continuous variables were scaled to the interval [-1, 1], using the 5th and 95th percentiles as the boundaries, and absolute cut offs were placed at [-4, 4]. Binary variables were coded as 1 and 0. Categorical variables were converted to one-hot encodings.

Table 4.1: eICU cohort summary.

Number of patients	89,123
Train	62,385
Validation	13,369
Test	13,369
Gender (% male)	54.4%
Age in years (mean)	63.6
LoS in days (mean)	3.69
LoS in days (median)	2.28
In-hospital mortality	9.52%
Number of node features	38
Time series	18
Static	20

4.5.2 Diagnoses

I extracted diagnoses in a similar way to the previous chapter (Section 3.5.3.2), but with a more generous prevalence cut-off of 0.5%. To recap, the diagnosis coding in eICU is hierarchical e.g. "neurologic | disorders of vasculature | stroke | hemorrhagic stroke | subarachnoid hemorrhage | with vasospasm". To preserve the hierarchical structure, I assigned separate features to each class level i.e. 'neurologic' and 'disorders of vasculature' etc. are their own features. I then transformed the data into multi-hot encodings with each position indicating '0' or '1' depending on whether a particular diagnosis was present:

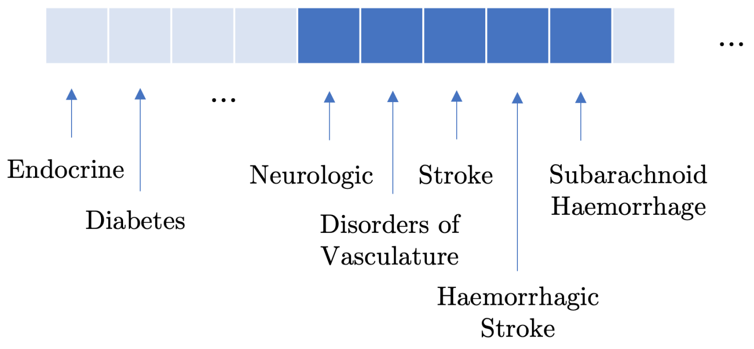


Figure 4.5: An example showing an EHR diagnosis vector.

This produces a vector of size 4,436 with an average sparsity of 99.5%. Note that if a disease does not make this threshold, it is still included via any parent classes that do qualify (e.g. in the above example we retain everything up to 'subarachnoid hemorrhage'). To prevent leaking from future data into the predictions, I only included diagnoses that were recorded before the 24th hour in the ICU.

Table 4.2: Non-time varying features. Age >89, Null Height and Null Weight were added as indicator variables to indicate when the age was more than 89 but has been capped, and when the height or weight were missing and have been imputed with the mean value.

Feature	**Type**	**Source Table**
Gender	Binary	*patient*
Age	Discrete	*patient*
Hour of Admission	Discrete	*patient*
Height	Continuous	*patient*
Weight	Continuous	*patient*
Ethnicity	Categorical	*patient*
Unit Type	Categorical	*patient*
Unit Admit Source	Categorical	*patient*
Unit Stay Type	Categorical	*patient*
Physician Speciality	Categorical	*apachepatientresult*
Eyes	Discrete	*apacheapsvar*
Motor	Discrete	*apacheapsvar*
Verbal	Discrete	*apacheapsvar*
Meds	Discrete	*apacheapsvar*
Intubated	Binary	*apacheapsvar*
Ventilated	Binary	*apacheapsvar*
Dialysis	Binary	*apacheapsvar*
Age >89	Binary	
Null Height	Binary	
Null Weight	Binary	

4.5.3 Time Series

For each admission, 18 time-varying features (Table 4.3) were extracted from each hour of the stay, and up to 24 hours before. The variables were processed in the same manner as the static features. In general, the sampling was irregular, so the data was re-sampled according to one hour intervals and forward-filled (as in Section 3.5.3.3).

4.6 Experiments

4.6.1 Prediction Tasks

We evaluate the performance of all our models on both length of stay (LoS) and in-hospital mortality. By including two different tasks we give a more holistic impression of the performance of the models. Note that mortality is a relatively easier task than mortality. This is covered in Section 3.3.

Table 4.3: Time series features. 'Time in the ICU' and 'Time of day' were not part of the tables in eICU but were added later as helpful indicators to the model.

Feature	Source Table
Bedside Glucose	*lab*
FiO_2	*respiratorycharting*
SaO_2	*respiratorycharting*
Non-Invasive Diastolic	*vitalaperiodic*
Non-Invasive Mean	*vitalaperiodic*
Non-Invasive Systolic	*vitalaperiodic*
CVP	*vitalperiodic*
Heart Rate	*vitalperiodic*
Respiration	*vitalperiodic*
st1	*vitalperiodic*
st2	*vitalperiodic*
st3	*vitalperiodic*
Systemic Diastolic	*vitalperiodic*
Systemic Mean	*vitalperiodic*
Systemic Systolic	*vitalperiodic*
Temperature	*vitalperiodic*
Time in the ICU	
Time of day	

4.6.2 Evaluation Metrics

Length of Stay We reported on 6 LoS metrics: mean absolute deviation (MAD), mean absolute percentage error (MAPE), mean squared error (MSE), mean squared log error (MSLE), coefficient of determination (R^2) and Cohen Kappa Score. See Section 3.6.3.1 for further details.

In-Hospital Mortality For mortality we reported the area under the receiver operating characteristic curve (AUROC) and the area under the precision recall curve (AUPRC). See Section 3.6.3.2 for more detail.

4.7 Results

4.7.1 LSTM-GNN Performance on Mortality and Length of Stay

Table 4.4 shows our main results table where we compared the LSTM-GNN performance to two LSTM baselines [46, 106].

Table 4.4: Performance of various LSTM-GNN models. We compared these models to an LSTM baseline (with and without^{-d} diagnosis concatenation). The error margins are 95% confidence intervals (CIs) from 15 independent training runs. The best results are highlighted in blue. If a result is statistically different from the *LSTM* (with diagnoses) on a two-tailed t-test ($p < 0.05$), then it is indicated with ‡ or † to show better or worse performance respectively.

	In-Hospital Mortality		Length of Stay					
Model	AUROC	AUPRC	MAD	MAPE	MSE	MSLE	R^2	Kappa
LSTM^{-d}	$0.837\pm0.001^{\dagger}$	$0.390\pm0.004^{\dagger}$	$1.97\pm0.01^{\dagger}$	**49.4 ± 0.6**	$17.6\pm0.2^{\dagger}$	$0.398\pm0.004^{\dagger}$	$0.089\pm0.008^{\dagger}$	$0.224\pm0.006^{\dagger}$
LSTM	**0.858 ± 0.001**	0.429 ± 0.002	1.95 ± 0.01	49.8 ± 0.9	17.0 ± 0.1	0.382 ± 0.001	0.118 ± 0.003	0.245 ± 0.006
LSTM-SAGE	$0.851\pm0.003^{\dagger}$	0.426 ± 0.010	$1.87\pm0.00^{\ddagger}$	$50.9\pm0.5^{\dagger}$	$14.8\pm0.1^{\ddagger}$	$0.377\pm0.002^{\ddagger}$	0.119 ± 0.005	0.237 ± 0.006
LSTM-GAT	$0.854\pm0.001^{\dagger}$	0.427 ± 0.003	**$1.86\pm0.00^{\ddagger}$**	49.7 ± 0.3	$14.6\pm0.1^{\ddagger}$	$0.371\pm0.001^{\ddagger}$	$0.129\pm0.004^{\ddagger}$	$0.258\pm0.004^{\ddagger}$
LSTM-MPNN	$0.852\pm0.001^{\dagger}$	**0.433 ± 0.004**	**$1.86\pm0.01^{\ddagger}$**	50.5 ± 1.3	**$14.5\pm0.1^{\ddagger}$**	**$0.369\pm0.001^{\ddagger}$**	**$0.136\pm0.007^{\ddagger}$**	**$0.261\pm0.005^{\ddagger}$**

Table 4.5: Performance of various dynamic LSTM-GNN^{-d}s compared to LSTM^{-d}. These models do not have diagnoses in their static features; they create the graph from the temporal features alone.

	In-Hospital Mortality		Length of Stay					
Model	AUROC	AUPRC	MAD	MAPE	MSE	MSLE	R^2	Kappa
LSTM^{-d}	0.837 ± 0.001	**0.390 ± 0.004**	1.97 ± 0.01	**49.4 ± 0.6**	17.6 ± 0.2	0.398 ± 0.004	0.089 ± 0.008	0.224 ± 0.006
Dyn. LSTM-GCN^{-d}	**$0.839\pm0.001^{\ddagger}$**	0.388 ± 0.002	**$1.96\pm0.01^{\ddagger}$**	50.2 ± 1.0	**$17.0\pm0.1^{\ddagger}$**	**$0.387\pm0.002^{\ddagger}$**	**$0.117\pm0.007^{\ddagger}$**	**$0.251\pm0.005^{\ddagger}$**
Dyn. LSTM-GAT^{-d}	$0.832\pm0.001^{\dagger}$	$0.358\pm0.005^{\dagger}$	1.97 ± 0.01	50.2 ± 1.4	$17.3\pm0.1^{\ddagger}$	$0.393\pm0.003^{\ddagger}$	$0.105\pm0.007^{\ddagger}$	$0.236\pm0.006^{\ddagger}$
Dyn. LSTM-MPNN^{-d}	0.837 ± 0.001	0.389 ± 0.002	**$1.96\pm0.01^{\ddagger}$**	50.0 ± 1.1	$17.1\pm0.1^{\ddagger}$	$0.389\pm0.002^{\ddagger}$	$0.113\pm0.007^{\ddagger}$	$0.248\pm0.005^{\ddagger}$

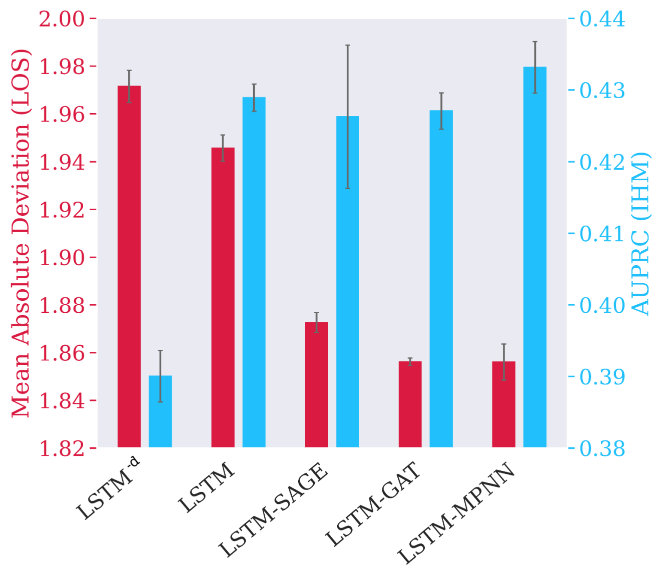


Figure 4.6: A subset of the results in Table 4.4 comparing the performance on one LoS metric (MAD) and one in-hospital mortality metric (AUPRC).

The first baseline (LSTM^{-d}) does not take any diagnoses as input[2], whereas the second baseline (LSTM) processes diagnoses according to the commonly applied encoder concatenation approach. The first thing to note is that LSTM significantly outperforms LSTM^{-d}, confirming that diagnoses add predictive value to both tasks.

Importantly, all of the LSTM-GNN models demonstrate significant performance gains compared to both LSTM baselines on the *LoS* task. The LSTM-MPNN in particular demonstrates impressive performance, surpassing LSTM by $3 - 15\%$ on all LoS metrics except MAPE. This is not particularly surprising, because the LSTM-MPNN model is the most expressive of the GNNs evaluated, as it has the capacity to model edge features (similarity scores from Equation 4.1) while other GNNs do not (Section 2.4.4).

Additional investigation revealed that the error reduction in the LSTM-GNN models corresponds to long length of stays, where LSTM-GNNs were more accurate in giving estimates for patients staying longer than 2-3 days. This explains the disproportionate reduction in MSE but not MAPE, as the MSE is more influenced by long LoS outliers. On mortality, the LSTM-GNN models tend to show a small (but statistically insignificant) increase in AUPRC, but a reduction in AUROC.

The performance benefit in the LoS task but not the mortality task can be clearly seen in Figure 4.6. This may be attributable to the increased reliance on operational factors for LoS e.g. different discharging practices [18], which in turn depend on the diagnoses. This is not upheld on the mortality task because the vital signs and a few common diagnoses (which can be easily extracted from the diagnosis encoder) remain the most reliable predictors of mortality risk.

[2]I use $^{-d}$ to denote all models which exclude the diagnosis vector from input $\mathbf{s}$.

4.7.2 Dynamic LSTM-GNNs

Until now I have proposed a fixed patient graph constructed using diagnoses. However, we also investigated whether a useful graph can be learnt dynamically[3] from the time series alone (in the absence of diagnoses). This is to test whether the graph approach is only useful for modelling sparse information, or whether it can be beneficial even without the sparse data to encode. Inspired by Dynamic Graph CNNs [127], we explored a *dynamic* variant of LSTM-GNN. Here we trained an LSTM on the time series $\mathbf{x}$ with mini-batching, each time computing the pairwise Euclidean distance of the hidden vectors $\mathbf{h}^L$ in the batch. Again, I applied k-NN to obtain the graph.

Table 4.5 shows that LSTM^{-d} and dynamic LSTM-GNN^{-d}s generally perform similarly on mortality, but the dynamic LSTM-GNN^{-d}s again have an advantage on the LoS task. We also observed that the dynamic LSTM-GCN^{-d} model, despite not having access to any diagnoses, performs similarly to LSTM (second row of Table 4.4). This suggests that relating patients using a graph structure has value for modelling patient outcomes independently of diagnoses. This is possibly because where the data is poor quality or missing, the model can rely more on the neighbouring patients. However, the most visible gains (Table 4.4) still come from using diagnoses for the graph construction.

4.7.3 Ablation Studies

To understand the impact of our design choices, we studied the model performance under different ablations i.e. without the diagnosis vector component and the LSTM encoder.

Table 4.6a shows the performance of the LSTM-GNN models without a diagnosis vector presented in the traditional way (LSTM-GNN^{-d}). Firstly, we can see that all of the LSTM-GNN^{-d} models easily outperform LSTM^{-d}, indicating that the patient graph alone is an informative representation of diagnosis data.

When we considered the impact of the graph only vs. the combination approach (i.e. various LSTM-GNN^{-d}s vs. LSTM-GNNs in Table 4.4), we see that the combined approach in the LSTM-GNNs produces the best results. However, the difference is more marginal for the LoS task, which suggests that the graph confers the largest benefit for LoS, whereas the encoder is more important for mortality prediction. This can be explicitly verified by comparing the LSTM-GNN^{-d}s to LSTM; where the LSTM-GNN^{-d} models do indeed outperform on LoS, but not on mortality.

Table 4.6b shows the performance of the graph on the raw time series (i.e. no LSTM component). Both of these models perform significantly worse than their respective LSTM-GNNs, which validates the need for an LSTM component for time series processing.

[3]I use the term 'dynamic' because the graph is calculated per mini-batch as the LSTM is training. This is in contrast to the scoring based graph described in Section 4.4.1 which is fixed at the start of training. Note that I do *not* mean that the graph is changing over the course of the patients' time series.

Table 4.6: Ablation Studies. (a) shows the performance of various LSTM-GNN^{-d} models (without diagnoses). The t-tests are performed with respect to LSTM^{-d}. (b) shows the results when GraphSAGE and GAT operate without an LSTM i.e. we provided the raw time series as input to the GNN. They are compared to their respective LSTM-GNN models.

		In-Hospital Mortality				**Length of Stay**			
	Model	**AUROC**	**AUPRC**	**MAD**	**MAPE**	**MSE**	**MSLE**	$\boldsymbol{R^2}$	**Kappa**
(a)	LSTM^{-d}	0.837±0.001	0.390±0.004	1.97±0.01	**49.4±0.6**	17.6±0.2	0.398±0.004	0.089±0.008	0.224±0.006
	LSTM-SAGE^{-d}	**0.840±0.001**‡	**0.397±0.006**‡	1.88±0.01‡	50.7±1.2	14.9±0.1‡	0.380±0.001‡	0.117±0.006‡	0.240±0.006‡
	LSTM-GAT^{-d}	0.838±0.002	0.384±0.008	1.88±0.01‡	50.1±1.4	15.0±0.1‡	0.383±0.003‡	0.108±0.009‡	0.234±0.005‡
	LSTM-MPNN^{-d}	0.836±0.001	0.392±0.003	**1.87±0.01**‡	50.8±1.7	**14.7±0.2**‡	**0.377±0.004**‡	**0.128±0.011**‡	**0.255±0.008**‡
(b)	SAGE	**0.853±0.001**	0.406±0.003†	1.96±0.00†	**50.7±0.9**	17.1±0.1†	0.389±0.001†	0.113±0.006	**0.239±0.004**
	LSTM-SAGE	0.851±0.003	**0.426±0.010**	**1.87±0.00**	50.9±0.5	**14.8±0.1**	**0.377±0.002**	**0.119±0.005**	0.237±0.006
	GAT	0.833±0.001†	0.357±0.003†	2.02±0.01†	52.2±1.0†	18.0±0.1†	0.423±0.003†	0.066±0.006†	0.186±0.006†
	LSTM-GAT	**0.854±0.001**	**0.427±0.003**	**1.86±0.00**	**49.7±0.3**	**14.6±0.1**	**0.371±0.001**	**0.129±0.004**	**0.258±0.004**

4.7.4 Interpretability

The LSTM-GAT model in particular provides a unique interpretability benefit. As explained in Section 2.4.2, the GAT model assigns attention weights to the edges in the graph, meaning that we can qualitatively assess what the model is examining.

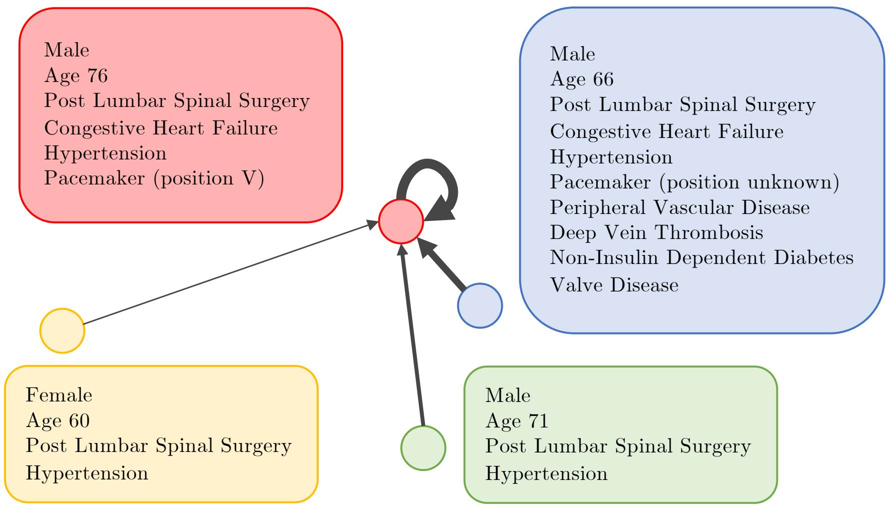


Figure 4.7: An example showing graph attention weights in LSTM-GAT^{-d}. The value of the attention weight is indicated by the edge thickness.

Figure 4.7 depicts a 76 year-old post-lumbar spinal surgery patient and his neighbours. It is typical for post-surgical patients to have shorter stays in the ICU [41], but this patient has congestive heart failure which is associated with high mortality and longer recovery times [1]. By examining the attention weights in a learned LSTM-GAT^{-d} model, we see that it places greatest importance on the self-node (the patient's own data), followed by another patient who shares his congestive heart failure diagnosis and other cardiovascular comorbidities. Less importantly, there are two other post-surgical patients but they do not share the heart failure diagnosis so they are downweighted. The model places slightly greater emphasis on the older male patient. Altogether we see that the LSTM-GAT is behaving as expected, and this could offer an important sanity-checking benefit for the end user.

4.8 Summary

In this work, I proposed and evaluated a new LSTM-GNN approach for sparse data processing in EHRs, using diagnoses as an example. Our results demonstrate that the representation of diagnoses as a graph confers an independent and substantial performance benefit when combined with the commonly applied encoder approach on the LoS task. This makes intuitive sense when considering the different architectures because the encoder method may be preferable for representing common comorbidities which confer strong correlations with the prediction task e.g. sepsis. However, the graph method provides a context for rarer patterns of disease by directly presenting example cases in the local neighbourhood. Note that the graph may also help to augment the data where the quality in the original patient is poor, and offer interpretability benefits (see Section 4.7.4). Since our approach and the traditional approach offer complementary insights, their respective contributions can be combined to obtain better performance in the LSTM-GNN.

CHAPTER 5

Dynamic Outcomes-Based Clustering

This work came about after I returned to a topic that I had studied during my first year of the PhD programme. My previous work focused on using reinforcement learning to learn optimal ventilator settings for patients with Acute Respiratory Distress Syndrome (ARDS). This work was never published and has been omitted from this dissertation, however, it left me with a sense of unresolved interest in mechanical ventilation. I mentioned this to a collaborator, Dr Ari Ercole, after working together on a project to forecast COVID-19 bed occupancies across England in March 2020, and he was keen to help.

After Ari Ercole and Ioana Bica – a friend and fellow PhD student – came onboard the project, we refocused our efforts to the discovery of hidden phenotypes in the data, rather than the optimisation of ventilator settings. I exchanged ideas with Ari and Ioana throughout, but the following work is my own, from the ideas to the experimentation and writing of the text. I was accepted to the Learning from Time Series for Health (TS4H) Workshop at NeurIPS 2022, and three workshops at AAAI 2023, including the Health Intelligence (W3PHIAI) Workshop where I won Best Paper!

5.1 Why Cluster Patients?

The advancement of Electronic Health Records (EHRs) and machine learning have enabled an increasingly data-driven and personalised approach to healthcare. One step in this direction is to uncover patient sub-types with similar disease trajectories in a heterogeneous population. Patients in the ICU can deteriorate rapidly over hours, minutes or even seconds. Understanding the stability, or otherwise, of a particular physiological state at a particular time is important in terms of flagging the potentially deteriorating patient and enabling early intervention.

Furthermore, randomised controlled trials (RCTs) in the ICU are overwhelmingly negative [34, 59, 124]. One important reason for this is the extreme heterogeneity in the ICU population. Unlike diseases such as ischaemic heart disease or COVID-19 pneumonitis with potentially devastating but simple aetiologies, the typical intensive care patient may have a wide variety of reasons for admission and mechanisms leading to physiological disruption, have complex multi-system disease, complex and varied comorbidities and treatment histories. There is a great deal of interest in finding stereotypical disease behaviours or 'phenotypes' or 'sub-phenotypes' in the hope that these might represent pathobiological 'endotypes' which may be both more homogeneous and represent a principled substrate for individualised intervention. Data-driven phenotype discovery is a particularly attractive idea and there have been numerous attempts to apply even relatively simple clustering techniques to either routinely collected data or, more recently, genetic, transcriptomic or metabolomic data in critical care patients. Associations with outcome have been demonstrated which is encouraging. Furthermore there have also been a number of areas where relatively simple clustering techniques have revealed *actionable* sub-phenotypes by secondary analysis of RCT data where intervention data is also available. For example, latent trajectory modelling of inflammatory biomarkers has revealed pro- and anti-inflammatory sub-types of Acute Respiratory Distress Syndrome (ARDS) [29] and these have a differential response to liberal or conservative fluid therapy policies. Clustering of transcriptomic data has revealed patient populations in which steroid therapy may be beneficial in sepsis [2]. Routinely collected data has also been used to find simple trajectory clusters for sepsis based on physiological parameters [7] which have been shown to have differential response to fluids.

Investigating the dynamic clustering of trajectories is appealing in the ICU for reasons touched on above. However, since the ICU already generates large panels of data, temporal trajectories are extremely high-dimensional spaces in which to undertake clustering. As I have shown in previous chapters, temporal neural network architectures are highly flexible models which can handle the heterogeneous population in the ICU, yielding high predictive performance. We also have reasons to believe that there is merit in an unsupervised approach to patient representation [70, 126]. This motivates examining a combined supervised and unsupervised approach to patient representation as a substrate for trajectory clustering.

Mechanical ventilation (by which, in this thesis, I take to mean *invasive* mechanical ventilation) is a common and high-risk intervention in intensive care and a key area of research although again many clinical trials [34] have been unrewarding. Mortality and morbidity can be high and the need for treatment over time is difficult to predict: some patients can be liberated from mechanical ventilation quickly, whereas others may need a prolonged period of ventilation and potentially tracheostomy to allow a more gradual transition to spontaneous breathing. This may be multifactorial: difficult weaning from mechanical ventilation may be due to either acute or chronic pulmonary injury, frailty or deconditioning due to other critical illness (critical illness may increase physiological demands on the cardiorespiratory system at the same time as being *catabolic* i.e. associated with muscle breakdown and weakness and this may include respiratory muscles). The inherent temporal patient heterogeneity is a major stumbling block for ventilation research: the discovery of more homogeneous *phenotypes* is an encouraging area in the hope that some of these phenotypes may subsequently turn out to be clinically actionable.

5.2 Key Contributions

In this chapter, I have developed a dynamic clustering approach for mechanically ventilated patients in the Intensive Care Unit (ICU). I used a mixture of supervised, self-supervised and unsupervised techniques to reveal key subgroups in the Amsterdam UMC critical care data set [116]. The clusters are designed to share similarities in *phenotype*, *trajectory* and *outcomes*. The assignment is dynamic, meaning that I generate a cluster for each hour that a patient remains in the ICU. This means that if an event happens which alters the predicted trajectory and outcomes, there will be a shift in the cluster assignment. This is interesting, not only because it can reveal which events are associated with these shifts, but it may allow us to gain insight into what could have happened if the ventilation strategy had been different. This would help to improve our understanding, and direct further study into sub-types of disease trajectory. If we can reliably sub-type a patient early on in their admission, we can conduct intervention studies on particular sub-types of patients. Additional clinical use cases include the development of interpretable early warning systems to alert clinicians of deteriorating patients and designating patients to specialised treatment protocols depending on their sub-type.

5.3 Background and Related Work

5.3.1 Mechanical Ventilation and Pulmonary Failure

Pulmonary injury is typically, but very imperfectly, described in terms of the degree of impairment in gas exchange. However patients on mechanical ventilation are a highly heterogeneous group, with widely differing outcomes. Some have relatively healthy lungs e.g. if they are recovering from surgery on another organ; whereas others have varying degrees of pulmonary failure. Pulmonary failure can be acute (e.g. ARDS) and deteriorate rapidly, or chronic, typically evolving slowly. Unfortunately, patients on ventilators have high mortality [74, 84] and there is no established consensus on optimal treatment strategies from randomised controlled trials [4]. Therefore, *there is significant potential benefit of a data-driven strategy to guide future clinical studies.*

5.3.2 Temporal Clustering

Temporal clustering is the unsupervised process of partitioning time series into homogeneous groups. When applied to clinical data, it can be challenging because the time series are not the same length, and the data contains significant noise (both in the underlying physiology and in the recording process). Temporal clustering approaches have been successful previously e.g. in Parkinson's [134], diabetes [100] and cystic fibrosis [62] and increasingly in intensive care as discussed above. However data-driven clustering has not been comprehensively undertaken for mechanical ventilation.

5.4 Methods

Broadly, my strategy was to train a temporal encoder to embed the patient data at every timestep (this is analogous to returning all the hidden states for an LSTM model). I used a mixture of supervised, unsupervised and self-supervised learning to do this. Once the encoder training was complete, I used an unsupervised method to cluster the embeddings, so that I get a cluster for every timestep in the patient's ventilation episode. My code can be found at `https://github.com/EmmaRocheteau/Mechanical-Ventilation-Clustering`.

The data consisted of both time series and static features (see Section 5.5). The supervised tasks included two binary tasks: predicting hospital mortality and the risk of receiving a tracheostomy[1], and two duration tasks: the remaining length of stay (LoS) from timestep t, and their remaining ventilation duration (VD). This ensured that the patient *outcomes* are stored within the embedding. In addition, I trained a decoder to reconstruct timestep t and the static data. This is an unsupervised approach which encourages the embedding to retain the patient *phenotype.* Finally, I predicted timestep $t + 1$, a self-supervised approach designed to embed the patient *trajectory* (see Figure 5.1).

5.4.1 Encoder

In recent years, LSTMs have been by far the most popular model for predicting clinical outcomes and have achieved state-of-the-art results [46, 87, 106]. They have also been applied to other patient prediction tasks e.g. forecasting diagnoses and medications [13, 65], and mortality prediction [12, 46, 107]. More recently, the Transformer model [121] has marginally outperformed the LSTM when predicting LoS [109]. In Chapter 3, I showed that

[1]A tracheostomy is a procedure designed for long term mechanical ventilation of a patient.

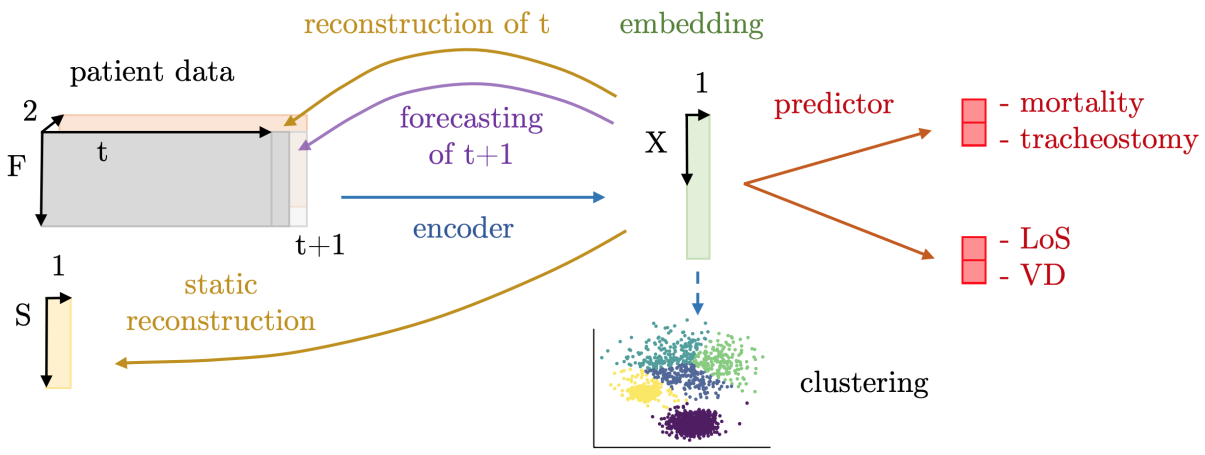


Figure 5.1: Overview of my model. Only one timestep, t, is shown for simplicity. F and S are the number of time series and static variables respectively. At timestep t, the static variables (yellow) and preceding time series variables (grey) and their corresponding decay indicator variables (orange, explained under 'Time Series' in 5.5.3) are given to the encoder, which produces an embedding (green) for timestep t. This embedding is then given to the decoder networks (yellow), forecasting network (purple) and the predictor network to obtain the four patient outcomes (red), outlined in Section 5.4. After training is complete, the test embeddings are used for clustering.

Temporal Pointwise Convolution (TPC) outperformed both the LSTM and Transformer models on mortality and LoS. Therefore, I chose to investigate these three encoders.

5.4.2 K-Medoids Clustering

I used a k-medoids clustering strategy to cluster the learned embeddings. K-medoids is similar to k-means, except that it operates with medoids rather than centroids. The relationship between centroids and medoids is similar to the relationship between means and medians; medoids and medians will always be a true observation in the data, while that is not necessarily the case for centroids and means. The main advantage of using k-medoids is that the method is less sensitive to outliers than k-means, which is more suitable in this context where the data is noisy and heavily skewed[2].

Both k-means and k-medoids operate on pairwise similarities. I decided to use Euclidean distance rather than cosine similarity. This is because intuitively, it is not only the direction that the patient is moving in that matters, but also the distance along that axis. For example, if a particular 'direction' represents acute decompensated heart failure, we also care how severe the decompensation is.

As detailed in Section 5.6.2, I applied batch normalisation [52] to the embeddings, to ensure that the embedding distribution remained within a reasonable range. The value of k (5 for all models) was chosen using the elbow method (see Section C.1.1 in the Appendix).

[2]Preliminary experiments revealed that k-means were more likely to produce small clusters which lay far away from the rest of the data, because it is more affected by outliers. This made the clustering process less reliable and reproducible, which is why I opted for k-medoids clustering.

5.5 Data

5.5.1 Amsterdam UMC Database

I used the Amsterdam UMC database version 1.0.2 [116], which contains 23,106 ICU admissions from 20,109 patients admitted between 2003 and 2016. I selected all of the mechanical ventilation episodes with a minimum duration of 4 hours, capping the maximum duration after 21 days to reduce computational costs. This corresponded to 14,836 episodes which occurred during 13,502 ICU admissions from a cohort of 12,597 unique patients. Of the 14,836 episodes, 13,783 ended in extubation or death of the patient, 648 ended with a tracheotomy procedure occurring within 21 days, 399 patients were still on ventilation at 21 days, and 6 patients were converted to a non-invasive ventilation setting on the ventilator.

I selected 31 time series features and 14 static features. The data were split such that 70%, 15% and 15% were used for training, validation and testing respectively. These were split by *patient*, not ventilation episode, to avoid data leakage from the train set. The cohort summary is shown in Table 5.1.

Table 5.1: Cohort summary for the Amsterdam UMC database. 'Remaining LoS' refers to the remaining duration in the hospital after the start of the ventilation episode.

Number of ventilation episodes	14,836
Train	10,395
Validation	2,230
Test	2,208
Sex (% male)	66.6%
Total LoS in days (mean)	8.26
Total LoS in days (median)	2.13
Remaining LoS in days (mean)	7.26
Remaining LoS in days (median)	2.01
VD in days (mean)	3.95
VD in days (median)	0.83
In-hospital mortality	14.6%
Tracheostomy patients	7.4%
'Urgent' patients	28.1%
Number of input features	45
Time series	31
Static	14

5.5.2 Static Features

I extracted 14 static features from the *admissions* table (Table 5.2). As in Section 3.5.3.1, discrete and continuous variables were scaled to the interval [-1, 1], using the 5th and 95th percentiles as the boundaries, and absolute cut offs were placed at [-4, 4]. Binary variables were coded as 1 and 0. Categorical variables were converted to one-hot encodings, with the exception of 'agegroup', 'heightgroup' and 'weightgroup'. These appear as ordered categories e.g. [18-39, 40-49, 50-59, 60-69, 70-79, 80+] for agegroup. I converted these to an ordered set centred on 0, [-1, -0.6, -0.2, 0.2, 0.6, 1], to preserve the quantitative significance of each category.

Table 5.2: Static features used in the model. 'Null Height' and 'Null Weight' were added as indicator variables to indicate when the height or weight were missing and have been imputed with the mean value. I added the variables 'Admission Count' and 'Ventilation Episode Count' based on previous admissions and ventilation episodes.

Feature	Type	Source Table
Sex	Binary	admissions
Age Group	Discrete	admissions
Height Group	Discrete	admissions
Weight Group	Discrete	admissions
Admission Count	Discrete	
Ventilation Episode Count	Discrete	
Urgency	Binary	admissions
Previous Ward	Categorical	admissions
Specific Location in ICU	Categorical	admissions
Physician Speciality	Categorical	admissions
Weight Source	Categorical	admissions
Height Source	Categorical	admissions
Null Height	Binary	
Null Weight	Binary	

5.5.3 Time Series

For each ventilation episode, I selected 31 time series variables, mostly from the *numericitems* table (these are shown in Table C.5 in the Appendix). I used a semi-automatic process for feature selection. To be included, the variable had to be present in at least 25% of patient stays, and these were further narrowed down with advice from Dr Ari Ercole. I extracted 'diagnosissubgroups' using a query from the AmsterdamUMCdb github repository [115] and ventilator settings from *listitems.* The ventilator settings classification is given in Table C.6 in the Appendix. I engineered the features 'lung compliance' and 'P/F ratio' because they are clinically important, and I have previously noted that neural networks are unreliable when performing divisions. I calculated lung compliance as:

$$\text{Lung Compliance} = \frac{0.73556 \times \text{Expiratory Tidal Volume}}{\text{Peak Inspiratory Pressure} - \text{PEEP}} \tag{5.1}$$

where 0.73556 is a conversion factor to convert lung compliance to its usual unit of ml/cmH_2O. I calculated the P/F ratio as:

$$\text{P/F Ratio} = \frac{\text{PaO}_2}{\text{FiO}_2} \tag{5.2}$$

where FiO_2 is expressed as a fraction rather than a percentage.

The time series variables were processed in the same manner as in Section 3.5.3.3. The decay indicators were calculated as 0.8^j, where j is the time since the last recording. If the variable is up to date, decay is 1, if it has been forward-filled it will be between 0 and 1, and if there is no previous value, decay is 0 and the variable is filled with the mean value for the training set.

5.6 Experiments

5.6.1 Prediction Tasks

5.6.1.1 Remaining Length of Stay and Ventilation Duration

I assigned a remaining length of stay (LoS) and remaining ventilation duration (VD) target to each hour of the ventilation episode, ending when the patient dies or is extubated. The remaining LoS is calculated by subtracting the time elapsed in the ICU from the total LoS. The remaining VD is calculated by subtracting the time elapsed in the ventilation episode from the total VD. I only trained on data from the first 21 days of the ventilation episode to protect against batches becoming overly long and slowing down training.

The remaining LoS and VD each have a significant positive skew which makes the duration tasks more challenging. I partly circumvent this by replacing the commonly used mean squared error (MSE) loss with mean squared *log* error (MSLE), as in Chapters 3 and 4.

5.6.1.2 Mortality and Tracheostomy

Unlike the duration tasks, these tasks are static, i.e. the labels do not change during the ventilation episode. Both tasks have significant class imbalance (only 14.6% and 7.4% of patients died or received a tracheostomy respectively). In order to encourage the model to prioritise learning these important outcomes, I applied class weighting to the task (where the weight is proportional to the inverse of the frequency of each outcome). I used binary crossentropy as the loss function for both tasks.

5.6.1.3 Reconstruction and Forecasting

As shown in Figure 5.1, I use the embedding to reconstruct the timestep t, and forecast one timestep $(t + 1)$ ahead. For the reconstruction of t and forecast of $t + 1$, I apply the mean squared error since the data can (very approximately) be assumed to be Gaussian centred at 0 following normalisation. I also reconstruct the following static features: sex, urgency of admission, agegroup, weightgroup, and heightgroup. The first two are binary, and so I apply the binary crossentropy loss function. The other three are ordered categorical (as explained in Section 5.5.2), therefore I use the mean squared error loss function.

The relative weightings of all of these tasks are given in Section C.1 in the Appendix.

5.6.2 Encoder Implementation

I tested three different encoder models to generate the embeddings. They were all trained as follows. Firstly, the time series are given to the encoder network which processes and then combines them with the static features. These are then passed through a small two-layer pointwise convolution to generate the embeddings (shown in green on Figure 5.1). These are given to a predictor network, a reconstruction network and a forecasting network.

The predictor network is one layer, with four outputs, corresponding to the four outcome tasks – tracheostomy, mortality, LoS and VD. For the binary predictions, I apply a sigmoid activation function to generate a prediction between 0 and 1 and for the duration predictions I apply an exponential function. This is intended to help to circumvent a common issue seen in previous models (e.g. Harutyunyan et al. [46], as they struggle to produce predictions over the full range of durations when the data is very skewed) because it effectively allows the upstream network to model log(LoS) instead of LoS. The log(LoS) distribution is much closer to a Gaussian distribution than the remaining LoS. No activation function is placed on the outputs of the forecasting or time series reconstruction networks, because the variables are continuous. Batch normalisation [52] and dropout [110] is used to regularise the model. The hyperparameter search methodology is described in Section C.1 in the Appendix.

The TPC model is implemented as described in Chapter 3. The LSTM [50] is very similar to the one used in a recent eICU benchmark paper including LoS prediction [106]. The Transformer [121] is very similar to its original implementation except that I added temporal masking to impose causality[3] (see Section C.1 for their hyperparameters).

[3]The processing of each time point can only depend on current or earlier positions in the sequence.

5.6.3 Evaluation Metrics

Binary Tasks For mortality and tracheostomy, I reported the area under the receiver operating characteristic curve (AUROC) and the area under the precision recall curve (AUPRC). These are explained in Section 3.6.3.2.

Duration Tasks I reported on 2 LoS and VD metrics: mean absolute deviation (MAD) and mean squared log error (MSLE). The MAD was used as the primary metric in Harutyunyan et al. [46] and MSLE is arguably the more holistic metric as discussed in Section 3.6.3.1.

Reconstruction and Forecasting Tasks Since these tasks are *auxiliary* (we are not interested in the performance as an outcome of the model), I reported their loss function values (see Section 5.6.1.3) as 'metrics' since they do not need to be interpretable. Note that for the forecasting task, the error can sometimes appear to be smaller than the time series reconstruction error, but this does not mean the model is more accurate when forecasting than reconstructing. It is because the forecasting task has one fewer timestep to reconstruct, because the first time point can never be forecasted (and so the loss across the time series appears to be smaller).

5.7 Results

In this section, I highlight important performance differences between the three encoders, analyse an ablation study on the tasks, and provide a detailed analysis of the clusters produced by the TPC model. A deeper evaluation of the results can be found in Section 5.8.

5.7.1 Task Performance

5.7.1.1 (a) – Full Task Setting

The TPC model performs significantly better than the LSTM and Transformer on the outcome tasks (Table 5.3a), which is in line with previous findings in MIMIC-IV and eICU presented in Chapter 3. The superiority of the TPC model is also evident in the variational and ablation experiments. Interestingly, the Transformer performs poorly on the binary tasks but better on the duration tasks with respect to the LSTM. Additionally, the LSTM performs the best on the reconstruction and forecasting tasks (Table 5.4a). Possible reasons for these findings are discussed in Section 5.8.

Table 5.3: Encoder performance on the prediction tasks averaged over 5 independent training runs. The error margins are 95% confidence intervals. See Section 5.6.3 for the metric definitions. For mortality and tracheostomy, higher AUROC and AUPRC is better; for LoS and VD, lower MAD and MSLE is better. (a) shows the full multi-task setting as shown in Figure 5.1, (b) is a variational alternative to the full task setting. Statistically significant differences are indicated by daggers († = p < 0.05, ‡ = p < 0.001). If the result is significantly better than the comparison models*, it is highlighted in **blue**, if it is significantly worse it is highlighted in **pink**. *In (a) the statistical testing compares the three model types, in (b) each model type is compared to its corresponding 'non-variational' model in table (a).

		In-Hospital Mortality		Tracheostomy		Length of Stay		Vent. Duration	
	Model	AUROC	AUPRC	AUROC	AUPRC	MAD	MSLE	MAD	MSLE
(a)	TPC	**0.833±0.010**†	**0.644±0.013**‡	**0.804±0.007**‡	**0.507±0.020**†	**7.20±0.13**‡	**0.359±0.010**‡	**3.24±0.07**‡	**0.210±0.008**‡
	Transformer	0.697±0.012	0.434±0.019	0.760±0.012	0.419±0.033	8.46±0.07	0.495±0.007	3.95±0.20	0.256±0.016
	LSTM	0.823±0.002	0.608±0.008	0.774±0.002	0.473±0.015	9.16±0.06	0.663±0.008	5.57±0.04	0.681±0.011
(b)	TPC	**0.807±0.006**‡	**0.584±0.014**‡	**0.775±0.008**‡	**0.437±0.012**‡	**9.06±0.10**‡	**0.555±0.018**‡	**4.42±0.03**‡	**0.347±0.006**‡
	Transformer	**0.660±0.023**†	**0.373±0.039**†	**0.714±0.020**‡	**0.353±0.018**†	**9.42±0.27**‡	**0.623±0.020**‡	**4.63±0.27**‡	**0.359±0.030**‡
	LSTM	**0.803±0.004**‡	**0.555±0.006**‡	**0.748±0.005**‡	**0.411±0.010**‡	**10.2±0.1**‡	**0.813±0.016**‡	**5.95±0.04**‡	**0.775±0.007**‡

Table 5.4: Losses for the reconstruction tasks and forecasting task averaged over 5 independent training runs. The error margins are 95% confidence intervals. See Section 5.6.1.3 for explanations of the losses shown. The meaning of (a), (b), the colour scheme and statistical tests are defined in the legend to Table 5.3 above.

	Model	Reconstruction Tasks			Forecasting
		Last Timestep	Static (Binary)	Static (Other)	
(a)	TPC	0.334±0.004	0.013±0.000	0.210±0.038	0.334±0.005
	Transformer	0.351±0.005	0.013±0.000	0.354±0.005	0.347±0.001
	LSTM	**0.297±0.006**‡	**0.012±0.001**†	**0.078±0.010**‡	**0.299±0.004**‡
(b)	TPC	**0.345±0.002**‡	0.013±0.000	**0.332±0.006**‡	**0.345±0.003**‡
	Transformer	0.355±0.006	**0.013±0.000**†	0.356±0.001	**0.353±0.005**†
	LSTM	**0.322±0.003**‡	0.012±0.000	**0.266±0.004**‡	**0.323±0.003**‡

5.7.1.2 (b) – Variational Embedding Spaces

I experimented with making the embeddings ‘variational’, by representing the embedding as a set of means and standard deviations to allow sampling of embedding coordinates. The rationale was that by forcing the embedding space to be smoother, I might improve the quality of the clustering as the distances between patients in the embedding space become more reliable. However, this was found to universally hurt performance (Table 5.3b and Table 5.4b) and it produced clusters which were more homogeneous in terms of outcomes and features, which was counter to the aim of producing clinically distinct clusters.

5.7.2 Ablation Study

I performed an ablation study on the tasks used to train the representation space. The results are shown in Table 5.5. Firstly, we see that the best results for all tasks (except for the duration tasks) are achieved in the full multi-task setting. Not a single metric improves in any of other ablation settings, and yet at least one metric in every ablation setting showed a deterioration in performance (the exception in task setting (g) is discussed below). Overall this indicates that having multiple competing learning objectives has a stabilising effect on learning the representation.

5.7.2.1 (c) – No Forecasting

Experiment (c) included all the tasks except forecasting one timestep ahead. When we compare experiment (c) to (a), we see that the results are mostly similar, but there is a consistent decrease in performance, which is statistically significant at the $p<0.05$ level on the tracheostomy task (AUPRC in the TPC model and AUROC in the Transformer model). On the reconstruction task, again the performance is similar but statistically worse in the last timestep reconstruction in the LSTM model. This means that the forecasting task is contributing slightly to the performance in (a), but the benefit is small.

5.7.2.2 (d) – No Reconstruction

Experiment (d) removes both the timestep t reconstruction and the static data reconstruction tasks, but keeps the forecasting task. The effect size is larger than in (c), but again is only statistically significant on the tracheostomy task. The forecasting task performs significantly worse in the Transformer and LSTM models without the reconstruction.

Table 5.5: Prediction task results for the task ablation study. The full task setting from Table 5.3a has been repeated for ease of comparison. Various task ablations are compared to (a): (c) includes all tasks except for the forecasting task, (d) includes all tasks except for the reconstruction tasks, (e) includes only the prediction tasks, (f) is only the binary tasks, and (g) is only the duration tasks. The colour scheme, metrics and statistical test comparisons are explained in the legend to Table 5.3.

		In-Hospital Mortality		Tracheostomy		Length of Stay		Vent. Duration	
	Model	AUROC	AUPRC	AUROC	AUPRC	MAD	MSLE	MAD	MSLE
(a)	TPC	0.833±0.010	0.644±0.013	0.804±0.007	0.507±0.020	7.20±0.13	0.359±0.010	3.24±0.07	0.210±0.008
	Transformer	0.697±0.012	0.434±0.019	0.760±0.012	0.419±0.033	8.46±0.07	0.495±0.007	3.95±0.20	0.256±0.016
	LSTM	0.823±0.002	0.608±0.008	0.774±0.002	0.473±0.015	9.16±0.06	0.663±0.008	5.57±0.04	0.681±0.011
(c)	TPC	0.831±0.006	0.645±0.009	0.796±0.006	**0.499±0.016**†	7.24±0.12	0.360±0.005	3.26±0.07	0.210±0.004
	Transformer	0.675±0.052	0.399±0.079	**0.743±0.011**†	0.406±0.022	8.44±0.29	0.492±0.024	3.95±0.25	0.251±0.026
	LSTM	0.820±0.003	0.608±0.003	0.773±0.005	0.473±0.014	9.16±0.04	0.663±0.005	5.60±0.05	0.685±0.010
(d)	TPC	0.832±0.005	0.645±0.016	0.796±0.007	**0.483±0.020**†	7.28±0.09	0.362±0.007	3.29±0.06	0.213±0.002
	Transformer	0.698±0.017	0.431±0.041	**0.743±0.008**†	0.391±0.008	8.44±0.23	0.492±0.019	3.91±0.39	0.253±0.033
	LSTM	0.820±0.003	0.608±0.007	0.773±0.002	0.464±0.011	9.19±0.04	0.669±0.006	5.59±0.03	0.688±0.010
(e)	TPC	0.828±0.004	0.643±0.010	0.798±0.005	**0.480±0.020**†	7.38±0.20	0.367±0.020	3.24±0.07	0.212±0.012
	Transformer	**0.676±0.019**†	0.410±0.034	**0.736±0.021**†	0.383±0.026	8.67±0.27	0.509±0.024	4.12±0.22	0.268±0.017
	LSTM	0.819±0.005	0.604±0.013	0.773±0.002	0.475±0.008	9.20±0.04	0.669±0.008	5.61±0.04	0.691±0.012
(f)	TPC	**0.823±0.006**†	**0.626±0.014**†	**0.793±0.002**†	**0.477±0.017**†	-	-	-	-
	Transformer	0.669±0.036	**0.373±0.048**†	**0.737±0.021**†	0.400±0.038	-	-	-	-
	LSTM	**0.817±0.003**†	**0.597±0.007**†	**0.767±0.003**‡	0.458±0.016	-	-	-	-
(g)	TPC	-	-	-	-	**6.99±0.10**†	**0.341±0.007**†	**3.08±0.09**†	**0.180±0.004**‡
	Transformer	-	-	-	-	**8.18±0.12**‡	**0.472±0.012**†	**3.68±0.18**†	**0.224±0.009**†
	LSTM	-	-	-	-	**9.05±0.05**†	**0.644±0.006**‡	5.55±0.01	**0.668±0.003**†

Table 5.6: Reconstruction and forecasting losses for the task ablation study. The full task setting from Table 5.4(a) has been repeated for ease of comparison. The following task ablations are compared to (a): (c) includes all tasks except for the forecasting task, (d) includes all tasks except for the reconstruction tasks. The colour scheme and statistical test comparisons are explained in the legend to Table 5.3.

	Model	Reconstruction Tasks			Forecasting
		Last Timestep	Static (Binary)	Static (Other)	
(a)	TPC	0.334±0.004	0.013±0.000	0.210±0.038	0.334±0.005
	Transformer	0.351±0.005	0.013±0.000	0.354±0.005	0.347±0.001
	LSTM	0.297±0.006	0.012±0.001	0.078±0.010	0.299±0.004
(c)	TPC	0.334±0.004	0.012±0.000	0.198±0.020	-
	Transformer	0.349±0.007	0.013±0.000	0.358±0.007	-
	LSTM	**0.305±0.003**†	0.011±0.000	0.085±0.010	-
(d)	TPC	-	-	-	0.339±0.006
	Transformer	-	-	-	**0.355±0.007**†
	LSTM	-	-	-	**0.309±0.006**†

5.7.2.3 (e) – Prediction Tasks Only

Experiment (e) includes the binary and duration prediction tasks, but no reconstruction or forecasting. The performance again deteriorates, particularly on the tracheostomy task, we also start to see a more noticeable deterioration in the duration tasks, although this is not yet statistically significant.

5.7.2.4 (f) – Binary Tasks Only

Experiment (f) follows the trend of worsening performance as tasks are removed. This means that the mortality and tracheostomy tasks consistently benefit from supplementary tasks which help to distinguish signal from noise.

5.7.2.5 (g) – Duration Tasks Only

Experiment (g) shows unexpected results; all of the models return better results when only predicting LoS and VD. This was not what we observed in Tables 3.7, A.7 and A.8, where the performance was either constant in the multitask setting, or it significantly improved for the LoS task. This is discussed further in Section 5.8.

However, overall the trend is such that the more tasks that are included, the better the average results across tasks.

Table 5.7: Average outcomes by cluster ± 95% confidence intervals for the TPC model in the full task setting. Each patient has been classified into a primary cluster, which is the cluster that they spent the majority of their time in. LoS and VD are shown in days.

Cluster	Patients	Mortality (%)	Tracheostomy (%)	Length of Stay	Vent. Duration
1	232	72.0±5.8	1.3±1.5	3.8±0.8	2.4±0.3
2	133	34.6±8.2	38.3±8.4	30.0±3.6	21.4±2.2
3	1,292	1.9±0.7	1.5±0.7	2.8±0.3	0.7±0.0
4	347	4.0±2.1	31.1±4.9	22.0±1.8	7.4±0.9
5	227	26.0±5.7	8.4±3.6	13.0±1.6	7.2±0.9

Table 5.8: Key features averaged by cluster ± 95% confidence intervals for the TPC model in the full task setting. 70+ (%) is the percentage of patients who are 70 years old or more. Sex is written as % male patients. Urgency (urg.) indicates whether the patient was flagged as 'urgent' at admission. Mandatory Ventilation (MV) refers to the ventilator setting and is defined in Table C.6 in the Appendix. The peak inspiratory pressure (Peak I. P.), P/F ratio (P/F) and PEEP are expressed in mmHg. A normal P/F ratio at sea level is ≈400-500mmHg, whereas 200-300mmHg is consistent with mild ARDS under the Berlin criteria [30]. Lung compliance (Lung C.) is expressed in ml/cmH_2O (a normal lung compliance for a mechanically ventilated patient is 50-100ml/cmH_2O).

Cluster	70+ (%)	Sex (%M)	Urg. (%)	MV (%)	Peak I. P.	Lung C.	P/F	PEEP
1	52.2±6.5	59.7±6.3	63.4±6.3	68.3±0.8	25.3±0.2	32.7±0.5	217±2	10.09±0.07
2	54.1±8.6	65.8±8.1	39.1±8.4	43.2±0.4	23.2±0.1	36.8±0.3	220±1	9.97±0.03
3	39.8±2.7	69.7±2.5	14.9±1.9	38.6±0.6	16.1±0.1	58.8±0.7	260±1	6.78±0.03
4	25.9±4.6	68.4±4.9	41.5±5.3	22.1±0.4	17.8±0.1	57.5±0.4	237±1	8.19±0.28
5	40.1±6.4	69.6±5.9	43.2±6.5	41.8±0.5	20.3±0.1	47.1±0.4	243±1	8.83±0.38

5.7.3 Cluster Analysis

As the best performing encoder, I have focused on analysing the clusters produced by the TPC model in the full task setting. In order to analyse the average differences between the patients in each cluster, it was necessary to flatten the clustering into one 'primary' cluster per patient. This was to prevent confusion, since patients can enter multiple clusters during their ICU stay (sometimes only for one or two time points), and this is disproportionately true of the long stay patients. The cluster in which each patient spent the majority of their time in was assigned its primary cluster. If there were multiple modes, then the mode experienced later in the sequence was chosen. The next two sections 5.7.3.1 and 5.7.3.2 characterise the behaviour of the primary clusters. Section 5.7.3.3 will then analyse the dynamic aspects of the clustering from multiple perspectives.

5.7.3.1 Differences in Phenotype and Outcomes

Table 5.7 shows the mean outcomes for each cluster. I also analysed some key features in the original data, to visualise differences in patient *phenotype* that the model identified.

The average values of key features in patients divided by primary cluster are shown in Table 5.8. Whilst further work is needed to understand these clusters, broadly we can say that:

- Cluster 1 contains the sickest patients, with an average mortality of 72.0%. They are relatively short stay patients and unsurprisingly they have the lowest rate of tracheostomy as most do not survive or stay long enough to require complex respiratory weaning. Table 5.8 shows they are primarily ventilated with 'mandatory' ventilation settings, meaning the machine is deciding the respiratory rate, and either the tidal volume or pressure differential across the lung i.e. there is no reliance on (or cooperation with) any respiratory effort from the patient. Furthermore, they have the least compliant lungs with higher peak inspiratory pressure and higher PEEP. This means that the physical properties of the lung have been damaged and it is harder to inflate the lung. The P/F ratio is also the lowest among the clusters, which indicates that the lung function is also impaired i.e. the patients cannot absorb as much oxygen. This is in keeping with severe respiratory distress. We could describe this phenotype as a *'early, life-threatening pulmonary injury'* patient group.

- Cluster 2 also display substantial mortality and, again, from Table 5.8 it becomes clear that these patients also represent a group with severe pulmonary dysfunction like cluster 1. However this phenotype is characterised by very long LoS and VD, with consequent high rates of tracheostomy: this phenotype represents patients who are difficult to wean from mechanical ventilation with resultant long ICU stay. This might be described as a *'pulmonary critical illness'* phenotype.

- Cluster 3 have the best outcomes, with short LoS and low mortality. They are extubated rather than requiring tracheostomy, because they have less pulmonary injury. This appears to be a *'short stay'* phenotype who require a brief period of organ support only, perhaps after significant surgery.

- Cluster 4 have relatively low mortality but high rates of tracheostomy. Table 5.8 shows modest levels of respiratory failure and good lung compliance. Thus, whilst these patients are difficult to wean from mechanical ventilation (like cluster 2), this is due to factors that are not primarily related to pulmonary pathology: once they receive a tracheostomy they are rapidly liberated from mechanical ventilation. Since the cluster seems to associate with younger patients, this does not seem to be related to frailty or chronic comorbidity. We could therefore describe them as a *'general critical illness'* phenotype.

- Cluster 5 shows a moderate to severe group of patients, who are not as acutely unwell as cluster 1, but are still high-risk for mortality. From Table 5.8 we see that

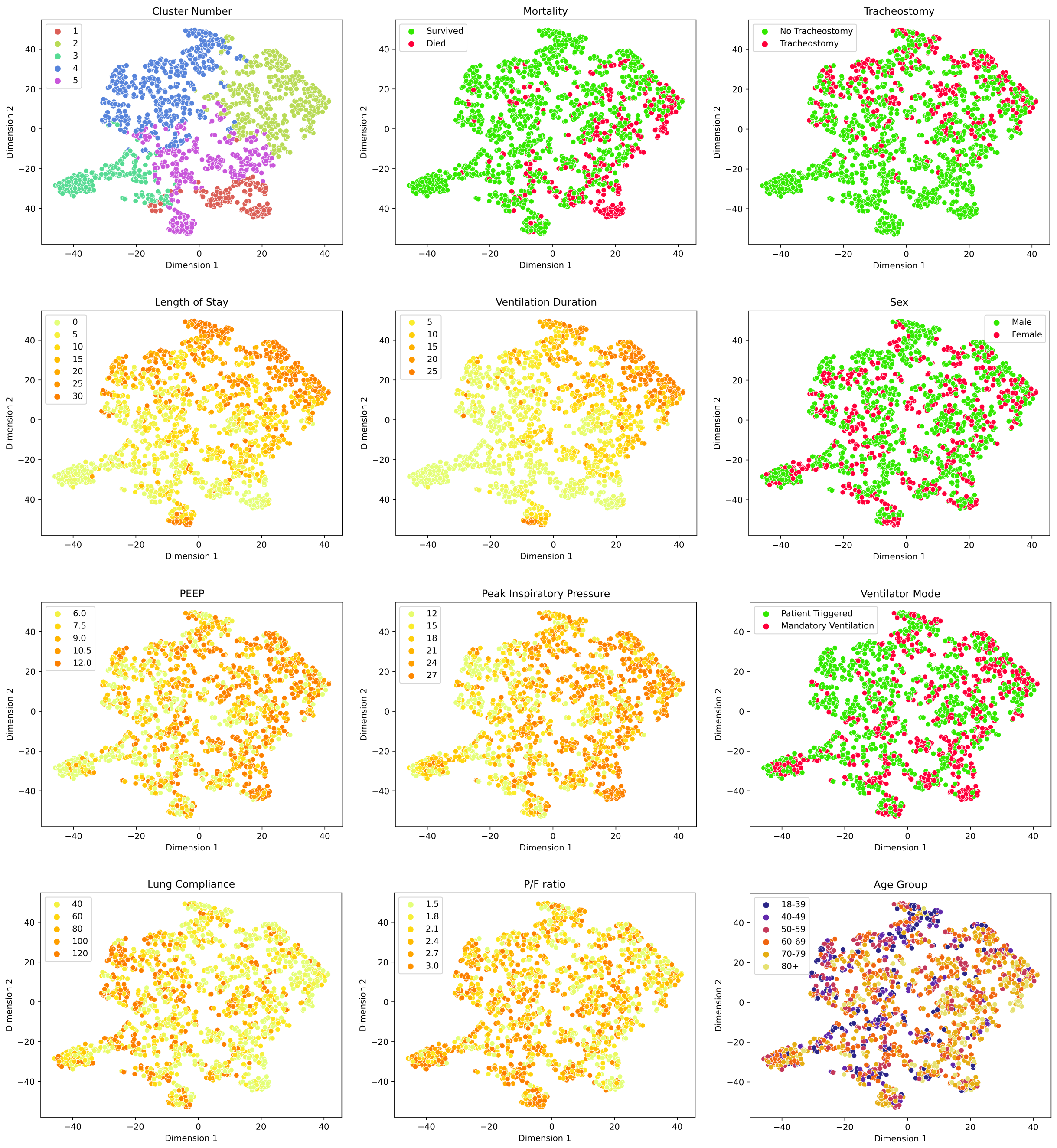


Figure 5.2: t-SNE plots for the embeddings produced by the TPC model in the full task setting. To generate these plots, 1500 random samples were selected from the test set. In each plot, a different attribute has been highlighted. For most variables, there is some form of visible distribution over the representation space (e.g. the upper part of the representation space has long LoS and high risk of tracheostomy), perhaps with the exception of sex. It is interesting to note an area of long LoS and VD at the bottom of the plots (attributed to cluster 5). This will be discussed later in Section 5.8.

pulmonary injury is not a prominent feature so we could characterise these patients as *'early, life-threatening non-pulmonary injury'* patients.

Overall, the findings from Tables 5.7 and 5.8 show that there are statistically significant and clinically meaningful differences between the clusters. These can be visualised in Figure 5.2, which provides maps of the representation space, showing how the clusters relate to one another as well as the distribution of outcomes and features.

5.7.3.2 Medoid Analysis

The medoids produced by the k-medoids clustering algorithm are shown in Figure 5.3 and give a description of a representative patient in each cluster. Note that each medoid corresponds to both a patient and a specific time-point in their ventilation episode[4].

- The medoid patient for cluster 1 (female, age 60-69) died 4 hours after the episode shown without a tracheostomy. Her deterioration is predictable from multiple parameters in the data but infection (high WBC) and pulmonary dysfunction requiring mandatory ventilation are particularly noteworthy. Her heart rate increases significantly towards the end of the episode. The peak inspiratory pressure and PEEP have been set high to overcome poor lung compliance, and yet her tidal volume remained low.

- The typical medoid patient representing cluster 2 (male, 80+ years old) received a tracheostomy 19 days after the episode shown, and was discharged at 23 days. This patient required late as well as early mandatory ventilation suggesting that he ran into later pulmonary complications – mostly likely infectious since his CRP remains high throughout the stay.

- The medoid patient in cluster 3 (female, age 60-69) was discharged from hospital the day after her brief window of ventilation. She does not display substantial physiological derangement.

- The patient in cluster 4 (female, age 60-69) received a tracheostomy 3 days after the sequence shown. Her lung compliance and P/F ratio are both high compared to clusters 1 and 2, indicating better lung function. Therefore, we can conclude that she needed a tracheostomy for reasons other than lung injury. This is also supported by the settings on her ventilator: low peak inspiratory pressure and PEEP and no mandatory ventilation modes. Her end tidal CO_2 levels suggest a degree of hypoventilation or inability to breathe unaided which improves as she is slowly weaned from mechanical ventilation.

[4]Not all of the ventilation episode for each medoid patient is shown in Figure 5.3 – only the time points from the start until the medoid time.

- Lastly, the patient in cluster 5 (female, 80+ years old) stayed for 9 further days in hospital before being discharged. The short duration of ventilation and relatively normal pulmonary physiology is again consistent with a non-pulmonary phenotype.

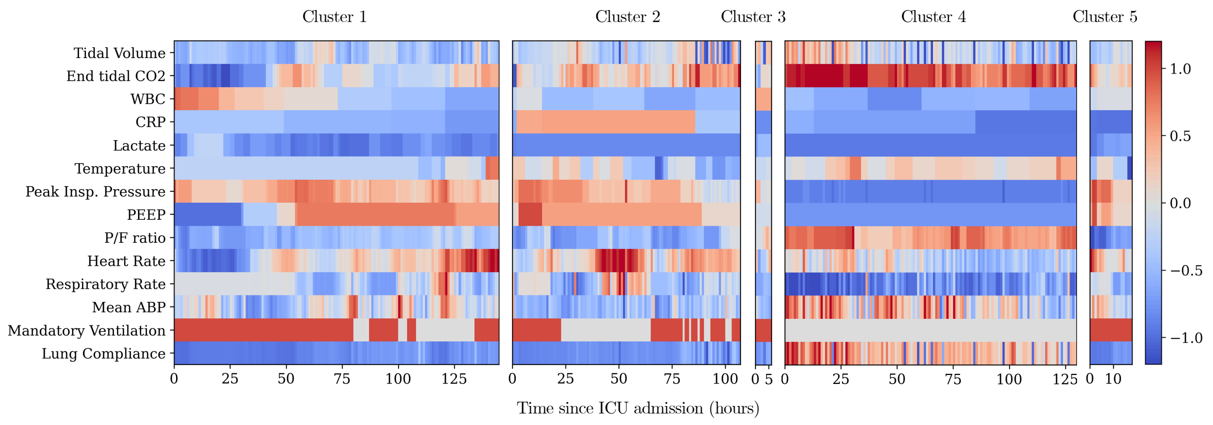


Figure 5.3: Raw data from each of the medoids produced by the TPC model in the full task setting (showing the start of the ventilation episode until the medoid time point). These sequences can be considered the 'architypal' patient trajectory for each cluster. The data have been standardised around the mean value for each feature. Red means the value is high (note that depending on the variable this can be either a good or a bad sign for the patient) and blue means low. We can see that each medoid largely follows the average pattern for the cluster shown in Table 5.8. WBC is white blood count, CRP is C Reactive Protein, ABP is arterial blood pressure.

5.7.3.3 Temporal Analysis

Broadly, there are two perspectives that we can use when evaluating the dynamic aspects of this clustering.

One is the 'Markovian' perspective, where we can examine the transition function between clusters. This is shown in Figure 5.4. Unsurprisingly, this reveals that the patient is always most likely to remain in the same cluster. However the most common inter-cluster transitions are from cluster 5 to cluster 4, and cluster 1 to cluster 5. Note that these clusters are next to one another and share lengthy borders in Figure 5.2 as this will become important later on. Most of the patients who transition to 'Died' come from cluster 1, and most of the 'Discharged' patients come from cluster 3.

The other perspective is to look at the number of patients in each cluster at different time points after admission, and observe the transitions between them (Figure 5.5). Transitions from cluster 3 to 'extubated' are very common within the first day of ventilation, but then they almost disappear by 3 days. This cannot be seen with the Markovian perspective in Figure 5.4. Cluster 2 contains patients with the longest ventilation episodes, which can be seen by its low rate of attrition over time.

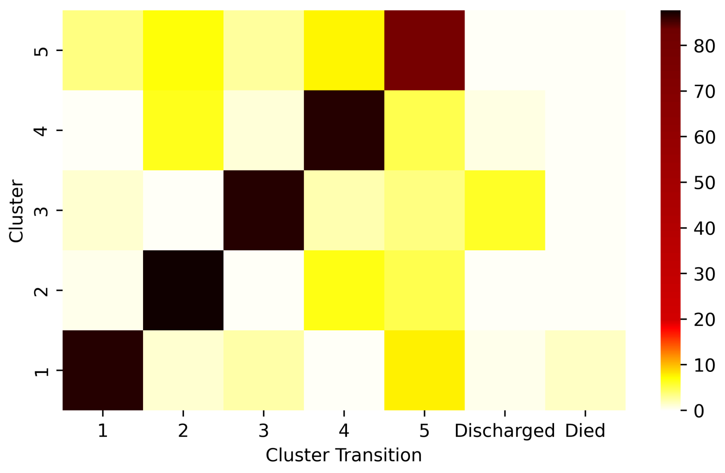


Figure 5.4: A cluster transition matrix for the TPC model in the full task setting, showing the probability of entering each cluster at time $t+1$, plus the categories 'discharged' or 'died', given their current cluster at time t.

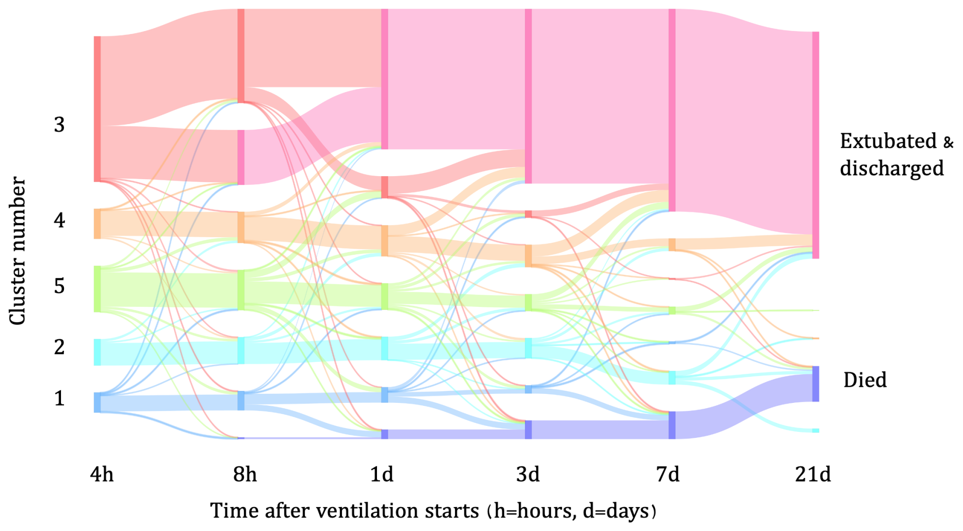


Figure 5.5: A sankey plot showing the evolution of the clustering across time. I begin at 4 hours to allow the clustering to stabilise at the start of the time series. At 21 days there are still some patients without a final outcome (mostly from cluster 2) but this is because they are ventilated for longer than 21 days and have been right censored.

Number of Clusters per Patient Figure 5.6 shows us that most of the patients remain in only one cluster during their ventilation episode. However, when the distribution is broken down by primary cluster, we can see that this is heavily driven by the behaviour of cluster 3 patients, which tend to remain in cluster 3 for their entire ventilation duration (note that they tend to have short VDs so this is not so surprising). In contrast, clusters 2 and 5 most commonly appear alongside other clusters during a single ventilation episode. This means that for most episodes attributed to cluster 2 or 5, there are transitions either into or out of these clusters. These are explored next.

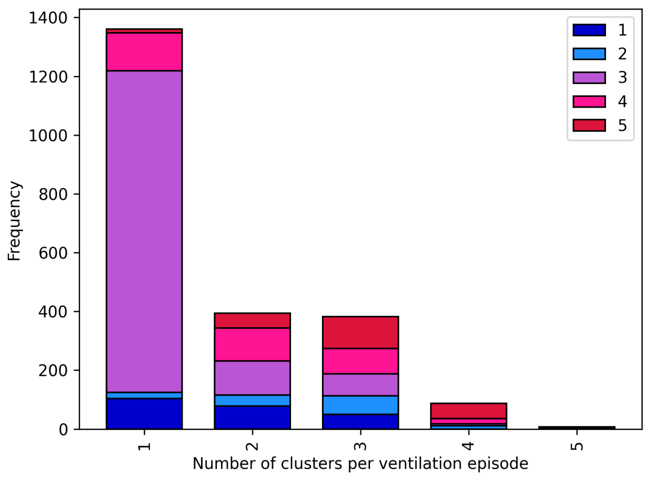


Figure 5.6: Distribution of the number of clusters that the patient enters during their ventilation episode, separated by *primary* cluster (shown by the colour key). For example, cluster 3 (purple) mainly appears on its own i.e. the patient starts the episode in cluster 3 and remains in cluster 3 for the whole duration. In contrast, cluster 5 (red) rarely appears on its own, most commonly cluster 5 patients appear in 2-4 other clusters during their stay.

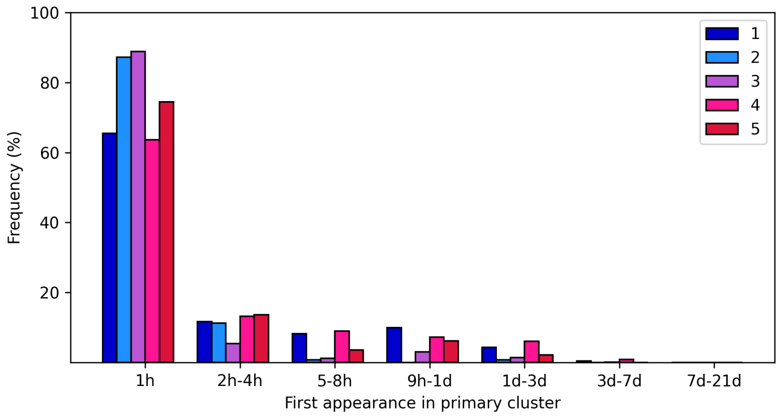


Figure 5.7: Percentage of patients who enter their primary cluster, by time since the start of the ventilation episode.

Cluster Transitions The clusters produced by the TPC model are remarkably stable over time, given that there is no explicit loss incentive to constrain the representation to behave in this way – it is purely learned from the tasks. Figure 5.7 shows the distribution of time points that the patients first enter their primary cluster. Clusters 2 and 3 are particularly likely to accurately assigned during the first hour of ventilation (87% and 89% respectively), while cluster 4 is the least likely to be identified early (64%).

Next, I looked into what I will refer to as 'stable' transitions between clusters. In order to be characterised as stable, the origin cluster needed to remain stable in the 5 hours preceding the transition, and the patient was not permitted to re-enter the origin cluster

Table 5.9: Stable cluster transitions (origin cluster → destination cluster) with a count of >10, sorted by destination cluster. The median rather than the mean time is displayed to show a more representative time of transition (as there is positive skew).

Transition	Count	Median Time	Mortality (%)	Tracheostomy (%)	Urgency (%)	VD	LoS
3→1	17	3	76.5	0.0	47.1	0.5	0.7
5→1	29	16	51.7	10.3	55.2	4.3	5.3
4→2	12	58	8.3	41.7	16.7	6.7	11.0
5→2	12	4	25.0	25.0	41.7	10.8	17.7
1→3	28	11	10.7	0.0	67.9	1.0	2.6
4→3	12	63	0.0	41.7	58.3	0.7	9.6
5→3	46	9	15.2	4.3	41.3	1.2	6.5
2→4	28	17	10.7	21.4	42.9	6.2	12.8
5→4	27	10	11.1	7.4	48.1	3.4	9.1
1→5	25	3	44.0	4.0	68.0	3.9	6.5
2→5	14	23	28.6	14.3	50.0	6.1	8.7
3→5	15	4	13.3	13.3	53.3	1.9	4.6
4→5	15	56	26.7	26.7	46.7	6.6	11.5

within the first 5 hours following the transition. This was primarily to screen out patients who were at the boundary between two clusters, continually crossing back and forth but not representing a true transition from one cluster to the other. Before screening, there were 22,036 cluster transitions, corresponding to 870 separate ventilation episodes (39% of the total in the test set). Of these transitions, only 291 represented stable movement between clusters. I further removed any transitions between two clusters that had fewer than 10 transition examples, as this would be insufficient to analyse. The remaining 280 transitions are shown in Table 5.9.

Firstly, it is noteworthy that the outcomes reflect the *destination* cluster, not the origin cluster. The exception to this is the 'urgency' column, which is not an outcome, but a label assigned at *admission* and hence is more likely to reflect the *origin* cluster (although it is worth mentioning that the origin cluster is not necessarily the cluster at admission).

Cluster 5 stands out as being disproportionately involved in inter-cluster transitions (114 movements into the cluster and 69 movements out, making 65% of the total number of stable transitions). Of these, the most common is 5→3, which occurs when the model overestimates the risk to the patient early on in the ventilation episode. Not shown in Figure 5.9, is that the average predicted risk of death drops from 56.4% 5 hours prior to the transition, to 41.7% at the point of transition. There is also a corresponding reduction in tracheostomy risk (-13%), LoS (-17.1% after adjustment[5]) and VD (-26.4% after adjustment) as predicted by the model, and dramatic improvements in physiological parameters such as lung compliance (+35%) and P/F ratio (+15%).

[5]There is a 5 hour gap between these predictions, therefore this time difference needs to be removed from the first prediction.

Another interesting transition is 3→1, which happens when the model initially believes the patient to be relatively healthy, but then quickly re-adjusts to predict poor outcomes. Looking in more detail at the raw data, I discovered that these patients are younger (only 23.5% are 70+), which could explain why the model was initially optimistic and why the deterioration is so rapid[6]. I also observed a deterioration in the lung compliance (-26.3%) and P/F ratios (-12.8%), and a change in the ventilator settings – namely higher PEEP and peak inspiratory pressure and lower tidal volumes – reflecting a drop in lung compliance of the patients. Most of these patients died within 12 hours of the transition to cluster 1.

5.7.3.4 Reliability

In this section, I aimed to investigate how reproducible these phenotypes were. I chose to analyse the clusters produced by: i) alternative encoder models, ii) retraining the TPC model with different random seeds and iii) varying the value of k.

Choice of Encoder Figure 5.8 compares the cluster assignments using different encoder models. It is encouraging that there is a strong cohesion between some of the clusters, meaning that the models are picking out genuine and consistent patterns in the data.

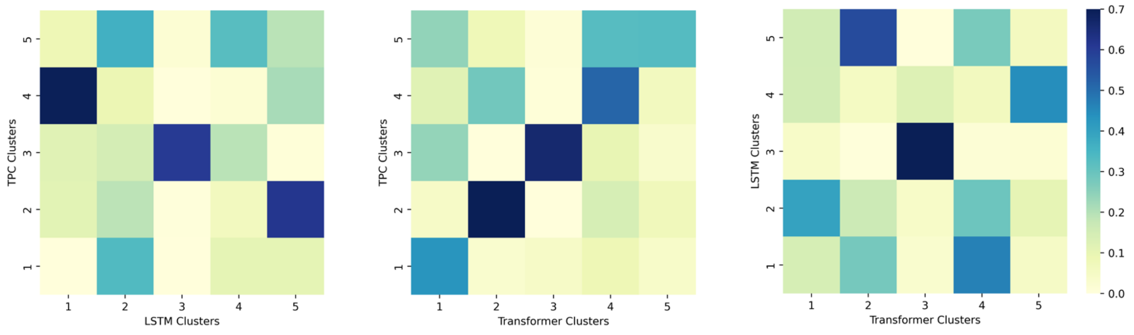


Figure 5.8: A comparison of the cluster assignments produced by different encoders in the full task setting. We can see that there is strong cohesion between some of the clusters. For example, cluster 3 appears to be the same in all three models.

If we examine the TPC/LSTM comparison (far left in Figure 5.8) we can see that clusters 2, 3 and 4 in the TPC correspond to 5, 3 and 1 respectively in the LSTM. In addition, clusters 1 and 5 (TPC) imperfectly map to 2 and 4 (LSTM) – the main difference being that some additional patients in cluster 5 in the TPC map to cluster 2 in the LSTM. This means that the k-medoids algorithm has placed a different boundary between these groups in the clustering process. Looking back to Figure 5.2, clusters 1 and 5 are revealed to be neighbours – in fact, cluster 5 appears to envelope cluster 1, suggesting that 1 is a

[6]This is because younger patients can mask a problem by compensating deceptively well, until they reach a point where the homeostatic mechanisms can no longer cope.

sub-cluster of 5. This is also consistent with the clinical picture shown in Table 5.7, where the main difference is that cluster 1 appear to have acute pulmonary dysfunction, most likely in addition to other organ failures. Therefore, the explanation for this discrepancy is that the LSTM has a more generous threshold than the TPC for inclusion in its highest risk *'early, life-threatening pulmonary injury'* category (cluster 2).

In the TPC/Transformer comparison, the clusters largely correlate, except that cluster 5 patients in the TPC have been placed into cluster 4 in the Transformer. Further investigation revealed these to be patients with increased risk of receiving a tracheostomy (i.e. the patients which lay closest to the decision boundary between the clusters).

The LSTM/Transformer comparison mirrors some of the correlations in TPC/LSTM but the mapping is less precise. This could be because the models do not perform as well, making the representations less reliable. It could also be because the models have different affinities for the various tasks, creating divergent biases in the representation space.

It is worth noting that in all three models, cluster 3 is the most distinct. This is unsurprising because it corresponds to patients whose physiology is closest to 'normal'. Therefore this group is the most homogeneous and can always be identified easily.

Retraining TPC I retrained the TPC model 5 times with different initialisation generated by different random seeds and compared the resulting clusters to the original (Figure 5.9). Overall, there is strong cohesion between the models, but sometimes there are shifts in the boundaries between neighbouring clusters in Figure 5.2. Especially between cluster 5 (moderate-severe) and cluster 1 (severe), where some models (TPC 2, TPC 3, TPC 4) allow patients in cluster 5 to enter their 'cluster 1' equivalent. Nevertheless, very distinct phenotypes are almost never mixed e.g. clusters 2 and 3, 1 and 4, or 1 and 3 (as defined by the TPC 1 model). As in the encoder comparisons, cluster 3 is always well characterised.

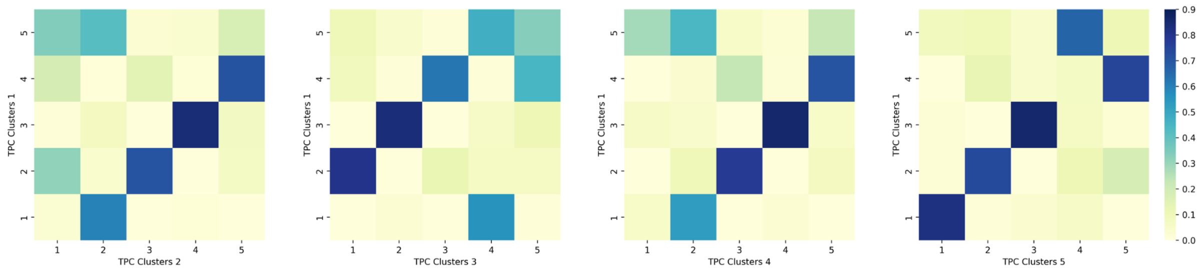


Figure 5.9: A comparison of the cluster assignments produced by TPC models which have been trained with different random seeds.

Number of Clusters The value of k was determined using the elbow method (see Section C.1.1 in the Appendix). In Figure 5.10, I show how the clusters would appear

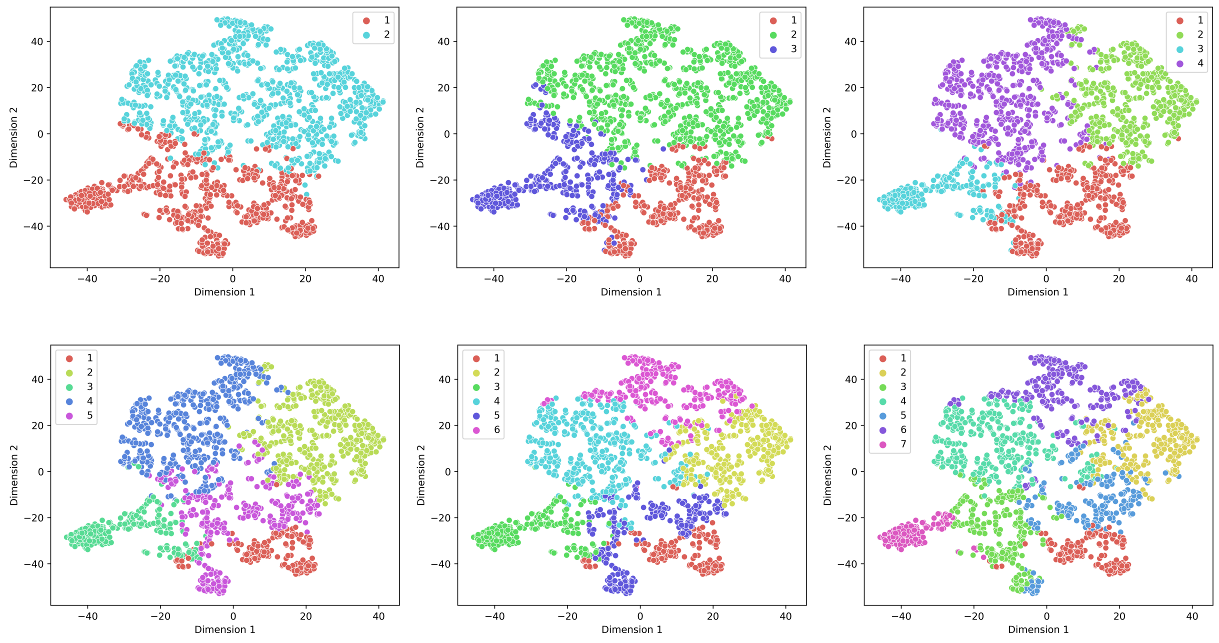


Figure 5.10: Cluster labels with increasing number of clusters (from k=2 to k=7).

with increasing value of k. What is most striking is that each time a cluster is added, the new cluster either inserts itself within an existing cluster, or it appears at the intersection between existing clusters. For example, as we move from 2 to 3 clusters, the new cluster 3 is almost completely contained within the old cluster 1. This pattern of sub-dividing an existing cluster generally continues until we reach 6 and 7 clusters, when the new cluster inserts itself at the boundary between two or more old clusters. In other words, increasing the value of k does not completely shift the position of all of the clusters, but rather it carefully subdivides them. The importance of this behaviour with increasing value of k is discussed in the next section.

5.8 Discussion

In this chapter, I have evaluated the use of TPC model, trained using supervised, unsupervised and self-supervised learning techniques, for the purposes of *phenotype* discovery in mechanically ventilated patients. I will discuss the most important findings in turn.

Firstly, I reaffirmed that the TPC model performs better than alternative encoders on EHR data for patient outcome prediction. This time on the Amsterdam UMC database [116], and with added tasks.

Secondly, the Transformer results in Table 5.3a differ slightly from those in Chapter 3, where the Transformer model performed marginally better than the LSTM on LoS (Table 3.4) and mortality (Tables A.7 and A.8). In Table 5.3, the Transformer outperformed LSTM on LoS and VD, but performed much worse on the mortality task, and slightly worse on the tracheostomy task. This may be because the task weighting – which was fixed between encoder models – was more favourouble to the LSTM and TPC models, whereas the Transformer would have benefited from greater weighting towards the binary tasks. Another possibility is that the binary tasks benefit from biases in the LSTM and TPC encoders, because these models naturally emphasise recent time points over distant ones (and recent time points are more important for solving the binary tasks). The Transformer model has a weaker sense of temporal structure, and hence would have to learn this from first principles. As for the reason that the Transformer performs relatively better on tracheostomy compared to mortality, it could be because of a positive correlation between long LoS, VD and the chance of acquiring a tracheostomy. Solving the duration tasks makes the tracheostomy task easier, whereas the relation to mortality is more complex e.g. both the healthiest patients (cluster 3) and sickest patients (cluster 1) both have relatively short LoS, VD and low chance of getting a tracheostomy, but have very different mortality risks. This can be visualised in Figure 5.2.

To briefly comment on the reconstruction results; initially it may seem surprising that the LSTM model performs best on the reconstruction and forecasting tasks. However, this could be explained if the LSTM creates a slightly simpler, 'lower level' representation that is easier to translate back to the original data using the two-layer fully connected decoder networks. Further work would be required to prove this.

Thirdly, Table 5.5 reveals a general trend that the more tasks that are added, the better results across all the tasks, with particular benefits to the tracheostomy task. The exception to this was the duration only setting (g), which achieved the stronger results for the LoS and VD tasks. Again, this was not the pattern in Chapter 3, where I found that mulitask settings benefited both mortality and LoS performance (Tables A.7 and A.8). There are two possible explanations for the discrepancy:

1. The task weighting applied to the duration task was not sufficient.
2. The tracheostomy task (but not the mortality task) reduces the performance on the duration tasks.

The former does not seem likely, because as we already discussed, the Transformer is likely to be under-weighting the binary tasks, and yet, it follows the same trend as the LSTM and TPC. The latter may appear to be counter-intuitive at first, especially in light of the correlation that exists between the duration tasks and tracheostomy procedures. However, strong associations between tasks may not always lead to better performance. It is logical that the greater the similarity between two tasks, the more likely that one task will interfere with crucial aspects of the representation required for solving the other. This is usually an advantage of multitask learning, because it enhances the signal:noise ratio when certain types of noise only apply to one task. However, when there is non-uniform correlation between tasks in the representation space, it could harm the performance. Looking closely at Figure 5.2, we can see that there is an area of patients near the bottom of the figure, in cluster 5. These patients have long VD and LoS but have been separated from the other long stay patients in clusters 2 and 4. The separation can be attributed to these patients never receiving tracheostomies, unlike the patients in clusters 2 and 4. Therefore, the addition of the tracheostomy task forces the representation space to separate these groups, when they would be otherwise be aligned in a duration-only setting. Given the simple nature of the predictor networks, this separation may harm the performance on the duration tasks because the predictor cannot effectively map these patients to appropriately long stay predictions. This theory could be formally tested by accompanying the duration tasks with the mortality task only.

Finally, regarding the repeatability of the clustering, I demonstrated that there are key aspects of the learned representations (both of different encoders and TPC instances) that translate to core phenotypic traits that are consistently recognised. The separation on other traits, especially when distinguishing the sickest patients from the moderately ill, was more malleable. This suggests that perhaps there is not a well defined distinction between these, but rather a scale of deterioration, through which an arbitrary line can be drawn.

5.9 Summary

In this chapter, I performed temporal clustering on patient representations with the aim of revealing clinically distinctive patient phenotypes. The purpose was to characterise and understand the behaviour of these groups, to guide future research on more personalised approaches to treatment in the ICU. I have made the following contributions:

1. I have reaffirmed that the TPC model outperforms alternative encoders on patient outcome prediction tasks, and can also be used for temporal clustering.

2. I have employed various visualisation techniques to show that the clusters do represent clinically interpretable clusters with meaningful differences in outcome, trajectory and phenotype.

3. Key features of the discovered phenotypes are stable across choices of encoder and k (the number of clusters).

4. The cluster assignment is remarkably stable over time, and membership is determined very early into admission. This is particularly encouraging as a substrate for future intervention studies that would rely identifying the phenotype of the patient early on.

5. I found that true cluster transitions do occur in a minority of patients, where the prognosis suddenly changes in light of an unexpected event. Studying these stable transitions in more detail with a view towards understanding the possible causes is an important avenue for future work.

Further work is required to be confident on the number of clusters and method for clustering and what factors lead to cluster transitions in these (relatively rare) instances.

CHAPTER 6

Conclusion and Future Directions

This dissertation has focused on exploiting clinical knowledge and clinical decision processes to augment the representation of the patient. I have explored medical *time series* and *sparse data* for patient outcome prediction and even the representation space itself for the purposes of clustering. I have addressed the subjects posed by my research questions in Section 1.3. However there is still much to be learned about representation learning for the patient, both in the ICU and beyond. In this Chapter, I will discuss the avenues for future work in the field, starting with my own work. Then, I will address the broader implications for AI in medicine, for the researcher, the clinician, and for the delivery of healthcare as a whole.

6.1 Limitations and Future Work

In **Chapter 3**, I proposed a new model for medical time series - the TPC model. However, there remain several avenues in which to build on the work. Firstly, we know that LoS is heavily influenced by operational factors such as staff working hours, and clinical practices can change over time [56]. Capacity to maintain performance over time is an important consideration before a system could be used in practice. In future work, it would be instructive to test how quickly the models become out-of-date by reserving more recent data as a test set [75]. Although I have included a large set of baselines, I acknowledge that a more exhaustive comparison could be performed, for example comparing with Gaussian Processes [85] or ODE-RNNs [21, 97] for handling irregularly sampled time-series. It also remains unclear why the TPC model gains more from the multitask setting than the other models. It seems likely that it is related to additional regularisation provided by the mortality task, but further investigation is needed to confirm my speculations. Finally, although I have motivated my study with bed management, this work describes a methodological proof of concept and does not constitute a real clinical system. Prospective study and integration into a real-world EHR is necessary to demonstrate real-world benefit, both of which pose their own challenges [87, 104].

In **Chapter 4**, I outlined a method to leverage data from patients with similarities in their diagnoses. A natural extension to the work would be to include shared medications and procedures in the similarity scores. I could also characterise the sensitivity of LSTM-GNN to parameters α, c and k. Nevertheless, our results thus far show that a graph representation of sparse EHR data is a potentially rewarding avenue for future research.

In **Chapter 5**, I performed temporal clustering on patient representations with the aim of revealing clinically distinctive phenotypes. There are many ways in which to expand on this work. Firstly, it is evident from the results that some clusters are more related than others. A tree based hierarchy of clusters seems more natural than a flat structure. It would be interesting to visualise the shifts in cluster assignment within this tree. I am particularly interested in modifying an approach currently used in genetics [11, 17, 81] in order to apply it to the ICU.

Secondly, it is obvious from Table 5.3 that the study is slightly under-powered, since there are only a few statistically significant results. Further experiments would be needed to ensure that these show significant differences between the tasks.

Although I found the resulting clusters to be stable across time and encoder type, I am interested in investigating the use of contrastive learning to further regularise the embedding space e.g. Yèche et al. [131]. Specifically, contrastive learning could be used to explicitly enforce the relative positioning of the embeddings, reducing the potential for unexpected behaviour. While the simple predictor and decoder networks I used in this study may have contributed to the observed stability, I believe that contrastive learning could provide additional benefits and I look forward to exploring this direction in future research.

Regarding generalisability, I have already shown that the results with different encoders are comparable, but repeating the work on another data set, such as MIMIC-IV or eICU would strengthen this assessment. Nevertheless, the work serves as a promising sign that it is possible to perform stable, early phenotyping in the heterogeneous ICU environment. Intervention studies would be required to demonstrate that the findings are actionable.

Although all of my methods were designed with the ICU time series in mind, the insights and modelling advantages extend to wider applications. Many other types of time series data exhibit autocorrelation, seasonality, and trends that could be exploited by the TPC model. The mean-squared logarithmic error (MSLE) is often underutilised in situations of extreme positive skew in labels. Similarly, a temporal encoder could be effectively enriched using a GNN whenever the data can be augmented with similar examples in the data.

6.2 Representation Learning Beyond the ICU

I will now broaden my perspective to consider the wider challenges of representation learning for patients outside the ICU, and into the future.

At present, data is arguably the most obvious constraint. This is because we do not always have the luxury of rich numerical data sets such as those available in the ICU. Currently, the main types of available data for prediction are demographics, diagnoses, medications, procedures, self-reported questionnaires and text from clinical encounters. There is significant potential for this to expand in the future, for example using self-reported data from patients, fitness trackers, GPS location and social media activity. However, at the moment we are mostly limited by the size and quality of data sets in health.

Another broad problem relates to the generalisability of the models. Empirically we know that a model trained on one set of training data can make catastrophic errors on a different data set, even if it is very similar to the first. As it is not practical to train a new model for every scenario, we need further work in this area. The performance of deep learning models is influenced by several stochastic processes during training, for example the order in which the training data is presented. This means that it is impossible to get solid guarantees for performance. In cases where the problems are heavily deterministic, it can be more reliable to simulate the system rather than use deep learning.

Finally, we need to consider the common scenario in medicine where the required task does not have an outcome that can be mathematically defined, for example, “deliver CBT”. It is more difficult to objectively measure whether the agent has performed correctly. For these tasks, we need a human in the loop to reinforce good behaviour and penalise unsuccessful behaviour. This slows down the learning process and is difficult to implement in practice.

In summary, there is great potential for AI-assisted technologies to enhance diagnosis, monitoring and treatment in healthcare. Even within the narrow scope of this dissertation, I have shown that it is possible to predict patient outcomes and make steps towards personalising treatment with deep learning. The vast potential has only just begun to be explored and realised. However, we must approach these technologies from a systematic research perspective and create a framework to assess the risk of any unintended negative consequences of their implementation. Now is the time to invest in AI research so that we can uncover new cost-effective and ethical strategies to reduce the burdens associated with ill-health around the world.

6.3 How Will AI Change the Delivery of Healthcare?

Finally, I will address the impact on *clinical practice* and the *healthcare system as a whole*. Artificial intelligence holds several promises when applied to health. The first is that it will personalise healthcare, recommending bespoke treatments to promote better outcomes for patients. Another is that it will make clinical workflows more efficient by automating processes previously requiring a human in the loop. This will in theory reduce costs and burden on healthcare staff (however this is assuming no new treatments will arise to squeeze the gap opened up by AI). Another hope is that it will improve patient safety and reduce medical errors, much like self driving cars are expected to reduce mortality from road traffic accidents. It also has the potential to make healthcare allocation more fair, provided further research can help to distance the treatment recommendations from human biases and healthcare inequalities in the data required to train them.

The routine collection of new data modalities is likely to increase, and may extend to the home of the patient in the form of wearables and even *implantables*. Note that we need to be wary of exacerbating healthcare inequalities. For example, arrhythmia alerts on smart watches currently have low specificity, and we should be careful that this does not distract attention and resources away from people who cannot afford smart devices but have greater health needs. Population based risk scoring tools will be replaced with more accurate and personalised risk prediction models. All of these could lead to a greater focus on promoting wellness and the primary prevention of disease, rather than the current model of treating problems after the patient has already developed symptoms.

I also believe that the responsibilities of the doctor will shift over time. AI will increasingly preside over the data interpretation requirements, leaving the doctor in a more supervisory role. For example, we could envisage AI running semi-autonomously over screening programmes or routine test results. However, if the model is uncertain, or if the output is critically important for the patient, or if the output does not align with the clinical signs and symptoms, a doctor will always be able to double check the result.

There are aspects of clinical practice where AI cannot perform as well as humans, at least for the foreseeable future. For example, providing advice to patients in light of their personal wishes and circumstances, making difficult ethical decisions, or coordinating the implementation of care plans for patients. Professionalism and good communication skills will continue to be highly important in order to gain trust and gather information from the patient, as will the ability to explain treatment options in a way that the patient can understand. The doctor will remain an expert in medicine and will be ultimately responsible for clinical decision making, but the need to interpret raw data modalities e.g. the ECG (and much more complex data formats) from first principles may become redundant, except for in specialist circumstances.

6.3.1 Preparing the Next Generation of Doctors

This next section contains some of my personal opinions on medical education, and in particular the need to prepare young doctors for the upcoming changes highlighted in the previous section.

As discussed, AI will filter into clinical workflows over the next decade or two, and as doctors we will be expected to use these systems to assist in our decision making while keeping our patients safe. We will need to be sufficiently educated in AI technology, both to competently use, but also to critically appraise the software e.g. knowing when it can and cannot be trusted. Overlooking this need for competent physician-machine interaction may quickly become the biggest hindrance to unlocking the benefits of AI, rather than the technical limitations already discussed. This is because historically, technology has moved much faster than clinical workflows.

There has been little interest in plans to integrate AI teaching into any part of the standard medical education pathway. This is despite the current levels of AI awareness among physicians being very low [42].

What is the Best Way Forward? I believe that there are two fundamental competency levels that we require within the medical workforce. The first is a minimum standard of knowledge that every doctor should have. For example, this could include:

- Critical appraisal of AI systems; including a basic awareness of their strengths and weaknesses.
- Data and model privacy; security vulnerabilities of AI models.
- AI-specific ethics teaching; including the ability to recognise bias in AI and understand why it arises.

Next, we need to foster clinicians with a special interest in AI, who are able to collaborate directly with researchers to guide efforts towards clinically impactful problems, and facilitate the implementation of that research. Currently, we have a very small community of these. This is because the barrier to entry is exceedingly high; the doctor or medical student must spend their spare time to gain the required knowledge and skills. This is usually an inefficient process because most do not know where to start, they have no mentor to guide them, and online resources are not designed for their background. Most who are successful have been lucky enough to encounter a fruitful collaboration or dedicated mentor early on. We need an established, integrated pathway for these AI-interested clinicians with bespoke training, to decrease the barrier to entry and streamline clinician-led progress in AI as a result.

References

[1] Kathy Albert, Bradley Sherman, and Barbara Backus. How Length of Stay for Congestive Heart Failure Patients Was Reduced Through Six Sigma Methodology and Physician Leadership. *Am J Med Qual.*, 25(5):392–397, 2010.

[2] David Antcliffe, Katie Burnham, Farah Al-Beidh, Shalini Santhakumaran, Stephen Brett, Charles Hinds, Deborah Ashby, Julian Knight, and Anthony Gordon. Transcriptomic Signatures in Sepsis and a Differential Response to Steroids. From the VANISH Randomized Trial. *Am J Respir Crit Care Med*, 199(8):980–986, 04 2019.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey Hinton. Layer Normalization, 2016. URL `https://arxiv.org/abs/1607.06450`.

[4] Thomas Bein, Salvatore Grasso, Onnen Moerer, Michael Quintel, Claude Guerin, Maria Deja, Anita Brondani, and Sangeeta Mehta. The standard of care of patients with ARDS: ventilatory settings and rescue therapies for refractory hypoxemia. *Intensive Care Medicine*, 42(5):699–711, 2016.

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives, 2012. URL `https://arxiv.org/abs/1206.5538`.

[6] Stan Benjamens, Pranavsingh Dhunnoo, and Bertalan Meskó. The state of artificial intelligence-based FDA-approved medical devices and algorithms: an online database. *npj Digital Medicine*, 3(1):118, 2020.

[7] Sivasubramanium Bhavani, Matthew Semler, Edward Qian, Philip Verhoef, Chad Robichaux, Matthew Churpek, , and Craig Coopersmith. Development and validation of novel sepsis subphenotypes using trajectories of vital signs. *Intensive Care Med*, 48(11):1582–1592, 2022.

[8] Mathias Blom, Karin Erwander, Lars Gustafsson, Mona Landin-Olsson, Fredrik Jonsson, and Kjell Ivarsson. The probability of readmission within 30 days of hospital discharge is positively associated with inpatient bed occupancy at discharge – a retrospective cohort study. *BMC Emergency Medicine*, 15(1):37, 2015.

[9] Andreas Böhmer, Katja Just, Rolf Lefering, Thomas Paffrath, Bertil Bouillon, Robin Joppich, Frank Wappler, and Mark Gerbershagen. Factors influencing lengths of stay in the intensive care unit for surviving trauma patients: a retrospective analysis of 30,157 cases. *Critical care (London, England)*, 18(4):R143–R143, 2014.

[10] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. BRITS: Bidirectional Recurrent Imputation for Time Series. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[11] Ines Chami, Albert Gu, Vaggos Chatziafratis, and Christopher Ré. From Trees to Continuous Embeddings and Back: Hyperbolic Hierarchical Clustering. *CoRR*, abs/2010.00402, 2020. URL `https://arxiv.org/abs/2010.00402`.

[12] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Scientific Reports*, 8(1):6085, 2018.

[13] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter Stewart, and Jimeng Sun. Doctor AI: Predicting Clinical Events via Recurrent Neural Networks. *JMLR workshop and conference proceedings*, 56:301–318, 2015.

[14] Edward Choi, Zhen Xu, Yujia Li, Michael Dusenberry, Gerardo Flores, Yuan Xue, and Andrew Dai. Graph Convolutional Transformer: Learning the Graphical Structure of Electronic Health Records. *arXiv preprint: 1906.04716*, 2019.

[15] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017.

[16] Jacob Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[17] Gabriele Corso, Rex Ying, Michal Pándy, Petar Veličković, Jure Leskovec, and Pietro Liò. Neural Distance Embeddings for Biological Sequences, 2021. URL `https://arxiv.org/abs/2109.09740`.

[18] Bérengère Couturier, Fabrice Carrat, and Gilles Hejblum. A systematic review on the effect of the organisation of hospital discharge on patient health outcomes. *BMJ Open*, 6(12), 2016.

[19] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[20] Deborah Dahl, Greg Wojtal, Michael Breslow, Randy Holl, Debra Huguez, David Stone, and Gloria Korpi. The High Cost of Low-Acuity ICU Outliers. *Journal of healthcare management / American College of Healthcare Executives*, 57:421–434, 2012.

[21] Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. GRU-ODE-Bayes: Continuous Modeling of Sporadically-Observed Time Series. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[22] Thuppahi Sisira De Silva, Don MacDonald, Grace Paterson, Khokan Sikdar, and Bonnie Cochrane. Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT) to Represent Computed Tomography Procedures. *Comput. Methods Prog. Biomed.*, 101(3):324–329, 2011.

[23] Jacob Deasy, Emma Rocheteau, Katharina Kohler, Daniel Stubbs, Pietro Barbiero, Pietro Liò, and Ari Ercole. Forecasting Ultra-early Intensive Care Strain from COVID-19 in England, v1.1.4. *medRxiv*, 2020.

[24] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, 2019.

[26] Alberto Garcia Duran and Mathias Niepert. Learning graph representations with embedding propagation. In *Advances in neural information processing systems*, pages 5119–5130, 2017.

[27] Anne Elixhauser, Claudia Steiner, Robert Harris, and Rosanna Coffey. Clinical Classifications Software, 2015.

[28] William Falcon. Pytorch lightning. *GitHub*, 2019. URL `https://github.com/PyTorchLightning/pytorch-lightning`.

[29] Katie Famous, Kevin Delucchi, Lorraine Ware, Kirsten Kangelaris, Kathleen Liu, Taylor Thompson, Carolyn Calfee, and ARDS Network. Acute Respiratory Distress Syndrome Subphenotypes Respond Differently to Randomized Fluid Management Strategy. *Am J Respir Crit Care Med*, 195(3):331–338, 02 2017.

[30] ARDS Definition Task Force, Marco Ranieri, Gordon Rubenfeld, Taylor Thompson, Niall Ferguson, Ellen Caldwell, Eddy Fan, Luigi Camporota, and Arthur Slutsky.

Acute Respiratory Distress Syndrome: The Berlin Definition. *JAMA*, 307(23): 2526–2533, 2012.

[31] Kunihiko Fukushima. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, 36(4):193–202, 1980.

[32] Thanos Gentimis, Ala' Alnaser, Alex Durante, Kyle Cook, and Robert Steele. Predicting Hospital Length of Stay Using Neural Networks on MIMIC III Data. In *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing*, pages 1194–1201, 2017.

[33] Justin Gilmer, Samuel Schoenholz, Patrick Riley, Oriol Vinyals, and George Dahl. Neural Message Passing for Quantum Chemistry. In *ICML'17*, page 1263–1272, 2017.

[34] Armand Girbes and Harm-Jan de Grooth. Time to stop randomized and large pragmatic trials for intensive care medicine syndromes: the case of sepsis and acute respiratory distress syndrome. *J Thorac Dis*, 12(Suppl 1):S101–S109, 2020.

[35] Ary Goldberger, Luis Amaral, Leon Glass, Jeffrey Hausdorff, Plamen Ivanov, Roger Mark, Joseph Mietus, George Moody, Chung-Kang Peng, and Eugene Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation*, 101(23):e215–e220, 2000.

[36] Jen Gong, Tristan Naumann, Peter Szolovits, and John Guttag. Predicting Clinical Outcomes Across Changing Electronic Health Record Systems. In *KDD*, 2017.

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[38] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks*, volume 2, pages 729–734. IEEE, 2005.

[39] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, 187:104816, 2020.

[40] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5): 602–610, 2005. IJCNN 2005.

[41] David Gruenberg, Wayne Shelton, Susannah Rose, Ann Rutter, Sophia Socaris, and Glenn McGee. Factors Influencing Length of Stay in the Intensive Care Unit. *American Journal of Critical Care*, 15(5):502–509, 09 2006.

[42] Joel Grunhut, Adam Wyatt, and Oge Marques. Educating Future Physicians in Artificial Intelligence (AI): An Integrative Review and Proposed Changes. *Journal of Medical Education and Curricular Development*, 8:23821205211036836, 2021.

[43] Çaglar Gülçehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. Noisy Activation Functions. *CoRR*, abs/1603.00391, 2016.

[44] Neil Halpern and Stephen Pastores. Critical Care Medicine Beds, Use, Occupancy, and Costs in the United States: A Methodological Review. *Critical care medicine*, 43(11):2452–2459, 2015.

[45] William Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *NIPS*, 2017.

[46] Hrayr Harutyunyan, Hrant Khachatrian, David Kale, Greg Ver Steeg, and Aram Galstyan. Multitask Learning and Benchmarking with Clinical Time Series Data. *Scientific Data*, 6(96), 2019.

[47] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-Inspired Artificial Intelligence. *Neuron*, 95(2):245–258, 2017.

[48] Mahmud Hassan, Howard Tuckman, Robert Patrick, David Kountz, and Jennifer Kohn. Hospital Length of Stay and Probability of Acquiring Infection. *International Journal of Pharmaceutical and Healthcare Marketing*, 4:324–338, 2010.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385, 2015.

[50] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[51] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Weinberger. Densely Connected Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[52] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR, 2015.

[53] Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. Medical Information Mart for Intensive Care IV. `https://doi.org/10.13026/a3wn-hq05`, 2020.

[54] Ida Joiner. Chapter 1 - Artificial Intelligence: AI is Nearby. In *Emerging Library Technologies*, Chandos Information Professional Series, pages 1–22. Chandos Publishing, 2018.

[55] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural Machine Translation in Linear Time. *CoRR*, abs/1610.10099, 2016.

[56] Amit Kalra, Robert Fisher, and Peter Axelrod. Decreased Length of Stay and Cumulative Hospitalized Days despite Increased Patient Admissions and Readmissions in an Area of Urban Poverty. *J Gen Intern Med.*, 25(9):920–935, 2010.

[57] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014.

[58] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint: 1609.02907*, 2016.

[59] John Laffey and Brian Kavanagh. Negative trials in critical care: why most research is probably wrong. *Lancet Respir Med*, 6(9):659–660, 2018.

[60] Kevin Laupland, Andrew Kirkpatrick, John Kortbeek, and Danny Zuege. Long-term Mortality Outcome Associated With Prolonged Admission to the ICU. *Chest*, 129 (4):954 – 959, 2006.

[61] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436, 2015.

[62] Changhee Lee and Mihaela van der Schaar. Temporal Phenotyping using Deep Predictive Clustering of Disease Progression, 2020. URL `https://arxiv.org/abs/2006.08600`.

[63] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. *arXiv preprint: 1707.01926*, 2018.

[64] Min Lin, Qiang Chen, and Shuicheng Yan. Network In Network, 2013.

[65] Zachary Lipton, David Kale, Charles Elkan, and Randall Wetzel. Learning to Diagnose with LSTM Recurrent Neural Networks. *CoRR*, abs/1511.03677, 2015.

[66] Andrew Maas, Awni Hannun, and Andrew Ng. Rectifier nonlinearities improve neural network acoustic models. In *In ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[67] Gregory Mak, William Grant, James McKenzie, and John Mccabe. Physicians' Ability to Predict Hospital Length of Stay for Patients Admitted to the Hospital from the Emergency Department. In *Emergency medicine international*, 2012.

[68] Brandon Malone, Alberto Garcia-Duran, and Mathias Niepert. Learning representations of missing data for predicting patient outcomes. *arXiv preprint: 1811.04752*, 2018.

[69] Kusum Mathews and Elisa Long. A Conceptual Framework for Improving Critical Care Patient Flow and Bed Use. *Annals of the American Thoracic Society*, 12(6): 886–894, 2015.

[70] Riccardo Miotto, Li Li, Brian Kidd, and Joel Dudley. Deep Patient: An Unsupervised Representation to Predict the Future of Patients from the Electronic Health Records. *Scientific Reports*, 6(1):26094, 2016.

[71] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model Cards for Model Reporting. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2019.

[72] Andrew Moon, Joseph Cosgrove, Dawn Lea, Amie Fairs, and David Cressey. An eight year audit before and after the introduction of modified early warning score (MEWS) charts, of patients admitted to a tertiary referral intensive care unit after CPR. *Resuscitation*, 82(2):150 – 154, 2011.

[73] Mostafa Mousavi, William Ellsworth, Weiqiang Zhu, Lindsay Chuang, and Gregory Beroza. Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking. *Nature Communications*, 11(1):3952, 2020.

[74] Jan Máca, Ondřej Jor, Michal Holub, Peter Sklienka, Filip Burša, Michal Burda, Vladimír Janout, and Pavel Ševčík. Past and Present ARDS Mortality Rates: A Systematic Review. *Respiratory Care*, 62(1):113–122, 2017.

[75] Bret Nestor, Matthew McDermott, Geeticka Chauhan, Tristan Naumann, Michael Hughes, Anna Goldenberg, and Marzyeh Ghassemi. Rethinking Clinical Prediction: Why Machine Learning must Consider Year of Care and Feature Aggregation. *CoRR*, abs/1811.12583, 2018.

[76] NHS Digital. DCB0084: OPCS-4.9 Requirements Specification, 2019.

[77] Jeeheh Oh, Jiaxuan Wang, and Jenna Wiens. Learning to Exploit Invariances in Clinical Time-Series Data using Sequence Transformer Networks. In *Proceedings of the 3rd Machine Learning for Healthcare Conference*, volume 85 of *Proceedings of Machine Learning Research*, pages 332–347, Palo Alto, California, 17–18 Aug 2018. PMLR.

[78] OpenAI. GPT-4 Technical Report, 2023.

[79] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. *arXiv preprint: 1902.10191*, 2019.

[80] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[81] Aman Patel, Daniel Mas Montserrat, Carlos Bustamante, and Alexander Ioannidis. Hyperbolic geometry-based deep learning methods to produce population trees from genotype data. *bioRxiv*, 2022.

[82] Michael Pencina, Benjamin Goldstein, Ann Marie Navar, and John Ioannidis. Opportunities and challenges in developing risk prediction models with electronic health records data: a systematic review. *Journal of the American Medical Informatics Association*, 24(1):198–208, 2016.

[83] Tom Pollard, Alistair Johnson, Jesse Raffa, Leo Celi, Roger Mark, and Omar Badawi. The eICU Collaborative Research Database, A Freely Available Multi-Center Database for Critical Care Research. *Scientific Data*, 5(1):180178, 2018.

[84] Jill Poole, Cliona McDowell, Ranjit Lall, Geraldine Perkins, Daniel McAuley, Fangbo Gao, and Duncan Young. Individual patient data analysis of tidal volumes used in three large randomized control trials involving patients with acute respiratory distress syndrome. *BJA: British Journal of Anaesthesia*, 118(4):570–575, 2017.

[85] Niranjani Prasad, Li-Fang Cheng, Corey Chivers, Michael Draugelis, and Barbara Engelhardt. A Reinforcement Learning Approach to Weaning of Mechanical Ventilation in Intensive Care Units. *CoRR*, abs/1704.06300, 2017.

[86] Sanjay Purushothama, Chuizheng Meng, Zhengping Che, and Yan Liu. Benchmarking Deep Learning Models on Large Healthcare Datasets. *Journal of Biomedical Informatics*, 83:112–134, 2018.

[87] Alvin Rajkomar, Eyal Oren, Kai Chen, et al. Scalable and accurate deep learning with electronic health records. *Nature*, 1(1):18, 2018.

[88] John Rapoport, Daniel Teres, Yonggang Zhao, and Stanley Lemeshow. Length of Stay Data as a Guide to Hospital Economic Performance for ICU Patients. *Medical Care*, 41:386–397, 2003.

[89] Narges Razavian and David Sontag. Temporal convolutional neural networks for diagnosis from lab tests. *CoRR*, abs/1511.07938, 2015.

[90] Narges Razavian, Jake Marcus, and David Sontag. Multi-task prediction of disease onsets from longitudinal laboratory tests. In *Proceedings of the 1st Machine Learning for Healthcare Conference*, volume 56 of *Proceedings of Machine Learning Research*, pages 73–100, Northeastern University, Boston, MA, USA, 18–19 Aug 2016. PMLR.

[91] Tony Robinson and Frank Fallside. The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Engineering Department, Cambridge University, Cambridge, UK, 1987.

[92] Emma Rocheteau. On the role of artificial intelligence in psychiatry. *The British Journal of Psychiatry*, page 1–4, 2022.

[93] Emma Rocheteau and Doyoon Kim. Deep Transfer Learning for Automated Diagnosis of Skin Lesions from Photographs, 2020.

[94] Emma Rocheteau, Pietro Liò, and Stephanie Hyland. Predicting Length of Stay in the Intensive Care Unit with Temporal Pointwise Convolutional Networks, 2020.

[95] Emma Rocheteau, Pietro Liò, and Stephanie Hyland. Temporal Pointwise Convolutional Networks for Length of Stay Prediction in the Intensive Care Unit. In *Proceedings of the Conference on Health, Inference, and Learning*, CHIL '21, page 58–68, New York, NY, USA, 2021. Association for Computing Machinery.

[96] Emma Rocheteau, Ioana Bica, Pietro Lio, and Ari Ercole. Dynamic outcomes-based clustering of disease trajectory in mechanically ventilated patients. In *NeurIPS 2022 Workshop on Learning from Time Series for Health*, 2022.

[97] Yulia Rubanova, Ricky Chen, and David Duvenaud. Latent ODEs for Irregularly-Sampled Time Series. *NeurIPS*, 2019.

[98] Sebastian Ruder. An Overview of Multi-Task Learning in Deep Neural Networks. *CoRR*, abs/1706.05098, 2017.

[99] David Rumelhart, Geoffrey Hinton, and Ronald Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[100] Alexander Rusanov, Patric Prado, and Chunhua Weng. Unsupervised time-series clustering over lab data for automatic identification of uncontrolled diabetes. In *2016 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 72–80, 2016.

[101] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[102] Jason Schoettler. Investors are piling into healthcare ai startups; there's room for more, 2021. URL `https://www.venturecapitaljournal.com/investors-are-piling-into-healthcare-ai-start-ups-theres-room-for-more/`.

[103] Jens Schrodt, Aleksei Dudchenko, Petra Knaup-Gregori, and Matthias Ganzinger. Graph-Representation of Patient Data: a Systematic Literature Review. *Journal of Medical Systems*, 44(4):86, 2020. ISSN 1573-689X. doi: 10.1007/s10916-020-1538-4.

[104] Mark Sendak, Madeleine Clare Elish, Michael Gao, Joseph Futoma, William Ratliff, Marshall Nichols, Armando Bedoya, Suresh Balu, and Cara O'Brien. "The Human Body is a Black Box": Supporting Clinical Decision-Making with Deep Learning. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, page 99–109, New York, NY, USA, 2020. Association for Computing Machinery.

[105] Mark Sendak, Michael Gao, Nathan Brajer, and Suresh Balu. Presenting machine learning model information to clinical end users with model facts labels. *npj Digital Medicine*, 3(1):41, 2020.

[106] Seyedmostafa Sheikhalishahi, Vevake Balaraman, and Venet Osmani. Benchmarking Machine Learning Models on eICU Critical Care Dataset, 2019.

[107] Benjamin Shickel, Tyler Loftus, Lasith Adhikari, Tezcan Ozrazgat-Baslanti, Azra Bihorac, and Parisa Rashidi. DeepSOFA: A Continuous Acuity Score for Critically Ill Patients using Clinically Interpretable Deep Learning. In *Scientific Reports*, 2019.

[108] Beth Smith, Joseph Chiovaro, Maya O'Neil, et al. Early Warning System Scores for Clinical Deterioration in Hospitalized Patients: A Systematic Review. *Annals of the American Thoracic Society*, 11(9):1454–1465, 2014.

[109] Huan Song, Deepta Rajan, Jayaraman Thiagarajan, and Andreas Spanias. Attend and Diagnose: Clinical Time Series Analysis using Attention Models. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 4091–4098, 2018.

[110] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[111] Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the Impact of Feature Attribution Baselines. *Distill*, 5(1):e22, 2020.

[112] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328, 2017.

[113] Harini Suresh, Nathan Hunt, Alistair Johnson, Leo Celi, Peter Szolovits, and Marzyeh Ghassemi. Clinical Intervention Prediction and Understanding with Deep Neural Networks. In *MLHC*, 2017.

[114] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. *CoRR*, abs/1409.4842, 2014.

[115] Patrick Thoral, Ronald Driessen, and Tariq Dam. AmsterdamUMCdb. *GitHub*, 2020. URL `https://github.com/AmsterdamUMC/AmsterdamUMCdb`.

[116] Patrick Thoral, Jan Peppink, Ronald Driessen, Eric Sijbrands, Erwin Kompanje, Lewis Kaplan, Heatherlee Bailey, Jozef Kesecioglu, Maurizio Cecconi, Matthew Churpek, Gilles Clermont, Mihaela van der Schaar, Ari Ercole, Armand Girbes, and Paul Elbers. Sharing ICU Patient Data Responsibly Under the Society of Critical Care Medicine/European Society of Intensive Care Medicine Joint Data Science Collaboration: The Amsterdam University Medical Centers Database (AmsterdamUMCdb) Example. *Critical Care Medicine*, 49(6), 2021.

[117] Nenad Tomašev, Xavier Glorot, Jack Rae, et al. A Clinically Applicable Approach to Continuous Prediction of Future Acute Kidney Injury. *Nature*, 572(7767):116–119, 2019.

[118] Sana Tonekaboni, Mjaye Mazwi, Peter Laussen, Danny Eytan, Robert Greer, Sebastian Goodfellow, Andrew Goodwin, Michael Brudno, and Anna Goldenberg. Prediction of Cardiac Arrest from Physiological Signals in the Pediatric ICU. In *MLHC*, 2018.

[119] Catherine Tong, Emma Rocheteau, Petar Veličković, Nicholas Lane, and Pietro Liò. *Predicting Patient Outcomes with Graph Representation Learning*, pages 281–293. Springer International Publishing, Cham, 2022.

[120] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *CoRR*, abs/1609.03499, 2016.

[121] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010. Curran Associates Inc., 2017.

[122] Petar Veličković. Tikz, 2017. URL `https://github.com/PetarV-/TikZ/tree/master/Long%20short-term%20memory`.

[123] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, 2018.

[124] Jean-Louis Vincent. We should abandon randomized controlled trials in the intensive care unit. *Crit Care Med*, 38(10 Suppl):S534–538, 2010.

[125] David Wallace, Derek Angus, Christopher Seymour, Amber Barnato, and Jeremy Kahn. Critical Care Bed Growth in the United States. A Comparison of Regional and National Trends. *American Journal of Respiratory and Critical Care Medicine*, 191(4):410–416, 2015.

[126] Yanshan Wang, Yiqing Zhao, Terry Therneau, Elizabeth Atkinson, Ahmad Tafti, Nan Zhang, Shreyasee Amin, Andrew Limper, Sundeep Khosla, and Hongfang Liu. Unsupervised machine learning for the discovery of latent disease clusters and patient subgroups using electronic health records. *Journal of Biomedical Informatics*, 102: 103364, 2020.

[127] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay Sarma, Michael Bronstein, and Justin Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions On Graphics*, 38(5):1–12, 2019.

[128] Zhongyuan Wang, Peng Yi, Kui Jiang, Junjun Jiang, Zhen Han, Tao Lu, and Jiayi Ma. Multi-Memory Convolutional Neural Network for Video Super-Resolution. *IEEE Transactions on Image Processing*, PP:1–1, 12 2018. doi: 10.1109/TIP.2018.2887017.

[129] World Health Organisation. *ICD-10: International Statistical Classification of Diseases and Related Health Problems*, volume 10th Revision. World Health Organision, 2011.

[130] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, 2019.

[131] Hugo Yèche, Gideon Dresdner, Francesco Locatello, Matthias Hüser, and Gunnar Rätsch. Neighborhood Contrastive Learning Applied to Online Patient Monitoring. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11964–11974. PMLR, 18–24 Jul 2021.

[132] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal Graph Convolutional Neural Network: A Deep Learning Framework for Traffic Forecasting. *arXiv preprint: 1709.04875*, 2017.

[133] Joe Zhang, Stephen Whebell, Jack Gallifant, Sanjay Budhdeo, Heather Mattie, Piyawat Lertvittayakumjorn, Maria del Pilar Arias Lopez, Beatrice Tiangco, Judy Gichoya, Hutan Ashrafian, Leo Celi, and James Teo. An interactive dashboard to track themes, development maturity, and global equity in clinical artificial intelligence research. *The Lancet Digital Health*, 4(4):e212–e213, 2022.

[134] Xi Zhang, Jingyuan Chou, Jian Liang, Cao Xiao, Yize Zhao, Harini Sarva, Claire Henchcliffe, and Fei Wang. Data-Driven Subtyping of Parkinson's Disease Using Longitudinal Clinical Records: A Cohort Study. *Scientific Reports*, 9(1):797, 2019.

[135] David Zimmerer, Jens Petersen, Gregor Köhler, Jakob Wasserthal, Tim Adler, Sebastian Wirkert, and Tobias Ross. trixi - Training and Retrospective Insight eXperiment Infrastructure. `https://github.com/MIC-DKFZ/trixi`, 2017.

[136] Jack Zimmerman, Andrew Kramer, Douglas McNair, and Fern Malila. Acute Physiology and Chronic Health Evaluation (APACHE) IV: Hospital Mortality Assessment for Today's Critically Ill Patients. *Critical Care Medicine / Society of Critical Care Medicine*, 34(5), 2006.

APPENDIX A

Temporal Pointwise Convolution

A.1 Hyperparameter Search Methodology and Implementation Details

The TPC and baseline models have hyperparameters that can broadly be split into three categories: time series specific, non-time series specific and global parameters (shown in more detail in Tables A.1, A.2 and A.3). The hyperparameter search ranges have been included in Table A.4.

First, I ran 25 randomly sampled hyperparameter trials on the TPC model to decide the non-time series specific parameters (diagnosis embedding size, final fully connected layer size, batch normalisation strategy, dropout rate and the parameter α) keeping all other parameters fixed. These parameters (indicated by stars) remained fixed for all the models which share their non-time series specific architecture (NB. the best value for α was 100 – not shown in the Tables).

I then ran 50 hyperparameter trials to optimise the remaining parameters for the TPC, standard LSTM, and Transformer models. To train the channel-wise LSTM and the temporal model with weight sharing, I ran a further 10 trials to re-optimise the hidden size (8 per feature) and number of temporal channels (32 channels shared across all features) respectively. For all other ablation studies and variations of each model, I kept the same hyperparameters where applicable (see Table 3.4 for a full list of all of the models). The number of epochs was determined by selecting the best validation performance from a model trained over 50 epochs. This was different for each model. For eICU this was 8 (LSTM), 30 (CW LSTM), 15 (Transformer) and 15 (TPC). For MIMIC-IV this was 8 (LSTM), 20 (CW LSTM), 15 (Transformer) and 10 (TPC). I noted that the best LSTM hyperparameters (Table A.2) were similar to that found in Sheikhalishahi et al. [106].

All deep learning methods were implemented in PyTorch [80] and were optimised using

Table A.1: The TPC model has 11 hyperparameters (Main Dropout and Batch Normalisation have been repeated in the table because they apply to multiple parts of the model). I allowed the model to optimise a custom dropout rate for the temporal convolutions because they have fewer parameters and might need less regularisation than the rest of the model. The best hyperparameter values are shown in brackets (eICU/MIMIC-IV). Hyperparameters marked with * were fixed across all of the models.

TPC Specific	
Temporal Specific	**Pointwise Specific**
Temp. Channels (12/11) Temp. Dropout (0.05/0.05) Kernel Size (4/5)	Point. Channels (13/5) Main Dropout* (0.45/0)
Batch Normalisation* (True/True) No. TPC Layers (9/8)	
Non-TPC Specific	**Global Parameters**
Diag. Embedding Size* (64/-) Main Dropout* (0.45/0) Final FC Layer Size* (17/36) Batch Normalisation* (True/True)	Batch Size (32/8) Learning Rate (0.00226/ 0.00221)

Table A.2: The LSTM model has 9 hyperparameters. I allowed the model to optimise a custom dropout rate for the LSTM layers. Note that batch normalisation is not applicable to the LSTM layers. The best parameters are shown as (eICU/MIMIC-IV).

LSTM Specific	**Non-LSTM Specific**
Hidden State (128/128) LSTM Dropout (0.2/0.25) No. LSTM Layers (2/1)	Diag. Embedding Size* (64/-) Main Dropout* (0.45/0) Final FC Layer Size* (17/36) Batch Normalisation* (True/True)
Global Parameters	
Batch Size (512/32) Learning Rate (0.00129/0.00163)	

Adam [57]. The data (including decay indicators) and the non-time series components of the models were the same as in TPC (Figure 3.5). I used trixi to structure our experiments and compare different hyperparameter choices [135].

The experiments were performed using resources provided by the Cambridge Tier-2 system operated by the University of Cambridge Research Computing Service (`www.hpc.cam.ac.uk`) funded by EPSRC Tier-2 capital grant EP/P020259/1.

A.1.1 Transformer

The Transformer is a self-attention model, originally designed for sequence-to-sequence tasks in natural language processing (more detail in provided in Section 2.5.1). It consists of both an encoder and decoder, however I only used the former. Our implementation is the same as the original encoder in Vaswani et al. [121], except that I added temporal masking to impose causality i.e. the current representation can only depend on current or earlier time points, and I omitted the positional encodings because they were not found to be helpful. This is probably because I already had a feature to indicate the position in the time series (Section 3.5.3.3).

Table A.3: The Transformer model has 12 hyperparameters. I allowed the model to optimise a custom dropout rate for the Transformer layers. The positional encoding hyperparameter is binary; it determines whether or not I used the original positional encodings proposed by Vaswani et al. [121]. Note that batch normalisation is not applicable to the Transformer layers (the default implementation uses layer normalisation). The best parameters are shown as (eICU/MIMIC-IV).

Transformer Specific	**Non-Transformer Specific**
No. Attention Heads (2/1)	Diag. Embedding Size* (64/-)
Feedforward Size (256/64)	Main Dropout* (0.45/0)
d_{model} (16/32)	Final FC Layer Size* (17/36)
Transformer Dropout (0/0.05)	/True)
No. Transformer Layers (6/2)	
Global Parameters	
Batch Size (32/64)	
Learning Rate (0.00017/0.00129)	

Table A.4: Hyperparameter Search Ranges. I took a random sample from each range and converted to an integer if necessary. For the kernel sizes (not shown in the table) the range was dependent on the number of TPC layers selected (because large kernel sizes combined with a large number of layers can have an inappropriately wide range as the dilation factor increases per layer). In general the range of kernel sizes was around 2-5 (but it could be up to 10 for small numbers of TPC Layers).

Hyperparameter	**Lower**	**Upper**	**Scale**
Batch Size	4	512	$\log_2$
Dropout Rate (all)	0	0.5	Linear
Learning Rate	0.0001	0.01	$\log_{10}$
Batch Normalisation	True	False	
Positional Encoding	True	False	
Diagnosis Embedding Size	16	64	$\log_2$
Final FC Layer Size	16	64	$\log_2$
CW LSTM Hidden State Size	4	16	$\log_2$
Point. Channels	4	16	$\log_2$
Temp. Channels	4	16	$\log_2$
Temp. Channels (weight sharing)	16	64	$\log_2$
LSTM Hidden State Size	16	256	$\log_2$
d_{model}	16	256	$\log_2$
Feedforward Size	16	256	$\log_2$
No. Attention Heads	2	16	$\log_2$
No. TPC Layers	1	12	Linear
No. LSTM Layers	1	4	Linear
No. Transformer Layers	1	10	Linear

A.2 Additional Results and Figures

Table A.5: The effect of changing the size of the training data on the LSTM, CW LSTM, Transformer, and TPC model performance in the eICU data set. A hundred percent of the training set represents 102,712 ICU stays, 50% is 51,356, 25% is 25,678, 12.5% is 12,839, and 6.25% is 6,420 stays.

Model (% train data)	**MAD**	**MAPE**	**MSE**	**MSLE**	R^2	**Kappa**
LSTM (100)	2.39±0.00	118.2±1.1	26.9±0.1	1.47±0.01	0.09±0.00	0.28±0.00
LSTM (50)	2.41±0.01	129.9±1.9	26.2±0.2	1.52±0.00	0.11±0.01	0.31±0.01
LSTM (25)	2.44±0.01	126.8±2.5	27.2±0.3	1.58±0.00	0.08±0.01	0.27±0.01
LSTM (12.5)	2.48±0.01	137.4±3.4	27.4±0.2	1.65±0.01	0.07±0.01	0.27±0.01
LSTM (6.25)	2.52±0.02	135.9±3.3	28.0±0.8	1.71±0.02	0.05±0.03	0.26±0.03
CW LSTM (100)	2.37±0.00	114.5±0.4	26.6±0.1	1.43±0.00	0.10±0.00	0.30±0.00
CW LSTM (50)	2.40±0.01	123.4±0.7	26.5±0.1	1.48±0.01	0.10±0.00	0.31±0.00
CW LSTM (25)	2.44±0.00	119.8±1.3	27.2±0.1	1.54±0.00	0.08±0.00	0.29±0.00
CW LSTM (12.5)	2.50±0.01	134.7±1.5	27.7±0.1	1.63±0.01	0.06±0.00	0.28±0.00
CW LSTM (6.25)	2.58±0.01	129.8±3.5	29.0±0.2	1.73±0.01	0.02±0.01	0.25±0.01
Transformer (100)	2.36±0.00	114.1±0.6	26.7±0.1	1.43±0.00	0.09±0.00	0.30±0.00
Transformer (50)	2.39±0.00	120.1±0.6	26.5±0.1	1.48±0.00	0.10±0.00	0.31±0.00
Transformer (25)	2.43±0.01	117.9±1.8	27.2±0.2	1.54±0.01	0.08±0.01	0.28±0.01
Transformer (12.5)	2.48±0.01	128.1±2.3	27.9±0.1	1.62±0.01	0.06±0.00	0.26±0.01
Transformer (6.25)	2.52±0.01	139.7±2.4	27.8±0.1	1.69±0.02	0.06±0.00	0.26±0.00
TPC (100)	1.78±0.02	63.5±4.3	21.7±0.5	0.70±0.03	0.27±0.02	0.58±0.01
TPC (50)	1.95±0.02	72.0±3.1	23.8±0.4	0.87±0.03	0.19±0.01	0.51±0.01
TPC (25)	2.09±0.01	89.0±3.8	24.8±0.3	1.09±0.02	0.16±0.01	0.45±0.01
TPC (12.5)	2.28±0.01	101.4±4.8	27.0±0.4	1.36±0.03	0.08±0.01	0.35±0.02
TPC (6.25)	2.49±0.02	139.9±5.5	28.0±0.3	1.64±0.03	0.05±0.01	0.28±0.01

Table A.6: The effect of training with the mean squared logarithmic error (MSLE) loss function when compared to mean squared error (MSE) on the eICU data set. This is an extension to Table 3.5 (refer to its legend for definitions of the metric acronyms, detailed of CI calculations and meaning of the colour scheme).

Model	MAD	MAPE	MSE	MSLE	R^2	Kappa
LSTM (MSE)	2.57±0.03	235.2±6.2	**24.5±0.2****	1.97±0.02	**0.17±0.01****	**0.28±0.01**
LSTM (MSLE)	**2.39±0.00****	**118.2±1.1****	26.9±0.1	**1.47±0.01****	0.09±0.00	**0.28±0.00**
CW LSTM (MSE)	2.56±0.01	218.5±4.0	**24.2±0.1****	1.84±0.02	**0.18±0.00****	**0.34±0.01****
CW LSTM (MSLE)	**2.37±0.00****	**114.5±0.4****	26.6±0.1	**1.43±0.00****	0.10±0.00	0.30±0.00
Transformer (MSE)	2.51±0.01	212.7±5.2	**24.7±0.2****	1.87±0.03	**0.16±0.01****	0.28±0.01
Transformer (MSLE)	**2.36±0.00****	**114.1±0.6****	26.7±0.1	**1.43±0.00****	0.09±0.00	**0.30±0.00****
TPC (MSE)	2.21±0.02	154.3±10.1	**21.6±0.2**	1.80±0.10	**0.27±0.01**	0.47±0.01
TPC (MSLE)	**1.78±0.02****	**63.5±4.3****	21.7±0.5	**0.70±0.03****	**0.27±0.02**	**0.58±0.01****

Table A.7: eICU multitask results. I compared the performance of each model on individual tasks (LoS or mortality prediction) to the multitask setting (both LoS and mortality). The results from Table 3.4 are repeated here for ease of comparison. Note that the 'mean' and 'median' models are only for LoS – there is no equivalent model for mortality prediction.

	In-Hospital Mortality		Length of Stay					
Model	AUROC	AUPRC	MAD	MAPE	MSE	MSLE	R^2	Kappa
Mean	–	–	3.21	395.7	29.5	2.87	0.00	0.00
Median	–	–	2.76	184.4	32.6	2.15	-0.11	0.00
	0.849±0.002	0.407±0.012	–	–	–	-	-	-
LSTM	–	–	2.39±0.00	118.2±1.1	**26.9±0.1***	**1.47±0.01**	**0.09±0.00***	**0.28±0.00**
	0.852±0.003	**0.436±0.007****	**2.40±0.01**	**116.5±0.8***	27.2±0.2	**1.47±0.01**	0.08±0.01	**0.28±0.01**
	0.855±0.001	0.464±0.004	–	–	–	–	–	–
CW LSTM	–	–	**2.37±0.00**	**114.5±0.4**	**26.6±0.1***	**1.43±0.00***	**0.10±0.00***	**0.30±0.00**
	0.865±0.002**	**0.490±0.007****	**2.37±0.00**	115.0±0.7	26.8±0.1	1.44±0.00	0.09±0.00	**0.30±0.00**
	0.851±0.002	0.454±0.005	–	–	–	–	–	–
Transformer	–	–	**2.36±0.00**	**114.1±0.6**	26.7±0.1	**1.43±0.00**	0.09±0.00	**0.30±0.00**
	0.858±0.001**	**0.475±0.004****	**2.36±0.00**	114.2±0.7	**26.6±0.1**	**1.43±0.00**	**0.10±0.00**	**0.30±0.00**
	0.864±0.001	0.508±0.005	–	–	–	–	–	–
TPC	–	–	1.78±0.02	63.5±3.8	21.8±0.5	0.71±0.03	0.26±0.02	0.58±0.01
	0.865±0.002	**0.523±0.006****	**1.55±0.01****	**46.4±2.6****	**18.7±0.2****	**0.40±0.02****	**0.37±0.01****	**0.70±0.00****

Table A.8: MIMIC-IV multitask results.

Model	In-Hospital Mortality AUROC	In-Hospital Mortality AUPRC	Length of Stay MAD	Length of Stay MAPE	Length of Stay MSE	Length of Stay MSLE	Length of Stay R^2	Length of Stay Kappa
Mean	–	–	5.24	474.9	77.7	2.80	0.00	0.00
Median	–	–	4.60	216.8	86.8	2.09	-0.12	0.00
LSTM	**0.895±0.001**	**0.657±0.003**	–	–	–	–	–	–
	–	–	**3.68±0.02**	**107.2±3.1**	**65.7±0.7**	1.26±0.01	**0.15±0.01**	**0.43±0.01**
	0.896±0.002	**0.659±0.004**	**3.66±0.01**	**106.8±2.7**	**65.3±0.6**	**1.25±0.01***	**0.16±0.01**	**0.44±0.00**
CW LSTM	**0.897±0.002**	**0.650±0.005**	–	–	–	–	–	–
	–	–	**3.68±0.02**	**107.0±1.8**	**66.4±0.6**	**1.23±0.01**	**0.15±0.01**	**0.43±0.00**
	0.899±0.002	**0.654±0.003**	**3.69±0.02**	**107.2±1.6**	**66.3±0.6**	**1.23±0.01**	**0.15±0.01**	**0.44±0.00**
Transformer	0.890±0.002	0.641±0.008	–	–	–	–	–	–
	–	–	**3.62±0.02**	**113.8±1.8**	**63.4±0.5**	**1.21±0.01**	**0.18±0.01**	**0.45±0.00**
	0.898±0.001**	**0.656±0.005***	**3.61±0.01**	**112.3±2.0**	**63.3±0.3**	**1.20±0.01**	**0.19±0.00**	**0.45±0.00**
TPC	0.905±0.001	0.691±0.006	–	–	–	–	–	–
	–	–	2.39±0.03	47.6±1.4	46.3±1.3	0.39±0.02	0.40±0.02	0.78±0.01
	0.918±0.002**	**0.713±0.007****	**2.28±0.07***	**32.4±1.2****	**42.0±1.2****	**0.19±0.00****	**0.46±0.02****	**0.85±0.00****

Table A.9: eICU time series features. ‘Time in the ICU’ and ‘Time of day’ were not part of the tables in eICU but were added later as helpful indicators to the model.

Source Table			
lab			***respiratorycharting***
-basos	MPV	glucose	Exhaled MV
-eos	O2 Sat (%)	lactate	Exhaled TV (patient)
-lymphs	PT	magnesium	LPM O2
-monos	PT - INR	pH	Mean Airway Pressure
-polys	PTT	paCO2	Peak Insp. Pressure
ALT (SGPT)	RBC	paO2	PEEP
AST (SGOT)	RDW	phosphate	Plateau Pressure
BUN	WBC x 1000	platelets x 1000	Pressure Support
Base Excess	albumin	potassium	RR (patient)
FiO2	alkaline phos.	sodium	SaO2
HCO3	anion gap	total bilirubin	TV/kg IBW
Hct	bedside glucose	total protein	Tidal Volume (set)
Hgb	bicarbonate	troponin - I	Total RR
MCH	calcium	urinary specific gravity	Vent Rate
MCHC	chloride		
MCV	creatinine		
nursecharting	***vitalperiodic***	***vitalaperiodic***	**N/A**
Bedside Glucose	cvp	noninvasivediastolic	Time in the ICU
Delirium Scale/Score	heartrate	noninvasivemean	Time of day
Glasgow coma score	respiration	noninvasivesystolic	
Heart Rate	sao2		
Invasive BP	st1		
Non-Invasive BP	st2		
O2 Admin Device	st3		
O2 L/%	systemicdiastolic		
O2 Saturation	systemicmean		
Pain Score/Goal	systemicsystolic		
Respiratory Rate	temperature		
Sedation Score/Goal			
Temperature			

Table A.10: MIMIC-IV time series features.

	Source Table	
chartevents		
Activity / Mobility (JH-HLM)	Mean Airway Pressure	Resp Alarm - High
Apnea Interval	Minute Volume	Resp Alarm - Low
Arterial Blood Pressure Alarm - High	Minute Volume Alarm - High	Respiratory Rate
Arterial Blood Pressure Alarm - Low	Minute Volume Alarm - Low	Respiratory Rate (Set)
Arterial Blood Pressure diastolic	Non Invasive Blood Pressure diastolic	Respiratory Rate (Total)
Arterial Blood Pressure mean	Non Invasive Blood Pressure mean	Respiratory Rate (spont)
Arterial Blood Pressure systolic	Non Invasive Blood Pressure systolic	Richmond-RAS Scale
Braden Score	Non-Invasive Blood Pressure Alarm - High	Strength L Arm
Current Dyspnea Assessment	Non-Invasive Blood Pressure Alarm - Low	Strength L Leg
Daily Weight	O2 Flow	Strength R Arm
Expiratory Ratio	O2 Saturation Pulseoxymetry Alarm - Low	Strength R Leg
Fspn High	O2 saturation pulseoxymetry	Temperature Fahrenheit
GCS - Eye Opening	PEEP set	Tidal Volume (observed)
GCS - Motor Response	PSV Level	Tidal Volume (set)
GCS - Verbal Response	Pain Level	Tidal Volume (spont)
Glucose finger stick (range 70-100)	Pain Level Response	Total PEEP Level
Heart Rate	Paw High	Ventilator Mode
Heart Rate Alarm - Low	Peak Insp. Pressure	Vti High
Heart rate Alarm - High	Phosphorous	
Inspired O2 Fraction	Plateau Pressure	
labevents		**N/A**
Alanine Aminotransferase (ALT)	MCHC	Time in the ICU
Alkaline Phosphatase	MCV	Time of day
Anion Gap	Magnesium	
Asparate Aminotransferase (AST)	Oxygen Saturation	
Base Excess	PT	
Bicarbonate	PTT	
Bilirubin, Total	Phosphate	
Calcium, Total	Platelet Count	
Calculated Total CO2	Potassium	
Chloride	Potassium, Whole Blood	
Creatinine	RDW	
Free Calcium	RDW-SD	
Glucose	Red Blood Cells	
H	Sodium	
Hematocrit	Sodium, Whole Blood	
Hematocrit, Calculated	Temperature	
Hemoglobin	Urea Nitrogen	
I	White Blood Cells	
INR(PT)	pCO2	
L	pH	
Lactate	pO2	
MCH		

APPENDIX B

Graph Representation Learning

B.1 Hyperparameter Search Methodology and Implementation Details

For each model, we conducted 10 random hyperparameter trials. The selected hyperparameters for the main set of results in Table 4.4 can be found in Tables B.1, B.2, B.3 and B.4. The hyperparameter search ranges can be seen in Table B.5. The hyperparameters for the dynamic models (Table 4.5) and ablation studies (Table 4.6) can be found in our code repository: `https://github.com/EmmaRocheteau/eICU-GNN-LSTM/blob/master/src/hyperparameters/best_parameters.py`.

All deep learning methods were implemented in PyTorch and optimised using Adam [57]. We used PyTorch Lightning [28] and Tune to structure our experiments and easily compare different hyperparameter choices. The maximum number of epochs was 25, although many models finished before this due to early stopping.

Table B.1: LSTM hyperparameters. The best parameters are shown for both the LoS task and mortality.

Hyperparameter	**LoS**	**Mortality**
Batch Size	32	256
Dropout Rate	0.134	0.039
Learning Rate	0.0010	0.0006
L2 Regularisation	0.00094	0.00002
LSTM Hidden State	64	64
Static Embedding Size	64	64
LSTM Layers	2	2
Final FC Layer Size	32	32

Table B.2: GraphSAGE hyperparameters. The best parameters are shown for both the LoS task and mortality.

Hyperparameter	**LoS**	**Mortality**
Batch Size	256	64
Dropout Rate	0.352	0.252
Learning Rate	0.0007	0.0006
L2 Regularisation	0.00005	0.00002
Neighbour Sample Size 1	10	10
Neighbour Sample Size 2	30	20
GNN Output Dimension	64	64
GNN Hidden Dimension	256	64

Table B.3: GAT hyperparameters. Hyperparameters marked with * were fixed across all of the models.

Hyperparameter	**LoS**	**Mortality**
Batch Size	128	128
Dropout Rate	0.224	0.224
Learning Rate	0.0007	0.0006
L2 Regularisation	0.00033	0.00033
Neighbour Sample Size 1	30	10
Neighbour Sample Size 2	10	20
GNN Output Dimension	256	64
GNN Hidden Dimension	128	64
GAT Attention Heads	12	12
GAT Output Heads	10	10
GAT Feature Dropout	0.454	0.7
GAT Attention Dropout	0.616	0.7

Table B.4: MPNN hyperparameters. Hyperparameters marked with * were fixed across all of the models.

Hyperparameter	**LoS**	**Mortality**
Batch Size	32	32
Dropout Rate	0.486	0.252
Learning Rate	0.0007	0.0006
L2 Regularisation	0.00002	0.00002
Neighbour Sample Size	20	10
GNN Output Dimension	512	64
GNN Hidden Dimension	64	64
MPNN MP Steps	4	4

Table B.5: Hyperparameter Search Ranges.

Hyperparameter	**Lower**	**Upper**	**Scale**
Batch Size	32	256	$\log_2$
Dropout Rate	0	0.5	Linear
Learning Rate	0.0005	0.001	Log-Linear
L2 Regularisation	0.00001	0.001	Log-Linear
Node Sample Size	5	30	Linear
GNN Output Dimension	64	512	$\log_2$
GNN Hidden Dimension	64	512	$\log_2$
LSTM Hidden State	64	512	$\log_2$
Static Embedding Size	64	512	$\log_2$
GCN Dropout	0.2	0.7	Linear
GAT Attention Heads	8	12	Linear
GAT Output Heads	6	10	Linear
GAT Feature Dropout	0.2	0.7	Linear
GAT Attention Dropout	0.2	0.7	Linear
MPNN MP Steps	1	5	Linear
LSTM Layers	1	5	Linear
Final FC Layer Size	32	256	$\log_2$
α	0.5	3	Log-Linear

APPENDIX C

Dynamic Outcomes-Based Clustering

C.1 Hyperparameter Search Methodology and Implementation Details

All the encoders have hyperparameters that can broadly be split into three categories: time series specific, non-time series specific and global parameters (shown in more detail in Tables C.1, C.2 and C.3). The hyperparameter search ranges have been included in Table C.4. The search was conducted very similarly to that described in Section A.1. I ran 10 hyperparameter trials to optimise the remaining parameters for the TPC, LSTM, and Transformer models. The number of epochs was determined by selecting the best validation performance from a model trained over 300 epochs (early stopping was then used for each individual model). All deep learning methods were implemented in PyTorch [80] using PyTorch Lightning [28] and were optimised using Adam [57].

I also optimised for the weighting between the tasks. I simply multiplied the loss for each component by a hyperparameter. The best overall learning curves were found when the task weighting coefficients were: 0.5 for the duration tasks, 1 for the binary tasks, 0.1 for time series reconstruction and forecasting, and 0.002 for binary feature reconstruction. The reason for the small weighting for binary feature reconstruction was that the task appeared very easy for the models, especially predicting the sex of the patient, and so the representation became dominated with this at the expense of the other tasks.

Table C.1: The TPC model has 11 hyperparameters (Main Dropout and Batch Normalisation have been repeated in the table because they apply to multiple parts of the model). I allowed the model to optimise a custom dropout rate for the temporal convolutions because they have fewer parameters and might need less regularisation than the rest of the model. The best hyperparameter values are shown in brackets. Hyperparameters marked with * were fixed across all of the models.

TPC Specific	
Temporal Specific	**Pointwise Specific**
Temp. Channels (6)	Point. Channels (14)
Temp. Dropout (0.05)	Main Dropout* (0.05)
Kernel Size (3)	
Batch Normalisation* (True)	
No. TPC Layers (6)	
Non-TPC Specific	**Global Parameters**
Batch Normalisation* (True)	Batch Size (128)
Main Dropout* (0.05)	Learning Rate (0.0001)
Final FC Layer Size* (16)	Embedding Size (128)

Table C.2: The LSTM model has 8 hyperparameters. I allowed the model to optimise a custom dropout rate for the LSTM layers. Note that batch normalisation is not applicable to the LSTM layers. The best hyperparameter values are shown in brackets. Hyperparameters marked with * were fixed across all of the models.

LSTM Specific	**Non-LSTM Specific**	**Global Parameters**
Hidden State (128)	Batch Normalisation* (True)	Batch Size (128)
LSTM Dropout (0.05)	Main Dropout* (0.05)	Learning Rate (0.0001)
No. LSTM Layers (2)		Embedding Size (128)

Table C.3: The Transformer model has 9 hyperparameters. Note that batch normalisation is not applicable to the Transformer layers (the default implementation uses layer normalisation). The best hyperparameter values are shown in brackets. Hyperparameters marked with * were fixed across all of the models.

Transformer Specific	**Non-Transformer Specific**	**Global Parameters**
No. Attention Heads (2)	Batch Normalisation* (True)	Batch Size (128)
Feedforward Size (256)	Main Dropout* (0.05)	Learning Rate (0.0001)
d_{model} (16)		
Transformer Dropout (0.05)		
No. Transformer Layers (6)		

Table C.4: Hyperparameter Search Ranges. I took a random sample from each range and converted to an integer if necessary. For the kernel sizes (not shown in the table) the range was dependent on the number of TPC layers selected (because large kernel sizes combined with a large number of layers can have an inappropriately wide range as the dilation factor increases per layer). In general the range of kernel sizes was around 2-5 (but it could be up to 10 for small numbers of TPC Layers).

Hyperparameter	**Lower**	**Upper**	**Scale**
Batch Size	4	512	$\log_2$
Dropout Rate (all)	0	0.5	Linear
Learning Rate	0.0001	0.01	$\log_{10}$
Batch Normalisation	True	False	
Final FC Layer Size	16	64	$\log_2$
Point. Channels	4	16	$\log_2$
Temp. Channels	4	16	$\log_2$
LSTM Hidden State Size	16	256	$\log_2$
d_{model}	16	256	$\log_2$
Feedforward Size	16	256	$\log_2$
No. Attention Heads	2	16	$\log_2$
No. TPC Layers	1	12	Linear
No. LSTM Layers	1	4	Linear
No. Transformer Layers	1	10	Linear

C.1.1 Number of Clusters

The value of k was determined using an average value from the elbow method across various encoders. Specifically I looked for the point at which the Within Cluster Sum of Squares (WCSS) started to tail off with increasing values of k. Figure C.1 shows an example elbow plot. I selected the value 5 across all the models.

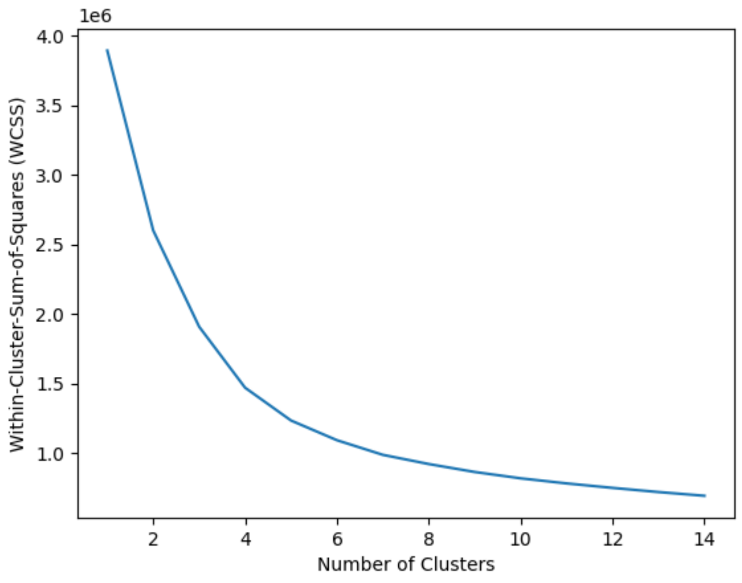


Figure C.1: Elbow plot for the TPC model.

C.2 Additional Results and Figures

Table C.5: Time Series features. The features which do not have a source table were calculated from the other features available in the data. 'Mandatory Ventilation' and 'Patient Triggered' were calculated from the ventilator settings as outlined in Table C.6.

Feature	**Type**	**Source Table**
ABP gemiddeld	Continuous	numericitems
Ademfreq.	Continuous	numericitems
Alb.Chem (bloed)	Continuous	numericitems
Bilirubine (bloed)	Continuous	numericitems
CRP (bloed)	Continuous	numericitems
End tidal CO2 concentratie	Continuous	numericitems
Exp. tidal volume	Continuous	numericitems
Glucose (bloed)	Continuous	numericitems
Hartfrequentie	Continuous	numericitems
Ht (bloed)	Continuous	numericitems
Kalium (bloed)	Continuous	numericitems
Kreatinine (bloed)	Continuous	numericitems
Lactaat (bloed)	Continuous	numericitems
Leuco's (bloed)	Continuous	numericitems
Natrium (bloed)	Continuous	numericitems
O2 concentratie	Continuous	numericitems
P/F ratio	Continuous	
PC	Continuous	numericitems
PEEP (Set)	Continuous	numericitems
PO2 (bloed)	Continuous	numericitems
Piek druk	Continuous	numericitems
Saturatie (Monitor)	Continuous	numericitems
Temp.	Continuous	numericitems
Thrombo's (bloed)	Continuous	numericitems
TroponineT (bloed)	Continuous	numericitems
UrineCAD	Continuous	numericitems
lung compliance	Continuous	
mandatory ventilation	Binary	
pCO2 (bloed)	Continuous	numericitems
pH (bloed)	Continuous	numericitems
patient triggered	Binary	
Time in the ICU	Discrete	

Table C.6: Ventilator Settings Classification, used to produce the features 'Patient Triggered' and 'Mandatory Ventilation' in Table C.5.

Patient Triggered Ventilation	**Mandatory Ventilation**
Bi Vente	MMV
NAVA	VC
PRVC	PC
PRVC (trig)	Pressure Controled
PS/CPAP (trig)	PC (No trig)
SIMV(PC)+PS	PRVC (No trig)
SIMV(VC)+PS	VC (No trig)
VC (trig)	CPPV
VS	IPPV
SIMV_ASB	SIMV
CPAP	BIPAP
BIPAP-SIMV/ASB	
MMV_ASB	
MMV/ASB	
ASB	
IPPV/ASSIST	
CPPV/ASSIST	
CPPV_Assist	
IPPV_Assist	
SIMV/ASB	
CPAP_ASB	
PS/CPAP	
BIPAP/ASB	
CPAP/ASB	