NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

MIGUEL SIMÃO DÓRDIO CARDOSO

Bachelor in Computer Science and Engineering

HEALTH OUTCOME PATHWAY PREDICTION

A GRAPH-BASED FRAMEWORK

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
November, 2021

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

HEALTH OUTCOME PATHWAY PREDICTION

A GRAPH-BASED FRAMEWORK

MIGUEL SIMÃO DÓRDIO CARDOSO

Bachelor in Computer Science and Engineering

Adviser: Flávio Martins
Assistant Professor, Instituto Superior Técnico

Co-advisers: Nuno Manuel Robalo Correia
Full Professor, NOVA University Lisbon

Ana Rita Londral
CEO, Value for Health Colab

Examination Committee:

Chair: Luís Manuel Marques da Costa Caires
Full Professor, NOVA University Lisbon

Rapporteur: David José da Silva Aresta Belo
Senior Scientist, FRAUNHOFER Portugal

Co-adviser: Nuno Manuel Robalo Correia
Full Professor, NOVA University Lisbon

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
November, 2021

Health Outcome Pathway Prediction

To me and my future. To my family and my partner. To my friends and everyone I met along the way who helped me shape who I am today.

Acknowledgements

Naturally, I am grateful for the years spent in FCT-UNL, I would not be where I am, or who I am if I chose another path. It taught me much, but most important of all, it taught me how to learn, and how to be a real problem solver, an engineer.

I also extend my gratitude towards VOH.CoLAB and the FrailCare.AI project, where I had numerous opportunities to grow while developing my Msc.Thesis for the past year. To those I worked side by side, Pedro Dias, Salomé Azevedo, and my both work and college (and project) colleague Simão Gonçalves, thank you for being good and motivated in what you do and inspiring me to be better. Also, Dr. Ana Rita Londral, which also supported me throughout this journey, and more recently Dr. Federico Guede, who went above and beyond his responsabilities to help me improve this document. For VOH.CoLAB I can only offer praises and wish the best.

For the countless discussions that helped me grow, I thank all of my colleagues. I thank Guilherme Fonseca for challenging me; Paulo Mimoso, for that one project, and all the laughs and João Santos, for surviving through that OCaml project. All of you taught me a lot, and for that I am sincerely thankfull.

Finally, my deepest gratitude to my family, my mother and father that raised me to be what I want to be, and follow whatever dream I might have. You layed out the road for me, and for that I am forever grateful. I also thank my brother for pushing me to be better even unbeknownst to him, my grandmother, which probably still does not know exactly what I do and my partner Alizée, for supporting me greatly throughout this entire process. This work is supported by DSAIPA project FrailCare.AI (DSAIPA/0106/2019/02) and by NOVA LINCS (UIDB/04516/2020) with the financial support of FCT – Fundação para a Ciência e a Tecnologia, through national funds.

"We can only see a short distance ahead, but we can see plenty there that needs to be done. " (Alan Turing)

Abstract

This dissertation is part of the project FrailCare.AI, which aims to detect frailty in the elderly Portuguese population in order to optimize the SNS24 (telemonitoring) service, with the goal of suggesting health pathways to reduce the patients frailty. Frailty can be defined as the condition of being weak and delicate which normally increases with age and is the consequence of several health and non-health related factors.

A patient health journey is recorded in Eletronic Health Record (EHR), which are rich but sparse, noisy and multi-modal sources of truth. These can be used to train predictive models to predict future health states, where frailty is just one of them. In this work, due to lack of data access we pivoted our focus to phenotype prediction, that is, predicting diagnosis. What is more, we tackle the problem of data-insufficiency and class imbalance (e.g. rare diseases and other infrequent occurrences in the training data) by integrating standardized healthcare ontologies within graph neural networks. We study the broad task of phenotype prediction, multi-task scenarios and as well few-shot scenarios - which is when a class rarely occurs in the training set. Furthermore, during the development of this work we detect some reproducibility issues in related literature which we detail, and also open-source all of our implementations introduding a framework to aid the development of similar systems.

Keywords: SNS24, Frailty, Health Outcomes, Diagnosis Prediction, Few-shot Learning, Graph Neural Networks, Domain-Knowledge

Resumo

A presente dissertação insere-se no projecto FrailCare.AI, que visa detectar a fragilidade da população idosa portuguesa com o objectivo de optimizar o serviço de telemonitorização do Sistema Nacional de Saúde Português (SNS24), e também sugerir acções a tomar para reduzir a fragilidade dos doentes. A fragilidade é uma condição de risco composta por multiplos fatores.

Hoje em dia, grande parte da história clinica de cada utente é gravada digitalmente. Estes dados diversos e vastos podem ser usados treinar modelos preditivos cujo objectivo é prever futuros estados de saúde, sendo que fragilidade é só um deles.

Devido à falta de accesso a dados, alteramos a tarefa principal deste trabalho para previsão de diágnosticos, onde exploramos o problema de insuficiência de dados e desequilíbrio de classes (por exemplo, doenças raras e outras ocorrências pouco frequentes nos dados de treino), integrando ontologias de conceitos médicos por meio de redes neuronais de gráfos. Exploramos também outras tarefas e o impacto que elas têm entre si. Para além disso, durante o desenvolvimento desta dissertação identificamos questões a nivel de reproducibilidade da literatura estudada, onde detalhamos e implementamos os conceitos em falta. Com o objectivo de reproducibilidade em mente, nós libertamos o nosso código, introduzindo um biblioteca que permite desenvlver sistemas semelhantes ao nosso.

Palavras-chave: SNS24, Fragilidade, Resultados de saúde, Previsão de diagnosticos, Few-shot Learning, Redes Neuronais de Grafos, Conhecimento externo do domínio

Contents

List of Figures

List of Tables

Glossary

big data term commonly used to refer to the processing of large datasets, normally it is tied to the field of predictive analytics. i, 5

comorbidity presence of one or more additional health conditions often co-occurring with a primary health condition. i, 22

gigapixel a digital image bitmap composed of one billion (109) pixels (picture elements), 1000 times the information captured by a 1 megapixel digital camera. i, 1

inpatient a person who goes to a hospital for treatment and whose condition requires admission to a hospital i, 10

outpatient a patient who attends a hospital for treatment without staying there overnight. i

phenotypes phenotypess are clinical conditions or characteristics that represent a patient i, 22

Acronyms

AE Autoencoder i, 14, 22
AI Artificial Intelligence i, 1, 2, 3, 5, 25
ANN Artificial Neural Network i, 11, 12, 29

BERT Bidirectional Encoder Representations from Transformers i, 18, 25, 26
BI-RNN Bidirectional Recurrent Neural Network i, 16, 25

CCS Clinical Classification Software i, 31, 33, 34, 44, 48
CDSS Clinical Decision Support System i, 10, 11, 74
CNN Convolutional Neural Network i, 14, 15, 18, 24, 28

DL Deep Learning i, 1, 2, 5, 7, 9, 11, 13, 14, 17, 19, 21, 28, 31, 35, 37, 50, 60, 70, 72, 73, 75

EHR Eletronic Health Record i, vii, 1, 5, 6, 7, 9, 10, 11, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 34, 50, 52, 53, 75
ETL Extract-Transform-Load i

GAT Graph Attention Network i, 19, 35, 36
GCN Graph Convolutional Network i, 36
GCNN Graph Convolutional Neural Network i, 19, 20, 27
GMPNN Graph Message Passing Neural Network i, 19
GNDP Graph Neural Network-Based Diagnosis i, 27, 28
GNN Graph Neural Network i, 18, 19, 20, 21, 26, 27, 28, 29, 30, 73, 74
GRU Gated Recurrent Unit i, 16, 24

HOPP Health Outcome Pathway Prediction i, 7, 30, 31, 61, 75

ICD International Classification of Diseases i, 6, 10, 23, 31, 33, 34, 41, 42, 44, 46, 47, 50, 53

ICU Intensive Care Unit i, 6, 10, 40, 42

LSTM Long Short-term Memory i, 16, 24, 25

ML Machine Learning i, 1, 2, 50

MLP Multilayer Perceptron network i, 12, 13, 14, 22, 37

MR Medical Record i, 9

RNN Recurrent Neural Network i, 15, 16, 17, 18, 23, 24, 25, 26, 27, 28, 64

SOTA State-of-the-art i, 16, 18, 23, 29, 71

1 Introduction

1.1 Context

The accelerated growth in the development of healthcare information systems (specifically in data processing and warehousing capabilities) paved the way to more intricate systems which leverage Eletronic Health Record (EHR) for the betterment of patient care. Moreover, the increasing standardization of healthcare related processes has led to an increased interest in making use of the patient EHRs for secondary purposes besides their original intention - billing and maintaining a lenghty historic of *mostly unused* records. These EHRs represent a patient health journey throughout time through a diversity of sparse medical events - clinical notes, medications, vital signs, diagnoses, medications, etc. The heterogeneity and longitudinal aspect of this records make its analysis a challenging problem. Using these records to assist physicians or other clinical staff to augment patients care to prevent and predict health related problems is not an easy task. Even so, with the increasing research within Machine Learning (ML) and later Deep Learning (DL), several successful use-cases have been reported within this domain. For instance, Google's work on detection of diabetic retinopathy (Y. Liu et al., 2017), and detection of cancer metastases on gigapixel pathology images (Gulshan et al., 2016). Furthemore, in the larger ecossystem of digital health there startups and research centers which we also highlight as an indicator of the rising interest within this field, like UpHill Health[1], KNOK[2], abtrace[3] and VOH.CoLAB[4] just to name a few, which work diligently within the digital health and EHR analysis domain. Whereas, generally speaking, their goal is to implement the latest more relevant technologies for the betterment of society, overall improving patient care.

Most, if not all, of the latest successes and interest in EHR analysis is correlated to the rising use of ML techniques, in several contexts. Machine Learning (ML) is one sub-area of the widely known field of Artificial Intelligence (AI), which has been booming for some

[1] https://uphillhealth.com/
[2] https://www.knokcare.com/en/
[3] https://www.abtrace.co/
[4] https://vohcolab.org/

years. The term Artificial Intelligence (AI) was coined by John McCarthy in 1956, long before the advent of machine learning, and later, deep learning. However, nowadays, AI is mostly used to describe its sub-fields that encompass "learning" algorithms within ML and DL. Conceptually ML is when a machine can learn by itself how to perform a certain task without being explicitly programmed to do so. The premise can sound complicated, however, the most common and widely used ones are not particularly complex (e.g logistic regression or naive-bayes). The pursue of artificial general inteligence, i.e. machines that can do any task without being explictly programed or trained to, still appears to have a long path to go. Furthermore, the definition of inteligence itself and the research directions to achieve such inteligence are topics on its own. Nonetheless, the community appears to agree that DL is, as of now, the most promising way to achieve the goal of artifical general inteligence, where the learning process is solely based on the experience of others, without the need to hand-craft what the algorithms need. These brain-inspired algorithms, as history goes, receive raw data - be it numbers, images or sound - and learn on its own the relevant features that optimize the algorithm's task. With that said, the basis of DL are neural networks, which exist in an enormous variety and are used for a multitude of tasks in a wide variety of contexts - predicting diseases, suggesting treatments or other relevant courses of actions or even translate texts, summarization, question answering among many other possible and interesting use-cases. Nevertheless, even if there are several types of neural networks, they all share the same basic unit - the perceptron (an artificial neuron). The perceptron shares similar responsabilities with brain neurons, that is, to process and pass information, in the format of a mathematical function with a learnable parameter called weight. Hence, a neural network learns by adapting the learnable weights so that the output of the the neural network maps closer to the desired output.

Deep Learning (DL) albeit not a new concept, has been increasing in popularity and given a decent amount of attention in the past decade, even if its basis has been formulated more than 20 years ago. The increasing attention on the field led to a profound impact in a wide variety of applications: natural language processing, where a striking example is the capabilities of OpenAI's GPT-3 language model (Brown et al., 2020); reinforcement learning with DeepMind's MuZero (Schrittwieser et al., 2020, 7839), which can master several different tasks like playing chess and shogi without being explicitly told any rules of these games; the outpouring research within computer vision which is boosting the self-driving industry and facial recognition tasks; and more recently AlphaFold2 (Senior et al., 2020, 7792) also from DeepMind, who appears to have solved the protein folding problem. One could say that the possibilities of DL are endless and we are just starting to witness a rise of its applications throughout society.

1.2 Motivation and Problem

This work arised from FrailCare.AI[5], a FCT funded project with the goal of studying frailty in the portuguese elderly, through the use of AI techniques applied on patient data. With that said, these techniques will allow the creation of more detailed health pathways, predicting and preventing adverse health conditions and overall reducing the burden in the national health system, both economically and human resources wise. More specifically, the project aims to understand the root causes of frailty among the Portuguese elderly population, by flagging at national level citizens at risk, to then be further evaluated by specialized teams of clinicians. Furthermore, the developed analysis and AI models will be augmented with a visual tool that allows the clinicians to explore the patient health trajectory, find similar cases, with the goal of augmenting the clinicians ability to assist the flagged citizens towards a positive outcome. The current pilot employed within the context of *SNS24 senior*, a telemonitoring program that targets elderly people, consists of a pool of pre-selected citizens which after being flagged as frail are followed through phone calls throughout several weeks, where the clinicians monitor their frailty level using standardized questionnaires. We can define frailty as the condition of being weak and delicate, which is the consequence of several health factors normally correlated with old age(Xue, 2011). As of now in order to assess the level of frailty among the elderly, each patient answers a standardized questionnaire known as Tillburg (Gobbens, van Assen, Luijkx, Wijnen-Sponselee, & Schols, 2010), which results in the Tillburg Frailty Indicator - a value between 0 to 15, where more than 7 is considered as frail, and the closer to 15, the more frail. In fig. 1.1 we illustrate what the full system resulting from the FrailCare.AI project looks like.

FrailCare.AI is a project with several partners, such as: Comprehensive Health Research Centre (CHRC), Faculty of Sciences and Technology (NOVA-FCT), SPMS's and Value For Health CoLAB, where this work was developed. Value for Health CoLAB (VOH.CoLAB) is a non-profit organization founded in 2019 whose mission is to measure value in Health. More specifically, to accelerate the fundamental restructuring of Healthcare delivery towards a paradigm shift to Value-Based Healthcare and Patient Empowerment[7]. With that said, within this context, three more works were developed in parallel with FrailCare.AI, with different levels of contributions.

- TREC Precision Medicine 2021 (Cardoso & Martins, 2020), where we studied and developed information retrieval systems to improve search engines within the healthcare domain with the goal of creating a precision medicine system that retrieves relevant biomedical literature given a specific health state of a patient.

- TREC Clinical Trials 2021, a joint effort with Simão Gonçalves, also under the

[5] https://frailcareai.vohcolab.org/
[6] Icons taken from https://www.flaticon.com/
[7] Taken from VOH.CoLAB website

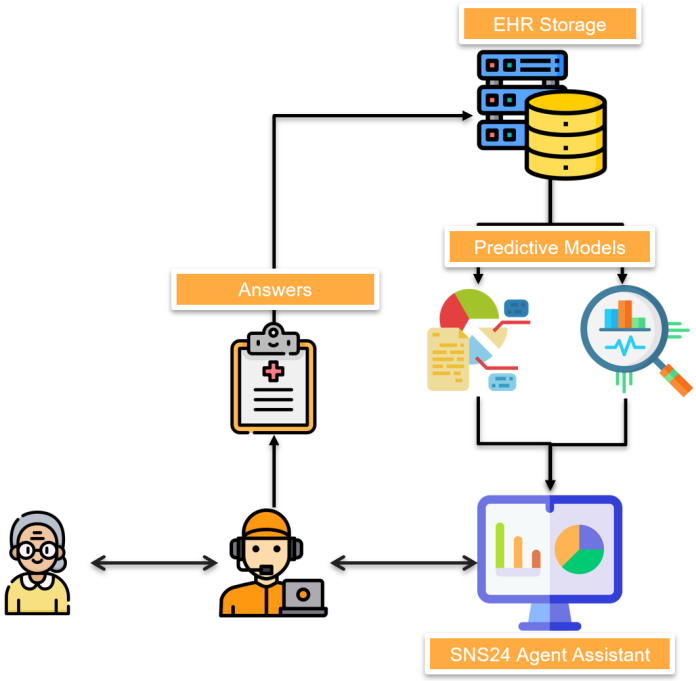

Figure 1.1: System overview of FrailCare.AI [6]

supervison of Flávio Martins, where we explored clinical trials matching techniques. From standard information retrieval systems to more elaborate graph-based aproaches.

- EasyHealth4Covid[8], a digital health platform developed for a nursing home with the goal of closely monitoring the elderly health status by measuring vital signs and pre-defined health-related questionnaires. The developed platform also offered a drug management tool, in a all-in-one platform optimized for elderly care.

- CoaguCheck Chatbot, a task-oriented rule-based chatbot used within the cardiothoracic service of a portuguese hospital to monitor hypocoagulation(INR) postoperation. At the moment of writting, the application is in a early stage of testing with only one user. The goal is to scalate the application to the entire service. In addition, an workshop paper is well underway.

Both FrailCare.AI and the remaining developed works showcase the shift and the rising interest and advantages in digitalizing healthcare systems. With that said, our focus in this work is the implementation and validation of the predictive models component of FrailCare.AI, whereas in (Rebelo, 2021), another work inserted within the context of this project, the focus is in developing an UI to assist the telemonitoring of the elderdy, and (Gonçalves, 2022), also another work within this context, to assess the impact of

[8] https://easyhealth4covid.vohcolab.org/

uncertainty in predictive models, and also how to leverage it to make a better use of said models.

That being said, FrailCare.AI lies within the overarching context of EHR analysis, a challenging problem to tackle, due to the data inherent complex and sparse nature. For example, Vuik, Mayer, and Darzi, 2016 explores how patient segmentation benefits healthcare, focusing the analysis on the patient and not on the provider. Similarly, Pathak, Kho, and Denny, 2013, Murdoch and Detsky, 2013 and Evans, 2016, Suppl 1, explore the possibilities of applying big data and AI solutions, discussing the recent advances, the current challenges and how EHRs paved the way for a more personalized, patient-centric healthcare. Namely, promote interoperability both in EHR and the systems architecture themselves.

Nowadays, is becoming ubiquitous using DL for EHR analysis mostly due to the necessary feature engineering effort that DL normally does not require. The more proeminent examples lie within computer vision field which enabled quick diagnoses of several diseases and ailments through images (Esteva et al., 2021). With that said, is is also known that most of the existing DL methods suffer from these two pitfalls: data insufficiency, and interpretability. What is more, in the specific case of EHRs tasks, both of these issues escalate further for two distinct reasons. First, the data insufficiency problem is latent within the data - even when the volume is deemed sufficient - due to the fact that there are several diagnosis, treatments and other medical concepts that are rarely diagnosed and given i.e rare diseases. The frequency that such medical events occur in the data is incredibly low, incrementing the difficulty of the learning process of the traditionally data hungry DL algorithms. Second, interpretability is an important challenge to overcome, healthcare tasks are applied to human beings, thus, a model being interpretable and transparent is a requirement, for both monitoring and trustworthiness.

In addition, we also identify another more structural issue in the field. The reproducilibity issue known in DL also haunts the researched context, which hinders progress due to the fact that it is difficult, or sometimes impossible, to build upon previous solution. Either, no code is given, or is far from production-ready levels, or the documents themselves are not clear on practical details. Several related literature studied for the implementantion of this works present these reproducibility issues.

FrailCare.AI, as most of the research within EHRs and Health Informatics, aims to achieve a scalable and interpretable precision medicine model. A way of thinking healthcare which defends that each treatment should be tailored specifically for each individual, and revolve around the patient. Thus, healthcare should not generalize too much since everyone is different. However, there is only so much that a human can do and manually pinpoint. Thus, the need of automated methods that discover and suggest possible outcomes is required to achieve a scalable, accurate precision medicine model.

In practice, a precision model system has several components, in fig. 1.2 we show a holistic view of what a complete system would look like.

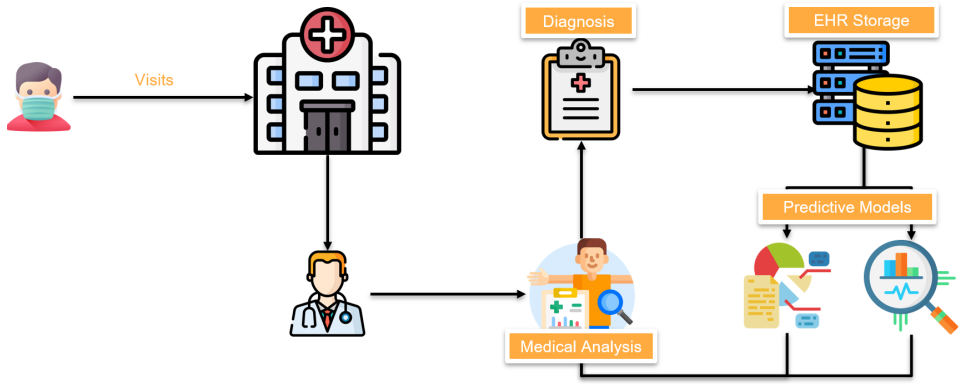

Figure 1.2: System Overview for one patient [9]

1.3 Objective

The objective of this work lies within the context of health outcomes prediction i.e predicting future health states of a patient, namely frailty. Which materializes in a precision medicine model that use both the patients EHRs and phone-call information, with the goal of assisting the clinicians in the care of the patients. Improving the care of the elderly by preventing and reacting sooner to any abrupt negative health change if needed. However, due to several constraints outside of our control, we could not have access to the EHRs pertaining FrailCare.AI, which are fundamental to predict health outcomes. As a solution, we pivoted the proposed research to use publicly available datasets like MIMIC III and eICU, which have been used throughout the literature for a multitude of tasks. This dataset shift also changed the domain from primary care to intensive care, since both MIMIC III and eICU are Intensive Care Unit (ICU) datasets - more condensed in time. What is more, these datasets follow the International Classification of Diseases (ICD) coding scheme - which organizes diseases by codes - hence the research done with his datasets can be generalized to others that follow this standard. With that being said, the tasks that interest us now are:

- Phenotype prediction (predicting diagnoses)
- Procedure suggestion (predict procedures)
- Mortality prediction, both in-hospital mortality and mortality at 30 and 90 days range

The aforementioned tasks can be used as a proxy to frailty, thus our research still lies within the context of FrailCare.AI.

With that said, in order to tackle the three issues identified in the section above, we research the use of Graph Neural Networks within the context of the proposed tasks.

[9] Icons taken from https://www.flaticon.com/

More spefically, we develop an ontology guided Graph Attention Network based architecture, that is applied on heterogeneous graphs with an arbitrary of data modalities per patient. We used (Choi, Bahadori, Song, Stewart, & Sun, 2017; Y. Li, Qian, Zhang, & Liu, 2020; Ma et al., 2018) as the foundations for this work, which showcase that its ontology component, augments the architecture promoting better results for infrequent outcomes. What is more, we decided upon an attention based architecture since these mechanisms are used throughout the literature to help intepret the results, unraveling the so-called black-blox. In addition, since we are interested in several tasks, we prepared the proposed architecture to be used to predict multiple outcomes simultaneously, inspired by the notion that learning multiple correlated tasks simultaneously should improve peformance, or at the very minimum promote faster convergence. That being said, multi-task learning is not a novel area of research, however there is no unified framework within the context of the proposed tasks that properly define its usage. Furthermore, it has shown promising results as demonstrated in Harutyunyan, Khachatrian, Kale, Ver Steeg, and Galstyan, 2019; L. Liu et al., 2020.

Furthermore, we also set out the goal of building as near as possible production-ready framework that uses the mentioned architecture and documenting it properly. Finally, with the focus on reproducilibity and to assist further research, we had the objective to perform an extensive ablation study within the context of the proposed framework, displaying several metrics and possible variations within the proposed architecture, validating the findings with the literature.

1.4 Approach and Document Structure

We approach this work by researching the relevant literature with the goal of building on top of it. However, due to several constraints (e.g., pivoting away from FrailCare.AI due to lack of data, reproducibility issues and hardware constraints) the effective approach diverged into focusing in validating our empirical findings with the literature, besides implementing the proposed framework. With that said, we structure the remaining of the document as follows:

- Chapter 2 introduces EHR with more detail and discusses the related work starting with DL concluding with its application within EHR analysis
- Chapter 3 introduces the Health Outcome Pathway Prediction (HOPP), describing the used architecture, all of its components and the respective experimental setup used to evaluate its performance.
- Chapter 4 we show and discuss the results of the proposed architecture.
- In Chapter 5 an ablation study is performed, where we study the impact of different configurations of the proposed architecture.

- Chapter 6 concludes this work, where it discusses the limitations and future work

2

Related Work

2.1 Introduction

In this Chapter, we analyze the different artificial intelligence methods applied on Eletronic Health Record (EHR) for a multitude of tasks, with the focus on deep learning methods. First, we define the concept of EHR giving a brief historical review, referencing the obstacles and advantages of digitizing medical records. Afterwards, we overview the relevant concepts within Deep Learning (DL) to then describe the architectures that were studied for this work. Then, we explore how prior work used DL within the context of healthcare related tasks, more specifically with EHR as its primary data source. We conclude this chapter with a critical summary correlating the relevant characteristics of each studied work that guided this research.

2.2 Electronic Health Records

Humans historically write down and store a multitude of documents. Within healthcare, is no different. In matter of fact, Medical Records (MRs) appear to have existed at least since Egyptian times, as stated by the survey done in Evans, 2016, Suppl 1. However, in the USA, it was not only until 1900-1920 that these records started to be more steadily used. Furthermore, back then, only paper records were available, thus these medical record data were mostly only stored for billing purposes and never seen again.

As technology evolved, hardware got cheaper, and new software was developed to handle the growing needs of society. Eventually, it was cheap to process and record medical data using computers, simplifying administrative, billing and other tasks within healthcare facilities. As a consequence, medical data was now easier to read, search and share, and later on with the advent of the world wide web, accessable from almost any location in the world. Thus, Eletronic Health Records, were born. Medical records stored electronically within both online and offline databases.

Nowadays, EHRs are abundant and widely used throughout the world, typically stored in relational databases, where they can be easily accessed, read and manipulated.

Albeit still not widely standardized there are several initiatives across the healthcare domain to facilitate the interchangeability and interoperability of the data, for instance, openEHR[1] and FHIR [2]. Nonetheless, even if the overall format is still not standardized across borders, and sometimes not even within a country, EHRs already make use of standardized unique identifiers for medical codes, like the ICD[3], currenly on its 10th revision, well structured ontologies and software like SNOMED[4] and CCS[5] that connect the mentioned identifiers between themselves and among standards.

With that said, EHRs are widely adopted to record longitunal patient health data, thus recording a patient health journey throughout time, and space (i.e. hospitals and clinics). EHRs are multi-modal and can span between data about primary care, day-to-day consultations and intensive care units, being it one time visits or prolonged stays in hospitals, diagnoses, laboratory tests and results, prescriptions, procedures, radiological images, clinical notes, demographic information and possibly much more. Due to the nature of such data, EHRs are proprietary and private, which in a way hinders research, since it can be hard to access it. Nonetheless, there are examples of anonymized datasets that are accessible and used within the academia, where we specifically highlight two: MIMIC III and eICU. MIMIC III (Johnson et al., 2016, 1), which stands for "Medical Information Mart for Intensive Care", is a widely used dataset within the academia which holds information from the ICUs in Beth Israel Deaconess Medical Center between 2001 and 2012. From the same provider[6], eICU (Pollard et al., 2018), is a larger and newer dataset when compared to MIMIC III , with slightly different characteristics. These two datasets require only the researcher to pass the CITI "Data or Specimens Only Research" course[7] and refer only to inpatient data, since both of the datasets are retrieved from ICUs. In addition to these two datasets, we also point out the CPRD[8] dataset, which collects anonymized patient data from a network of GP practices across the UK. This dataset has over 50M patients, where 16M are currently registered, being the largest by a wide margin of the three described datasets. However, it is behind a paywall, thus not easily accessible.

The wide use of EHRs paved the way to new interesting secondary uses: it can not only be used to understand what types of diseases occur frequently as well as inform researchers so they can understand why act upon them; it can be used to understand what type of resources a hospital should bring in so that more and more patients are dealt accordingly; it can be leveraged by insurance companies or other businesses that require insights from healthcare data or for Clinical Decision Support Systems (CDSSs) but also

[1] https://www.openehr.org/
[2] https://www.hl7.org/fhir/
[3] https://icd.codes/
[4] https://www.snomed.org/
[5] https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp
[6] https://physionet.org/
[7] https://www.citiprogram.org
[8] https://www.cprd.com/

for research and exploratory purposes. CDSSs benefit greatly from the rich variety and quantity of data that is present on EHRs. According to Kawamoto, Houlihan, Balas, and Lobach, 2005, around the year 2005 in the US, adults only received around half of the recommended care. Furthermore, it is estimated that up to 98,000 US residents die each year as the result of preventable medical errors. These statistics appear to suggest that CDSSs back then were still either under-developed or under-used. However, such systems are beneficial to all parties involved. It can reduce hospital costs and overall improve patient care. More specifically, these type of systems can aim to predict mortality or the chance of a patient being readmitted, which might indicate that that patient needs extra attention because something might be wrong; it can forecast both length-of-stay and medical costs which can be used to optimize hospital resources; disease or trajectory prediction which augments the physician diagnosis capabilities; and patient phenotyping, which is the process of figuring out which group a patient belongs according to its characteristics. All of the aforementioned tasks are being explored using data-driven methods, more specifically statistical models or artificial intelligence methods like support vector machines or deep learning neural networks based techniques. Implementing a successful CDSS is a complicated and complex endeavor that join several fields of expertise.

2.3 Deep Learning Overview

Deep Learning (DL) is a vast area that encompasses several techniques and architectures, with a huge amount of research pouring daily. Nevertheless, all of these architectures share the central idea of *representation*. Deep Learning (DL) aims to ease the tedious and complex work of feature engineering. Traditionally, before applying machine learning algorithms, the input features would have to be hand-crafted from raw data. This process was done by a human and based on his/her experience and domain-knowledge. Hence, it could be time-consuming, complex and in some sense, similar to a creative process. In contrast, DL techniques try to learn optimal representations from the data itself without any human guidance, which also has the side-effect of discovering latent data relationships that might otherwise be unknown or hidden. These complex DL techniques are no more that a sequence of simpler operations, in fact, the vast majority of DL algorithms and architectures are built upon the framework of Artificial Neural Networks (ANNs), which are a set of simples structures, commonly called perceptrons, as illustrated in fig. 2.1.

Its output is computed as a non-linear combination given the input X, as follows $\hat{y} = \sigma(Wx + b)$, where W are the set of weights, b the bias and σ a non-linear function.

With that said, the perceptron "learns" by computing a loss function $loss(y, \hat{y})$ between the predicted y and the real y of the training data, and then adjusting the weights and bias following the gradient descent algorithm, which can be generally described as follows:

$$\theta = \theta - \eta \nabla_\theta J(\theta) \tag{2.1}$$

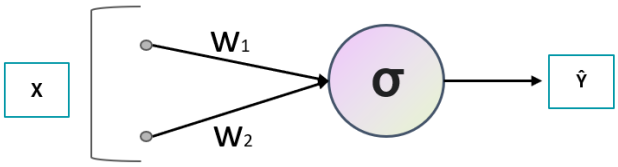

Figure 2.1: Perceptron

where θ are the parameters of the loss function, ∇_θ the gradient of the loss function $J(\theta)$ with respect to the parameters, and η the learning rate. The defined generic gradient descent formula can be used to optimize any loss function, as long as the gradient is defined. However, it is usual to use variations of said formula, like Stochastic Gradient Descent (SGD), which performs a parameter update for each training example, or Mini-Batch Gradient Descent (MBGD), which performs a parameter update for a subset of the training examples. Nevertheless, in practice, the aforementioned algorithms are normally augmented with more sophisticated algorithms like Momentum, AdaGrad, or Adam.

With that said, an ANN is a sequence of interconnected layers of nodes (perceptrons), whereas the layers that are between the input and output layer are commonly referred as hidden units. Thus, an ANNs "learn" by updating the set of weights of each perceptron.In this section, we will briefly describe the relevant architectures that inspired this work. However, for more complete and detailed explanations of the following architectures please refer to Goodfellow, Bengio, and Courville, 2016.

2.3.1 Multilayer perceptron

Multilayer Perceptron network (MLP) or Full Feedforward neural network is a type of ANN where each neuron in a layer i is fully connected to every node in the layer $i+1$ as illustrated in Figure 2.2.

In this type of architecture, each node of each hidden layer computes a non-linear combination of the previous layer by a weighted sum of the outputs of that layer followed by a non-linear activation (section 2.3.1.1) function with the computed value. This procedure is shown in Equation (2.2).

$$h_i = \sigma(\sum_{j=1}^{d} x_j w_{ij} + b_{ij}) \tag{2.2}$$

Here, d is the number of nodes in the previous layer, x_j the output from the previous layer's j^{th}, w_{ij} and b_{ij} the learnable weights and bias terms associated with each x_j, respectively, and σ the applied activation function,

[9] Taken from https://github.com/dair-ai/ml-visuals

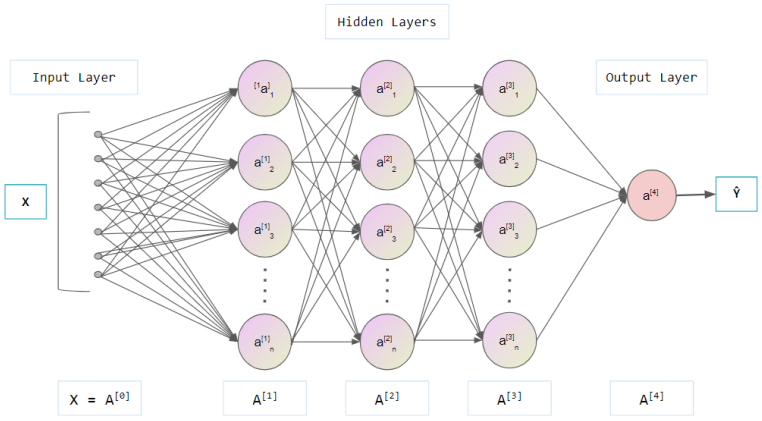

Figure 2.2: Multilayer Perceptron Neural Network[9]

An MLP learns by optimized the weights during training towards minimizing the defined loss function. The goal is to create a correlation between the training input data x and the training output y, with the goal of generalizing to unseen data within that distribution. The MLP architecture is one of the simplest models, and it is often used in other more complex and intricate models to achieve the final output y as the go-to function approximator in several cases.

2.3.1.1 Activation Functions

Activation Functions are used within DL architecture to help learn complex patterns of data since they are non-linear functions. As mentioned previously, each node applied a certain activation function to compute a non-linear combination of its inputs. Here we showcase the most common activation functions, which are:

$$ReLU(x) = max(0, x) \tag{2.3}$$

$$LeakyReLU(x) = max(\alpha.x, x) \tag{2.4}$$

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

$$Tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.6}$$

ReLU ignores inputs if they are negative thus not propagating its information.LeakyReLU allows just a small subset defined by α to passthrough, it speeds ups training. Sigmoid returns values between 0 and 1, which is often interpreted as a probability. Tahn always returns values between -1 and 1, making it zero centered which eases optimization.

2.3.2 Autoencoders

Autoencoders (AEs), or encoder-decoder architectures, are the traditional architectures within DL for unsupervised learning, which is a type of learning where no labels are given a priori. First introduced around 50 years ago, AEs aim to learn representations of the data by reconstructing it. These type of networks are designed to encode the input data into a (*normally*) lower dimensional space z, to then decode z and reconstruct it to match the original input. AEs in their most simplified form are basically a MLP network with only one hidden layer where the input layer and output layer have the same number of nodes. Nevertheless, they can also be stacked and have an arbitrary number of hidden layers or be forced to introduce noise in the data to further improve the representations learned. The basic architecture of an AE is illustrated in Figure 2.3.
As the network is trained, an AE learns a representation within the hidden layer that is able to reconstruct the initial data x from a lower dimensional space representation z into $\bar{x}$. Thus, the network objective is to minimize the reconstruction error $\|x - \bar{x}\|$. Once the network is trained, inputs can be fed into the AE and then retrieve the innermost hidden layer representations. In this manner, AE can be seen as a dimensionality reduction architecture that retrieves combinations of the most important dimensions of the data.

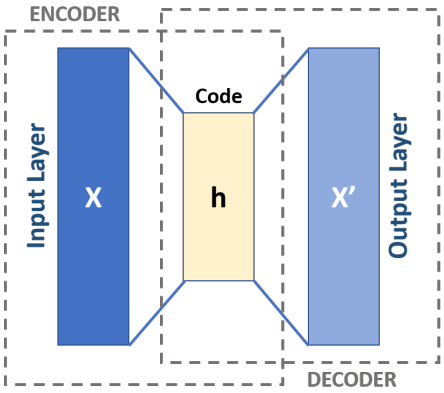

Figure 2.3: Schema of a basic Autoencoder[10]

2.3.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) were popularized by their successes within the image processing community. A CNN works by treating its input data as a set of local patches, through convolution operators, instead of uncorrelated individual data points. For example, instead of treating a 100x100 image as a 100,000 set of uncorrelated pixels, a CNN applies convolution operators across the image creating these local patches in order to try and retrieve more meaningful features of the image as illustrated in Figure 2.4.

[10]Taken from https://en.wikipedia.org/wiki/Autoencoder

Thus, aggregating the input data according to their neighborhood reducing the data dimensionality.

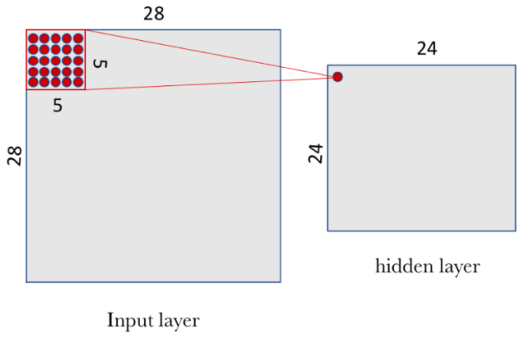

Figure 2.4: Convolution example [11]

Similarly, CNNs can also be applied to 1D data, or any other N-dimensional data, depending on the convolution operator used. This operator can range from a simple weighted sum of a point and its neighbors to other arbitrary complex functions, for instance it can be used to create sliding window of values across a 1D dataset that describe values throughout time, skip values at a certain timestamp and so forth. In Equation (2.7) we showcase the generic formulation of a 1D convolution, where x is the input signal and w the convolutional filter.

$$C_{1d} = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.7}$$

A key feature of CNNs is that the filters, the convolutions, are typically smaller than the input, which reduces the dataset size as noted, yielding in less parameters to be learned.

2.3.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are the traditional choice to handle sequential data of arbitrary length, (e.g time-series and natural language). Recurrent Neural Networks (RNNs) tend to outperform CNNs in the aforementioned tasks because the resulting extracted features from a convolution are sup-optimal and shallow. Mainly due to the fact that the these convolutions can only retrieve information within a fixed number of neighbors, thus, only a few selected neighborhood points will be used at each timestamp, ignoring distant dependencies. Whereas RNNs are designed to comprehend and make the most of long-range temporal dependencies. Recurrent Neural Networks, simply put, are Neural Networks where their output is also the input at the next time-step, thus the

[11] Taken from https://torres.ai/deeplearning/

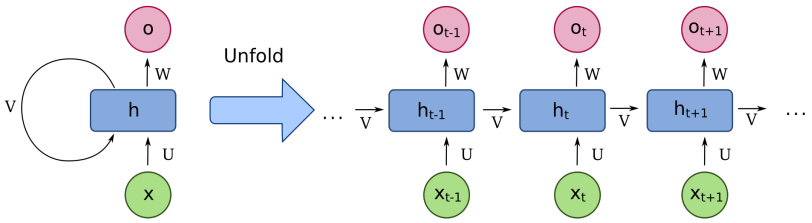

Figure 2.5: RNN illustration [12]

recurrent on its name. RNNs work by updating a hidden state h_t based on input x at time t but also on the previous hidden state h_{t-1}, which in turn was updated from x_{t-1} and h_{t-2}, and so forth as seen Equation (2.8) and illustrated in Figure 2.5. With this mechanism, after processing an entire sequence, the final hidden state contains the information from all its previous elements.

$$h_t = \sigma(h_{t-1}, x_t) \tag{2.8}$$

RNNs have several popular variants, namely the Long Short-term Memory (LSTM) and Gated Recurrent Unit (GRU). These variants differ the most from the base RNNs by adding mechanisms that control the flow of information within an hidden state, preventing the problems of vanishing or exploding gradient that might occur in long sequences. LSTM and GRU contain mechanisms to forget, reset or allow the information from the previous hidden states to passthrough.

RNNs can be applied in either direction of the input data, from the start to end, or end to start, yielding different outcomes. For instance, to predict stock prices, it is expected to start from time 0, whilst, to predict diseases it has been discussed that akin to how doctors operate, it yields better results if the model starts from the most recent event. Nonetheless, RNNs can also be applied both ways, in the format of Bidirectional Recurrent Neural Networks (BI-RNNs) (Schuster & Paliwal, 1997), which have been used to minimize the problems of long sequences, improving the predictive performance in several cases.

2.3.5 Transformers

Transformers are a novel architecture introduced in 2017 which are known to out-perform, in most tasks, State-of-the-art (SOTA) RNNs in sequence modeling and transduction tasks like machine translation and other natural language processing tasks. Introduced in the paper titled *Attention Is All You Need*, Vaswani et al., 2017, this architecture makes use of attention mechanisms and encoder-decoder architectures, dropping completely the need

[12] Taken from https://en.wikipedia.org/wiki/Recurrent_neural_network

of RNN components, in counterpart with other models that employ sequential components and attention mechanisms symbiotically.

Transformers, use encoder-decoder architectures to encode the input sequence to a sequence of continuous representations. These representations are then decoded to generate an output sequence of symbols one element at at time as illustrated Figure 2.6. Similarly to RNNs, this encoder-decoder is auto-regressive, which means that it consumes the previously generated symbols as an additional input at each time step. In addition, Transformers to make sure that a token only attends to the previous ones, a positional encoding is added to the respective input. However, this positional encoding is not a mandatory part of the architecture.

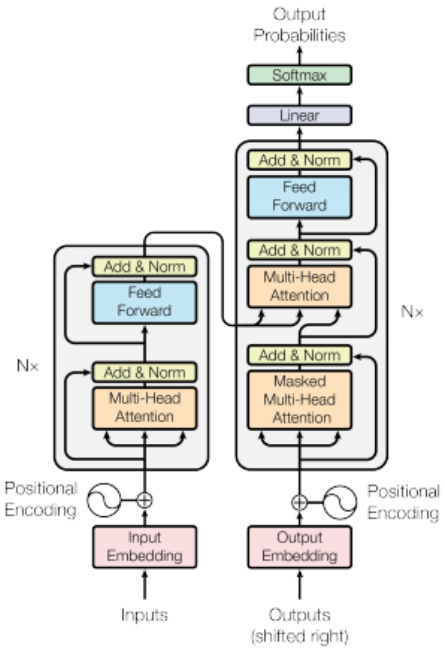

Figure 2.6: The Transformer - model architecture [13]

The novelty of the *Transformer* architecture is claiming that *Attention Is All You Need* (even if they also use encoder-decoders). Interestly enough attention is not a novel concept, conceptually, it simply states a weighted relationship between several tokens where all of the weights sum to 1. In a nutshell, attention within the field of DL can be broadly interpreted as a vector of importance weights between a given token and a vocabulary (set of related tokens).

In the case of the original Transformers architecture, the authors propose a scaled-dot product learnable self-attention mechanism that for a given token learns how to attend to all of the other tokens in the input data.

Attention mechanisms can be divided into three major categories: self-attention,

[13] Taken from Vaswani et al., 2017

which relates different positions of the same input sequence in order to compute a representation of the same sequence which has shown to be useful for machine reading tasks, *Transformers* make use of this mechanism; global attention, which attends to the entire input space and is tendencially used for computer vision tasks; finally, local attention, which attends only to a part of the input space, for instance a part of an image. There are several ways to mathematically compute attention, however, the premise is generally the same. Attention is a function with learnable parameters (e.g learnable weights W), which is computed for each token and then fed to a softmax function. A softmax function divides each value by the sum of all values so that the sum of all attention weighs for a given token sum up to 1, which makes it possible to intepret the resulting weighs as probabilities (e.g "Thinking" attends to "Machine" with the value of 0.12, thus "Thinking" concept relates with "Machine"with a 12% of probability).

This model architecture albeit recent is already widely used. In fact, Google uses a well-known Transformer-based architecture in its search engine [14], Bidirectional Encoder Representations from Transformers (BERT), introduced in Devlin, Chang, Lee, and Toutanova, 2019. Furthermore, according to the original paper, due to the non-existing RNNs components, the transformers architecture allows for significantly more parallelization and it reached a new SOTA in translation quality after being trained for as little as twelve hours on eight P100 GPUs. This claims were proved in the coming years with the multitude of transformer based architecture successes.

2.3.6 Graph Neural Networks

Graph Neural Networks (GNNs), first introduced in Gori, Monfardini, and Scarselli, 2005 and Scarselli, Gori, Ah Chung Tsoi, Hagenbuchner, and Monfardini, 2009, are a generalized version of CNNs, that operate on the graph-domain. CNNs are applied to grid-like data, which in turn can be seen as an instance of an unweighted undirected graph where each node is connected to its neighbors as illustrated in Figure 2.7.

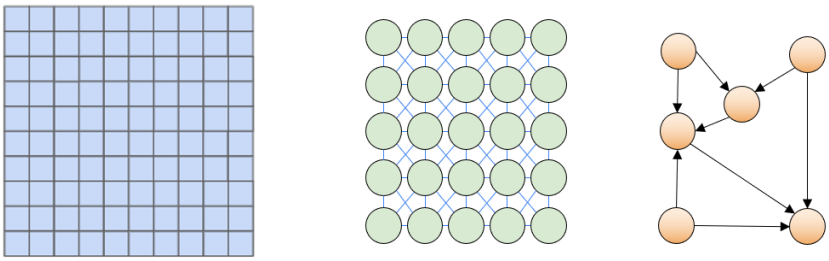

Figure 2.7: Image pixel-format (left). Image graph-format, where each node is a pixel (middle). Non-euclidean directed graph (right)

Graphs are a non-euclidean data structure that models set of objects (nodes) and their

[14] https://blog.google/products/search/search-language-understanding-bert/

relationships (edges). Hence, it explicitly uses the inherent structure of the domain to model the data, thus, allowing to express more complex, rich and intricate relationships. One of the traditional examples of a graph is a social network, where users are connected to other users through friendships, users are connected to posts and pages through likes. Formally, a graph is defined as $G = (V, E)$ where $V = \{1, ..., n\}$ are the nodes and $E \subseteq VxV$ the edges, as illustrated in the middle and right image in Figure 2.7.

GNNs belongs to the wider field of Geometric Deep Learning (Wenming, Yan, He, & He, 2020), which encompasses all DL architectures that operate on non-euclidean data, .i.e graphs and manifolds, for instance 3D shapes. Albeit not novel, since GNNs were first introduced in 2005, its usage and popularity have been increasing. A good example is that as of now, tech giants like Uber [15], and Pinterest [16] are using these type of algorithms to create richer recommender systems.

A Graph Neural Network works by aggregating the information in each node, propagating it throughout the graph. Its primary objective is that each node learns how to aggregate information of itself and its neighbors, yielding in a more complete and sound representation of the data, by leveraging the nodes feature and the graph structure itself. GNNs nowadays can be broadly divided in three categories, Graph Convolutional Neural Networks (GCNNs), Graph Attention Networks (GATs) and Graph Message Passing Neural Networks (GMPNNs) which we do not discuss in this work. However, there are several graph-based models that leverage random walks which do not fit exactly in any of this categories. The key difference between the aforementioned families is how they aggregate the information. Ultimately, the use of GNNs is to create rich representations in each node, thus these type of architectures can be used to aid supervised learning problems or be applied to fully unsupervised tasks, being it in a transductive setting or inductive setting. In this work, we are more interested in the inductive setting since it allows the model to generalize and work with unseen data, whilst transductive models only work with the data at hand in a semi-supervised learning context.

For this work, we highlight GAT, introduced in Veličković et al., 2018, which resides in the GAT category, *DeepWalk* (Perozzi, Al-Rfou, & Skiena, 2014), *Node2Vec* (Grover & Leskovec, 2016), *Metapath2Vec* (Dong, Chawla, & Swami, 2017), and *GraphSage*, (Hamilton, Ying, & Leskovec, 2017), in the random walk category and finally GCNN, as discussed in Kipf and Welling, 2016a from the GCNN category. For more details on GNNs and on it variants please refer to Zhou et al., 2020 and Wenming et al., 2020.

GAT makes use of self-attention mechanisms to learn the importance of edges of a graph given its nodes features. In this type of network the graph structure might be unknown, thus it can learn new edges akin to a Transformer-based architecture [17].

Within the random walk category, we highlight *DeepWalk*, which popularized the

[15] https://eng.uber.com/uber-eats-graph-learning/

[16] https://medium.com/pinterest-engineering/pinsage-a-new-graph-convolutional-neural-network-for-web-scale-recommender-systems-88795a107f48

[17] https://thegradient.pub/transformers-are-graph-neural-networks/

term, showed relevant improvements by leveraging short random walks to learn social representations. Random Walk techniques are highly inspired by language modeling techniques, where first a random walk is performed throughout the graph to create the equivalent to a sentence, so that then algorithms like *SkipGram* can be applied. Afterwards, *Node2Vec* improved upon *DeepWalk* by allowing bias in the random walks, which help guide the walks towards more representative paths, visiting a more diverse number of nodes. *GraphSAGE*, which is an acronym for Graph Sample and Aggregate, based off and improved upon the aforementioned previous works. *GraphSAGE* stands as a general framework for inductive node embedding and is a general framework for inductive node embedding that leverages node features to learn latent representation functions that can generalize to unseen nodes. In contrast to previous works, *GraphSAGE* learns aggregator functions instead of node embeddings vectors, which learn to aggregate feature information from a node's local neighborhood embeddings. Finally, *MetaPath2Vec* aims explicitly to create sound node representations on heterogeneous graphs (graphs with more than one type of node) by leveraging metapaths which helps guide the random walks similarly to *Node2Vec*. We note that working with heterogeneous graphs is a challenging task, both in theory and in practice, albeit most of the data in the real world is heterogeneous.

In Kipf and Welling, 2016a, the authors propose a semi-supervised GCNN, where the model's goal is to classify unlabeled points on partially labeled datasets (transductive learning). Conceptually, the authors follow the assumption that closely related nodes share labels and make use of an approximate yet efficient Graph Convolution to aggregate information.

The aforementioned architectures are only applied on what is considered *static* data, whereas EHRs are sequential and timestamped. With those requirements in mind, we also highlight Rossi et al., 2020 and Yan, Xiong, and Lin, 2018, which expands the capabilities of GNNs to timestamped dynamic data. The former approach makes use of memory modules and graph-based operators to constantly update a node state based on its past and its neighbors, in some sense, similar to a RNN. Whereas, the latter, generalizes GCNN to the Spatial-Temporal domain, which applies graph convolutions throughout timestamped graph snapshots, i.e multiple graphs. Additionally, the introduced GNNs until now are only meant to tackle homogeneous graphs, i.e graphs with only one node type and relationship type. However, graphs in the real world are more often heterogeneous than homogeneous, take the example of the social network graph introduced earlier. Relationships between users are different from relationships between a user and a post, and a homogenous GNN does not take that into account. With that in mind some recent works (Hu, Dong, Wang, & Sun, 2020; X. Wang et al., 2019), delved into heterogeneous, by design, GNNs. Heterogeneous Graph Transformer (HGT) (Hu et al., 2020), introduces a plethora of relevant concepts for the studied context; temporal encoding for heterogeneous graphs, heterogeneous graph sampling and a novel heterogeneous graph neural network. Heterogeneous Graph Attention Network (HGAT) (X. Wang et al., 2019) uses hierarchical attention, where first learns the relationships of meta-path neighbors of a

node (node level) and then learns how to aggregate them (semantic level). Whereas, HGT, also an attention based architecture, it differs from HGAT on the aggretation step. HGT breaks down the graph in meta relationships, parameterizing the weight matrices individually for each of these relationships, whereas the weights in HGAT are shared between meta-paths. This parameterization removes the need to manually curate meta-paths, and allows nodes to interact with each other no matter their type.

With that said, GNNs being a emerging subject among both the academia and industry, several tools and frameworks have been being developed, either to assist in the development, evaluation or deployment of said architectures. We highlight PyTorch Geometric (Pyg), which has just recently released v2.0, and the framework of choice for this work, Pytorch Geometric Temporal[18] an extension of Pyg to handle temporal and dynamig graphs and we also point to DGL[19] which appears to be less known.

2.4 Deep Learning on Electronic Health Records

Eletronic Health Records are a rich multi-modal, heterogeneous and by nature sparse source of information. Thus, manually engineering features is not feasible, hence, DL models are the go-to algorithms to fully use all of the present information.

In this section, we will discuss EHRs tasks that we deemed relevant for this work. We make use of the definitions above in section 2.3 and delve deeper into the mechanisms of the architectures used, describing their inner workings, where their novelty lies and where they need improvement. It is important to note that a big challenge on this field is the lack of generalized standard datasets and reproducible methods, thus, it is complicated to directly compare some of previous works. For example, several models use EHRs clinical notes and pre-process it to retrieve the relevant medical concepts, however, other models make use of datasets where the medical codes are already identified and semi-structured with no need of the pre-processing layer. Furthemore, even within the same task, the results differ greatly due to different processing pipelines.

The group of tasks that are relevant to this work are mostly sub-tasks of the more general task of health outcomes prediction. Here we are interested in predicting outcomes to be able to compute a patient trajectory, and assess risk. Nevertheless, we also discuss unsupervised tasks with the goal of creating sound representations for analysis and phenotyping or to be used for another models.

2.4.1 Representation Learning

Representation Learning is the task of learning representations of the raw input data, avoiding manual feature engineering. Deep Learning models by nature always perform representation learning, however, some models aim specifically at only retrieving rich

[18] https://github.com/benedekrozemberczki/pytorch_geometric_temporal
[19] https://www.dgl.ai/

low-dimensional vector representations of the data. Within EHRs, this representations can be leveraged to represent patients, discover phenotypes, comorbidities and other unseen relationships.

2.4.1.1 Patient Representation

Within the domain of representing patients in low-dimensional vectors, we highlight *Deep Patient* introduced in Miotto, Li, Kidd, and Dudley, 2016, 1. *Deep Patient* is built upon stacked denoising Autoencoders, with three layers that are independently trained, that uses raw EHR data as its input. This AE architecture inserts noise in the input data, thus the denoising part in its name. The model was trained with 704,857 patients and evaluated with a future diseases prediction task using random forest classifiers trained in every disease (one-vs-all learning). *Deep Patient* showcased that deep representations beat the traditional methods of representation learning like PCA. Convolutional Autoencoder (ConvAE), introduced in Landi et al., 2020, 1 with the same objective as *Deep Patient*, introduces a convolutional layer before the AE that convolutes through the time-axis. This layer allows the model to understand the evolution of a patient throughout time, whilst *Deep Patient* does nothing of the sort and leave it for future work. What is more, ConvAE is not applied directly on the raw EHR data like *Deep Patient*, instead, the authors first apply word2vec and then feed the resulting embeddings into the architecture.

2.4.1.2 Concept Representation

Patients are represented by the set of diseases, treatments, lab results, and other medical occurrence in their trajectory. Thus, developing ways to grasp representations of these building blocks might improve the representation of the patient as a whole. We differentiate two approaches when it comes to concept representations: learn by co-occurrence or learn by external domain-knowledge.

Med2Vec, introduced in Choi, Bahadori, Searles, Coffey, and Sun, 2016 and *EHR2Vec*, introduced in L. Wang et al., 2020 are similar architectures in the sense that they model the problem of learning representations and relations between concepts as a natural language processing task. *Med2Vec* is a MLP architecture inspired by *SkipGram* that is trained by learning medical codes co-occurence, whilst simultaneously given a visit *t* it predicts the codes in *t+1* and *t-1*. Whereas *EHR2Vec* is based on multi-headed self-attention and aims to capture the relations between medical codes within each of the patient's medical events (visits), akin to Transformer based architectures. The use of self-attention yields in more (internal) interpretable representations and leverages temporal relations better, since it can understand longer relationships. Furthermore, after the attention layer, *EHR2Vec* applies a MLP to capture co-occurrence between visits. *EHR2Vec* proved experimentally that the representations learned are more accurate than the ones produced by *Med2Vec*. With that said, the aforementioned, and in some sense, traditional

methods make use of word embeddings or similar algorithms to understand the relationships between the concepts as they occur, disregarding prior knowledge of the domain. This approach suffers greatly in few-shot scenarios where data is insufficient for some classes i.e., some words rarely or never appear in the training set.

In Choi, Bahadori, Song, et al., 2017, the authors leverage an ICD-9 hierarchy defined by CCS [20] to create richer representations of the concepts whilst tackling the problem of data insufficiency applied on a predictive task. With the same goal in mind, in Agarwal et al., 2019, the authors employ graph based representation methods on SNOMED[21] to encode the medical codes into richer more general representations that retain structural information of the concepts. The authors experiment with *node2vec*, *metapath2vec*, and *Pointcaré embeddings* (Nickel & Kiela, 2017), and empirically show that the proposed methods outperform previous SOTA co-occurrence algorithms on multiple tasks. *SNOMED2Vec* differentiates itself from Seq2Vec or Transformer like approaches as Choi, Bahadori, Searles, et al., 2016, L. Wang et al., 2020 and Y. Li, Rao, et al., 2020, 1 since it is applied directly on the already existing domain knowledge whereas the later approaches try to learn codes representation through co-occurrence within EHRs data.

2.4.2 Outcome Prediction

Outcome prediction is a superset of tasks with a high degree of revelance within biomedical informatics. Outcome prediction is any task that predicts and/or suggest outcomes for a patient, being it a diagnosis, treatments or procedures, mortality risk, unplanned re-admission, length-of-stay, frailty or other risk factors. Thus, predicting outcomes in a timely fashion with enough degree of accuracy and confidence helps improving quality of care by preventing later complications.

Within the context of outcome prediction tasks we can specify further static outcome prediction and dynamic outcome prediction, where the former disregards time, e.g aggregating all visits as one, and the later does not. Furthermore, in both of the aforementioned categories, some architectures can be investigated and interpreted, whereas others, typically RNN models, do good at the task itself but the learned information within the model can not be reason with.

With that said, for sake of simplicity, here we focus mostly on the tasks of **diagnoses prediction**, a set of tasks with the goal of predicting a single or set of diagnoses, hence it can vary from a binary task, to a multi-class task to a multi-label task; **mortality prediction** and **unplanned re-admission prediction**, binary tasks with the goal of predicting mortality risk and unplanned re-admission, respectively, within specific timespans.

Furthermore, as it has been discussed in previous works, in practice it is sufficient to predict the category of a disease (Choi, Bahadori, Song, et al., 2017; Y. Li, Qian, et al.,

[20] https://www.hcup-us.ahrq.gov/toolssoftware/ccs/AppendixCMultiDX.txt
[21] https://www.snomed.org/

2020; Ma et al., 2018). This loss of granularity in the prediction allows the model to generalize better since there are less possible labels.

2.4.2.1 Convolutional Neural Networks

CNNs, albeit not the go-to architecture for sequential data, where EHR fit in, it has also been explored for a variety of outcome prediction tasks.

In Cheng, Wang, Zhang, and Hu, 2016, the authors apply convolutions through the time axis on one-hot encoded representations of the medical codes present on EHRs data. Furthermore, the authors try different convolutions and different aggregations, which they called temporal fusion frameworks, with the goal to understand in which part of the architecture the visits should be aggregated. However, this model aims only it aims to predict the occurrence of a given single disease, thus it was not studied how it would fare against multiple diagnosis prediction task.

Furthermore, we also highlight *DeepR*, short for *Deep Record*, introduced in Landi et al., 2020, 1, which attempts at detecting regular clinical motifs from irregular episodic records through Convolutional Neural Networks (CNNs). *DeepR* tackles the problem as a natural language processing task, where it looks for local patterns (*motifs*) through 1D sliding window convolutions in learned representations (embeddings) of sequences of EHRs data. Nevertheless, the base concept is similar to the architecture presented previously. The authors in *DeepR* use the learned representations to predict unplanned re-admissions in 6 month time, showcasing good results.

2.4.2.2 Recurrent Neural Network

Recurrent Neural Networks (RNNs) are traditionally the go-to architecture when it comes to sequential data, which is the case of EHRs data. One of the earliest (as far as we know) works within the studied context is *Doctor AI*, introduced in Choi, Bahadori, Schuetz, Stewart, and Sun, 2016, with the objective of predicting diagnoses and treatments of the next visit using the information of diagnoses and treatments of previous visits. In this architecture, the authors used the GRU variant due to its simplicity and equivalent performance to the more traditional LSTM variant. The authors evaluate the model with top-k recall, which means that they evaluate how much of the top-k results of their model match the real predictions. This metric resembles on how a doctor thinks, making it very interesting to evaluate this kind of models/tasks. *Doctor AI* was a state-of-the-art model at the time, however, it has been improved upon, some examples are Choi, Bahadori, Song, et al., 2017 and Choi, Bahadori, Kulas, et al., 2017, where each of them tackled a different problem of the model, data insufficiency and interpretability respectively.

DeepCare, is a custom made RNN for the specific task of trajectory prediction. Introduced in Pham, Tran, Phung, and Venkatesh, 2017, *DeepCare* is an end-to-end deep dynamic memory neural network, built upon LSTM. The authors to avoid padding and

handle arbitrary-size sequences, learn continuous representations of discrete EHR, to then use it as input to the LSTM. However, an interesting aspect of *DeepCare* lies within the used variant of the forget gate. Here, the authors, extend the forget gate to be a function of irregular time gap between consecutive time steps, and also introduce two forgetting mechanisms, where a disease over time tends to matter less and some diseases can replace others. With this extensions, the model is better prepared to handle time irregularities between data entries, as well as complex interactions throughout time. In addition, *DeepCare* as noted was modeled to completely fit the problem of trajectory prediction, where interventions influence diseases (and thus diagnostics). Hence, interventions are explicitly part of the output gate and forget gate of the LSTM unit.

RetainVis, (Kwon et al., 2019), introduced *RetainEX* an improved version of *Retain* which stands for Reverse Time Attention Model. *Retain* looks to a sequence of admission backwards and computes attention vectors for both visit-level and code-level by applying RNNs from the latest input to the oldest one. The use of attention mechanisms allows the model to be more interpretable, by exploring the weights of the respective vectors.*Retain*, however, is a single outcome prediction model, being validated with the task of heart failure prediction. *RetainEX*, built upon the previous stands for Reverse Time Attention Model with extra time dimensions and embedding matrices, where the authors altered RNN components to BI-RNNs, thus interpreting the data in both directions. Another contribution of this paper, however, not within the scope of AI is a visual analytics tool that aim to improve interactivity and usability of the model itself for the end-user, allowing to perform what-if scenarios and exploratory analyses. In a similar fashion,*Dipole*, introduced in Ma et al., 2017, also takes *Retain* as a source of inspiration. However *Dipole*, has the objective to predict the $t+1$-th visit's medical codes. It employs BI-RNNs similarly to *RetainEX* to compute its attention weights, however, they differ on how the attention itself is calculated, experimenting with three different methods that yield different results depending on the data which the model was trained. Both *Dipole* and *RetainEX* according to its respective authors achieves betters results than *Retain*, however, as far as we are concerned *RetainEX* and *Dipole* are yet to be directly compared performance-wise. It is also relevant to note that *Dipole* and *Retain* tasks are slightly different, thus, *Retain* was adapted to perform the same task than *Dipole*.

2.4.2.3 Transformer

As far as we are concerned, the only full transformer based architecture applied within the health outcomes domain is *BEHRT*, introduced in Y. Li, Rao, et al., 2020, 1. As the name suggest, it is an adapted version of BERT, where its goal is to predict the next diseases codes from a sequence of diseases codes. *BEHRT* authors adapted the problem of diagnosis prediction to a natural language task in order to fit BERT architecture. Here, the authors transform a patient journey into a document where the sentences represent the visits of a patient, which hold the set of diagnoses in that visit. Thus, the next token

matches the top-1 next most probable disease code, being able to generalize to the top-k most probable disease codes, based on the training data. Furthermore, the age of the patient and the number of the visit that each code belongs is also concatenated to each code so that the model uses that information as well. With that said, *BEHRT* is a multi-label classification model which permits to simultaneously predict a probability for each and every disease (as long as it is present in the training set), avoiding then the necessity to train one predictive model per disease. However, it suffers from data insufficiency when it comes to predicting rare diseases, and makes no use of lab results or proposed treatments, which have rich information about the patient journey. However, when it comes to the interpretability requirement, a BERT based architecture is highly interpretable and there are several out-of-the-shelf tools [22] that allow the developers to delve deeper into the inner workings of the models.

Within this realm, we also highlight van Aken et al., 2021 which uses a BERT based model on clinical notes for a plethora of tasks, augmenting the data with other sources of truth like Pubmed.

2.4.2.4 Graph Neural Network

Graph Neural Networks (GNNs), or more generally graph based models have been recently applied with EHRs for outcome predictions tasks with success. We highlight two key reasons for this success of: first, it helps to tackle the problem of data insufficiency by allowing the easy integration of domain knowledge through ontologies, which are graph-based structures; and second understanding and using the inherent structure and relationships within the multi-modal EHRs data. With that said, we separate the graph-based approaches in Knowledge Guided approaches, and Data-Centric approaches, where the former augments the data and the later does not. Nevertheless, conceptually they can overlap.

Knowledge Guided. *GRAM*, introduced in Choi, Bahadori, Song, et al., 2017 is one of the first examples of leveraging domain knowledge with EHRs tasks within the deep learning domain. *GRAM* integrates the CCS multi-level hierarchy [23] to create richer representations of each medical code. In *GRAM*, each medical code is represented by the combination of all of its ancestors and itself. This representation is computed by means of an attention mechanism that gives different weighs to each ancestor based on how much frequent a given medical code is present in the data. Thus, the less frequent, the more general the representation of a code will be, promoting more accurate predictions when data is insufficient. *GRAM* objective is to predict the medical codes that will occur on the next visit, using a RNN architecture to predict the *t+1* visit given the visits [1,t]. However, when data is sufficient *GRAM* has a relatively comparable performance with other RNN variants like *Dipole*. *KAME*, introduced in Ma et al., 2018 is an improved version of *GRAM*,

[22] e.g https://allennlp.org/interpret
[23] https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp

where the key difference is that the authors make use of the medical ontology throughout the entire learning process in contrast to *GRAM*. In *KAME*, the authors use the ancestors code embeddings combining them via attention similarly to *GRAM*, concatenating them with the result of the RNN component, which in turn is implemented exactly like *GRAM*. *KAME* yields better performance in all of the presented metrics. This performance boost suggest that domain-knowledge should be present throughout the entire training process and leveraged as most as possible.

In Y. Li, Qian, et al., 2020, the authors followed the similar approach of integrating domain-knowledge with EHRs. In the proposed Graph Neural Network-Based Diagnosis (GNDP) model, the authors add the ancestors of each code in the data, and use it as input, not differentiating them, on contrary to *GRAM* and *KAME*. The model was based on Yan et al., 2018, and the task at hand adapted to fit the original architecture. Conceptually, there is a graph representation for each visit, where the authors use convolution operators across the temporal dimension to create representations that can then be used for a diagnosis prediction task.

Data-centric. The aforementioned methods approach the problem of data insufficiency, more specifically data sparsity by integrating domain-knowledge with the training data. However, GNNs can also yield interesting results and applications without integrating domain-knowledge. Namely, they can be used to leverage the structure of the data, when known or even learn it, i.e learn to which diagnosis a procedure or medication refers to, based solely in EHR data.

With that in mind, we highlight two architectures, Zhu and Razavian, 2019, Choi et al., 2020. In Zhu and Razavian, 2019, the authors propose a graph classification model where the graph represents a patient, and the nodes the medical codes, with the objective of diagnosing Alzheimer's disease. In this work, the authors base their model on a variant of Veličković et al., 2018, using self-attention to learn the weight of the edges, thus creating structural aware node representations. In a similar fashion, Choi et al., 2020, first learns the structure of the data in each visit, by means of self-attention mechanisms kin to Transformer-based architectures, forcibly disabling impossible connections (e.g a treatment is only related to a diagnosis) and then applies a variant of a GCNN to further improve the learned relations. The authors named this architecture Graph Convolutional Transformer. What is more, the authors evaluate the proposed model by performing a multi-prediction diagnosis task, and graph reconstruction tasks to validate if their model is learning properly the structure of the data, showcasing its success. However, none of the architectures mentioned actually use EHRs sequential aspect, nonetheless the authors in Choi et al., 2020, argue that a RNN component could be readily added to aggregate the learned graph representations of each visit. However, few of the proposed architectures fully leverage the multi-modal nature of EHR. Processing heterogeneous graphs is indeed complex and a hot topic both inside and outside academia as briefly introduced in section 2.3.6. With that said, Heterogeneous Similarity Graph Neural Network on Electronic Health Records Z. Liu, Li, Peng, He, and Philip, 2020, as the name

suggests, leverages more than one data type, namely: diagnoses, procedures, medications, lab tests and extracted information from free-text clinical notes by identifying relevant MeSH terms. The authors split the heterogeneous graph in several homogenous graph by computing meta-paths, fusing them in a later step in order to learn what the authors define as the *true relationships* between node pairs. Afterwards, the learned graph can be used as input to any GNN architecture for a downstream task, like diagnosis prediction (phenotype prediction), which the authors evaluate their method against, showcasing state-of-the-art results within their settings.

2.5 Critical Summary

The explored related work is only a subset of the enormous research pouring out daily within the field of DL. Nonetheless, as far as we know it is clear that the more prominent architectures for the proposed task are models that easily handle sequential data, like RNNs or Transformers. However, we note that CNNs as discussed in section 2.4.2.1 can also yield interesting results if adapted to properly model time, even if they are not the more traditional approach. More specifically, one can adapt the convolutional operator to aggregate timestamped events, but only on a fixed neighborhood of time, where RNNs and Transformers tend do better, since they understand longer and more distant patterns. We also highlight the fact that several works apply models to static single-label outcomes, e.g mortality or single disease prediction, thus not being directly compatible with the desired, arguably more complex, multi-label, multi-class outcome prediction task that is multiple diagnosis prediction. Nevertheless, it should be possible for such models to be adapted for a multi-class setting and studied in the proposed context. Furthermore, the remaining of the works studied tackled the objective at hand, thus being more easily comparable, helping guiding this research.

It is also clear, that the aforementioned models suffer greatly from data insufficiency as any other DL architecture. As stated, EHRs by nature are sparse, which increments the necessity of large and curated volumes of data. Besides the traditional lack of data, within the healthcare domain it is common for a certain set of diseases or treatments to be infrequent. Unfortunately, the rare diseases tend to be fatal or present a high risk to the patient, thus, it is an important problem to solve. *GRAM*, and then *KAME* introduced an interesting way to cope with data insufficiency. The aforementioned works, plus GNDP, show a common research direction, where integrating domain-knowledge improves the learning process. Thankfully/other word, within the healthcare domain there are several ontologies, which are well-known and defined hierarchical structures that correlate concepts defining knowledge of a certain domain in a graph-like format. *GRAM* pioneered the use of domain-knowledge within EHRs leveraging a Graph-based attention model, being improved in *KAME* and then inspiring GNDP. GNDP, however, is a Graph Neural Network architecture based of spatial-temporal graph convolutions, which attempts to fully exploit medical knowledge. What is more, Choi et al., 2020,

showcased that leveraging the inherent graph-like structure of EHRs data yields better results. Thus, the literature studied points to the direction that graph-based models help tackle the data insufficiency problem but also do better than other models by explicitly using the inherent hierarchical data structure of EHRs.

In addition, interpretability is a requirement of any model that might be applied within the healthcare domain, and as we know, ANNs are traditionally seen as "Black-boxes" where understanding their reasoning is typically complex. The interpretability issue has been tackled in several ways, however, we highlight attention mechanisms, which have been proven and used to successfully interpret the reasoning behind models' predictions. A good example of that within the domain of this work, is *BEHRT*, where we can explore how diagnoses correlate to each other, exploring the attention weights learned during training. Thus allowing to explore comorbidities and reason on why a certain diagnose was given by the model.

To conclude, the research so far suggests that combining GNNs, attention mechanisms and domain-knowledge are a good and sound attempt at improving the current SOTA multiple diseases prediction models, specifically in few-shot scenarios, whilst not sacrificing the interpretability requirement. What is more, generally speaking this work is mostly inspired by Choi, Bahadori, Song, et al., 2017, Ma et al., 2018,Agarwal et al., 2019,Y. Li, Rao, et al., 2020, 1,Choi et al., 2020 and Y. Li, Qian, et al., 2020, thus, we will follow the recommendations and directions of the aforementioned literature in the implementation of this work.

3

Health Outcome Pathway Prediction

3.1 Introduction

In this chapter we introduce the health outcome pathway prediction architecture, consisting of a extensible and modular graph neural network based architecture for predicting health outcomes of patients given their longitunal health records (EHRs). What is more, we also define the developed embedded data augmentation technique which leverages ontologies and a set of auxiliary tools to assist physicians to reason and explore the predictive model results. We organize this chapter, inspired by Y. Li, Qian, et al., 2020, by first defining the pathway prediction task and the necessary notations and formalisms. Then, we describe the developed GNN architecture detailing all of its components. We also elaborate on the methodology followed for this work, namely we detail the ETL processes that transform MIMIC III and eICU into the expected input, followed by the experimental setup used to evaluate the proposed architecture across diverse settings, also highlighting reproducibility found during the implementation of this work.

3.2 Pathway Prediction

We define pathway prediction as predicting or suggesting one - binary label - or multiple health outcomes - multi label - that are likely to occur in a patient health journey, being diagnoses, procedures, medications, frailty score, mortality risk, among others. In order to tackle this family of tasks we designed and developed the Health Outcome Pathway Prediction (HOPP) framework to assist in the implementation and analysis of the subtasks pertaining to the pathway prediction family. That being said, the Health Outcome Pathway Prediction (HOPP) framework is a library with a set of tools designed to help develop several variants of the HOPP architecture and handle all the required data manipulation, whereas the HOPP architecture is a graph-based approach prepared to predict multi health outcomes based on sparse and multi-modal EHRs. It is implemented with the goal of being extensible to more datasets and tasks besides the ones explored in this

work. What is more, it also permits a wide variety of configurations through several parameters. By default, HOPP makes use of ICD-9 diagnoses codes distributed throughout time (i.e through visits) of a patient, augmenting its information with a medical ontology constructed from the Clinical Classification Software (CCS) Multi-Diagnoses[1] ontololgy. With that being said, the architecture outputs a list of predicted phenotypes (i.e diagnostic code prediction), sorted from most likely code to less likely. The granularity of the predicted codes can vary. As a matter of fact within the ICD coding scheme, there are several ways to correlate codes to pre-defined groups, i.e., instead of predicting the code 018.01 (Acute miliary tuberculosis, bacteriological or histological examination unknown (at present)), we can generalize to the more broad condition, being in this example tubercolosis. Intuitively, the granularity of the predicted codes impacts greatly the performance of the architecture and its usage in real world settings. Hence, we designed and implemented the framework such that it offers a configurable level of granularity of the predicted classes - 14567[2] ICD-9 codes, 19 ICD-9 Chapters, 283 CCS Single Grouped diagnosis codes, or 1778[3] category-level codes - where it allows the user to configure which variant of the architecture to use and also handle the data accordingly. Both 283 CCS Single Grouped and 1778 category level codes have been discussed to be sufficient for real case use (Choi, Bahadori, Song, et al., 2017; Ma et al., 2018), reducing training time while boosting the models' performance. In addition, similarly to diagnoses, the framework also offers a configurable granularity for procedures. However, within pathway prediction tasks we also have binary tasks like mortality prediction which yield a single value between 0 and 1 for each patient, representing the risk score for each patient for that given task. With that in mind, similarly to the concept of ranking sorting diagnoses, we can, given each patient risk score, also rank patients according to their risk so that clinicians can prioritize the care of patients with predicted high risk. Hence, the systems implemented through this framework offer actionable insights for the clinicians for both patient-level and cohort-level.

Due to the underlying design of the architecture being a DL model, which learns representations of the data, HOPP also offers the possibility to project the representations to lower dimensional space to correlate different aspects of the data, possibly uncovering hidden relationships (e.g between patients or diagnoses). That being said, the framework can then rank patients according to their similarity throughout their health journey by means of cosine similarity (eq. (3.1)) of the learned representations.

$$cos(\theta) = \frac{A.B}{||A||.||B||} \tag{3.1}$$

Specifically, each visit's representation is compared to all other visits, with the possibility of filtering throught similar health journeys i.e., compare a certain visit t with only other visits t of other patients.

[1] https://www.hcup-us.ahrq.gov/toolssoftware/ccs/AppendixCMultiDX.txt
[2] Extracted from https://www.cms.gov/Medicare/Coding/ICD9ProviderDiagnosticCodes/codes
[3] Extracted from http://www.icd9data.com/2015/Volume1/default.htm

The system originated from the framework, similarly to previous works, aims to emulate the physicians' thought process by outputting a sorted list of possible diagnosis (or other outcome), which can be later trimmed down, re-ranked and analysed by a domain expert (medical doctor, clinical staff, etc). It is a common practice for medical doctors to perform differential diagnoses, where they compare several diagnosis selecting the most likely, trimming down unlikely scenarios. The proposed framework aims to facilitate this process by giving the tools to explore 'what if' scenarios, similar to Kwon et al., 2019, and be in total control of the diagnosis process.

That being said, the framework is readily prepared to be used within the context of FrailCare.AI, where the effort would only need to be focused in processing the dataset, where the framework offers guidelines and interfaces in format of APIs to do so. In fig. 3.1 we overview our architecture feature, which we will describe in more detail further ahead.

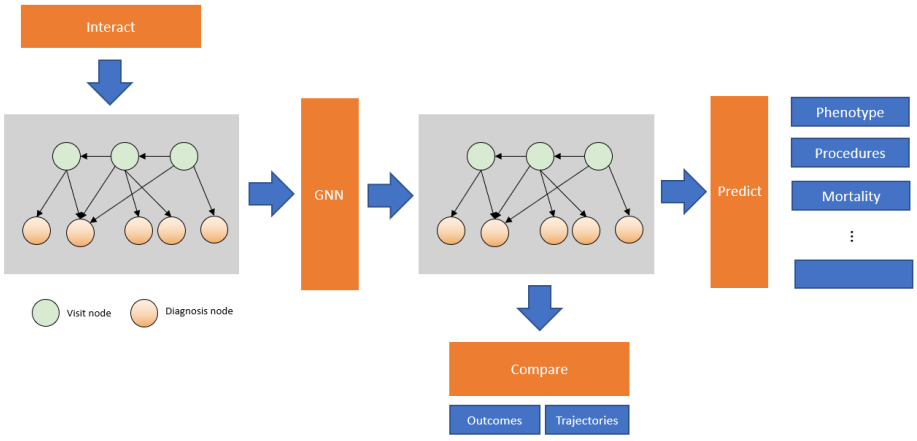

Figure 3.1: Predictive model overview

What is more, the proposed predictive model can then be inserted in systems as illustrated in fig. 3.2. Specifically, systems that make use of EHRs, with the goal of showcasing the relevant informations to the domain experts, augmenting their capabilities improving patient care.

3.3 Basic Notations

We define the set of medicals codes (i.e diagnoses, procedures, medication codes and so on), from the EHR data as M = $\{m_d1, m_d2, m_p1, .., m_{|K||J|}\}$ where $M_k|J|$ is the total number of codes of modality k, whereas the set of modalities present in the EHR data is defined as K, which can differ between datasets. Thus, we write $M = \bigcup_{j \in K} \bigcup_{i \in m_j} m_{ji}$, extending the definition in Y. Li, Qian, et al., 2020. Furthermore, and similarly to Y. Li, Qian, et al., 2020, a patient p who has T visit records can be represented as a sequence of visits: $P_p = \{x_1^p, x_2^p, ..., x_T^p\}$, where each visit x_t^* from an arbitrary patient contains multiple medical codes from the code set M ($x_t^* \subseteq M$). In addition, when $k = d$ or $k = p$ - medical codes

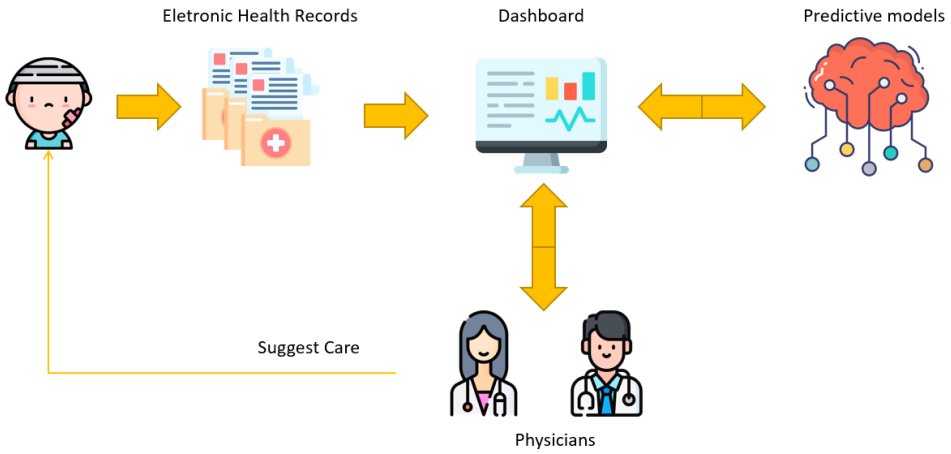

Figure 3.2: Health Outcome Pathway Framework usage

that pertain to diagnoses and procedures - respectively, we also define two distinct sets of medical ontologies for ICD-9 codes. An ontology represents knowledge specifiying meaning through properties and relationships between concepts. Intuitively, an ontology can be represented as a graph, where each node represents a concept and each edge represents a relationship between two nodes. Formally, $G = (V, E)$, where V is the set of vertices and E the set of edges such that $(u, v) \in E$ if, and only if, the vertices u and v are connected in any direction. However, if the direction of the connection can be specified, then the graph is defined as directed.

For that effect, similar to previous works, we make use of the CCS ontology which is composed of a single mapping ontology $f(x) : x \longrightarrow group$, where $||group|| = 283$, and a multi-level parent-child ontology, defined by two independent graphs, $G = G_d \cup G_p$. The former correlates diagnoses and the latter procedures. Hence, all of ICD-9 diagnoses and procedures codes are vertices of G, thus $M_d, M_p \in V$.

With that said, the leafs nodes of G are the lowest level representing the specific medical concepts M_d or M_p. The nodes in upper levels, refered as ancestors nodes, are more general. For example, the specific medical concept with the ICD-9 code *1534* (Malignant neoplasm of cecum) has the ancestor *Cancer of Colon*, which in turn has *Colorectal Cancer* as its ancestor, and finally *Neoplasms* as the most general category.

We note that in this work we use the terms *hidden representations* and *node embeddings* interchangably since we are working within the graph domain. Thus, learning *hidden representations* of the data, which are represented by nodes, is equivalent to learning *node embeddings*.

3.4 Graph EHR Construction

Making use of the notations defined in section 3.3, we build an EHR graph for an arbitrary number of patients, augmented with the extracted medical ontology G from CCS publicly available files.

Medical Ontology Construction The medical ontology G was extracted from *Appendix C: Multi-Level Diagnoses*[4] and *Appendix D: Multi-Level Procedures*[5], by parsing the .txt files with regex, identifying the ICD codes and the respective group where they belong, creating a directed edge between each ICD-9 code to its group and each group to its ancestor, using networkx[6] python library. In addition to creating the ontology tree, following Choi, Bahadori, Song, et al., 2017 work, we also create directed edges between all of a node ancestors and itself, so that each node can have direct access to their ancestry in one hop, as illustrated in fig. 3.3. Thus, even if the original ontologies are trees, the outputed ontology is not, since each node can have more than one parent. The medical ontology G has 20329 nodes and 92374 edges.

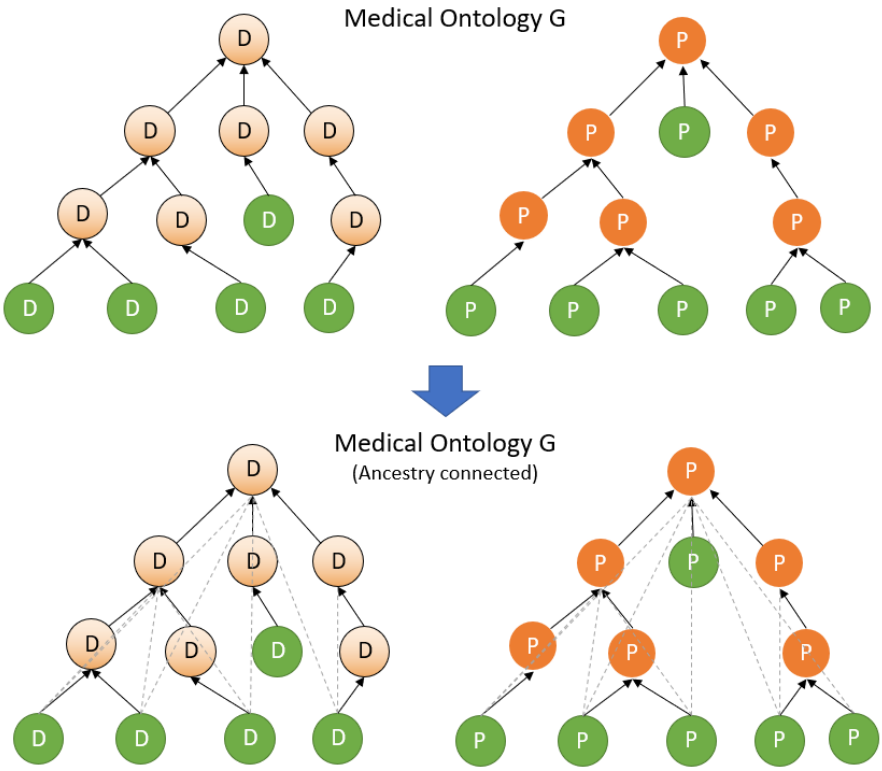

Figure 3.3: Medical Ontology illustration, the grey dotted lines represent the added edges to allow 1-hop between nodes. D are diagnosis nodes and P procedure nodes.

Graph EHR Construction Given a EHR data set with an arbitrary number of modalities and medical codes, we extract a Knowledge Graph represented as $KG = (V, E, R)$, where

[4] https://www.hcup-us.ahrq.gov/toolssoftware/ccs/AppendixCMultiDX.txt
[5] https://www.hcup-us.ahrq.gov/toolssoftware/ccs/AppendixDMultiPR.txt
[6] https://networkx.org/

V are the nodes, E the edges, and R the relationship types, using the medical ontology G as the starting point. The Knowledge Graph has two relationship types, *spatial*, which connects modalities to visits and *temporal*, which connects visits between themselves. The construction of the graph format *EHR* is as follows: given a patient with a sequence of visits, $P_p = \{x_1^p, x_2^p, ..., x_T^p\}$, for each visit, create a visit node and create a directed edge between each modality present in that visit to that specific visit, adding a new node to the graph if not present, initializing it accordingly. Furthemore, for each x_t^p, a directed edge is also created between all of the previous visits and t. We note that since the *GRAPH EHR* is built from the medical ontology G, each *Diagnosis* and *Procedure* code present in the EHR are connected to the ontology G. In fig. 3.4 we illustrate the output for one patient.

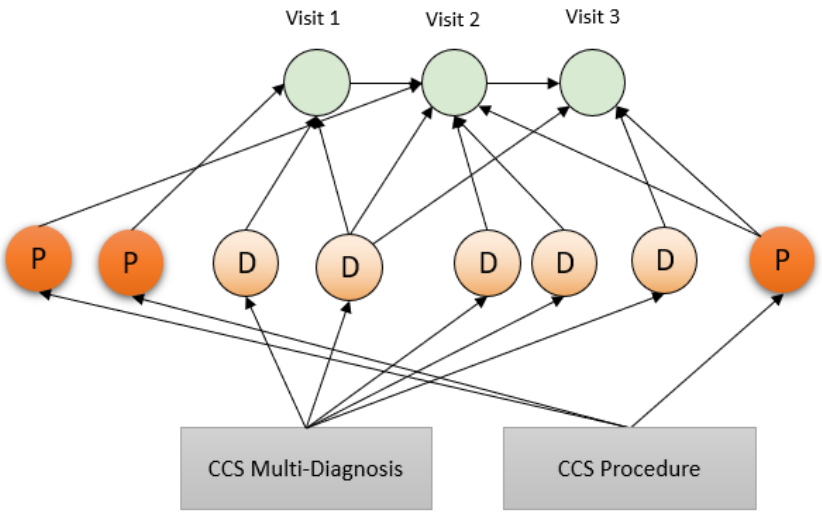

Figure 3.4: EHR Graph for one patient using diagnoses and procedures

We are aware that each procedure, should in fact, be connected to a specific diagnosis, however both MIMIC III and eICU do not disclose that information. Nonetheless, it should be possible to augment the graph EHR by learning those relationships, similar to Choi et al., 2020. We leave this component for future work, since it would require a more elaborate graph construction and more computational resources.

3.5 Architecture

The developed Deep Learning (DL) architecture key components are GAT layers, which are the building blocks of Graph Attention Networks, originally designed by Veličković et al., 2018. Implementantions of these blocks can be found across frameworks like PyTorch Geometric[7], which we leverage for this work. The proposed DL architecture goal as introduced in section 3.2 is of representation learning, that is, learn the representation of each

[7] https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html

visit node, and then learn how to map between the learned representation and the target - one or multiple health outcomes. Thus, predicting the future health state of a patient. The model originated from the proposed architecture is applied on heterogeneous data - we always have at least two node types (visit and a modality node e.g., diagnosis). Hence, every node embedding resides within the same latent space. Taking the example of the simplest instantiation of this architecture, where we leverage only diagnosis information, the architeture goal is to learn each diagnosis importance given the future dianogses of the patient. More specifically, learn how to aggregate each diagnosis representation at a visit level, whereas each visit also learns how to aggregate information of previous visits. Thus, a visit represents a specific health state in a certain point in time of a patient, which in turn is a aggregation of the diagnosis given in that visit and previous ones.

With that said, here we will describe the components of this architecture in detail, starting with the GAT layer, and then desfine the output layer that maps the visit node representantions to the expected outputs and their respective hyperparameters.

3.5.1 Graph Attentional Layer

A Graph Attention Network (GAT) generalizes further Graph Convolutional Networks (GCNs), by leveraging attention mechanisms that learn the importance of each edge, instead of simply convoluting the neighbors with the same weight. Nonetheless,, we highlight that Graph Convolutional Network (GCN) have been successfully used within the context of this research as shown in Y. Li, Qian, et al., 2020.

Veličković et al., 2018 defines that a graph attentional layer objective is to compute representations h (eq. (3.3)) for each node, aggregating each of their respective neighbouring nodes representations based on learnable *attention* (eq. (3.2)).

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in N_i} exp(e_{ik})} \tag{3.2}$$

$$h_i = \sigma\left(\sum_{j \in N_i} \alpha_{ij} W h_j\right) \tag{3.3}$$

Conceptually, a graph attentional layer performs the task of link prediction (eq. (3.4)) weighting the edges accordingly, and then computes the reprensentation of a node as the weighed average of its neighbouring nodes, as illustrated in fig. 3.5.

$$e_{ij} = \alpha(W h_i, W h_j) \tag{3.4}$$

As proposed by Vaswani et al., 2017, Veličković et al., 2018, also makes use of multi-head attention eq. (3.5) in its definition of the graph attentional layer. Multi-head attention promotes more consistent results.

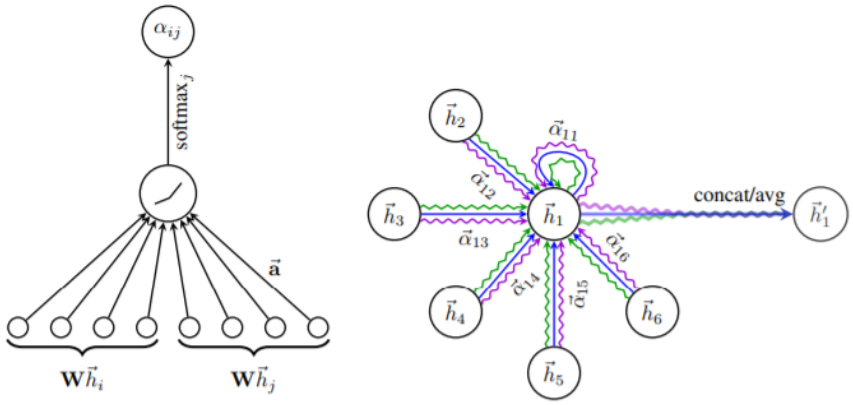

Figure 3.5: left: Edge computation; right: Aggregation

$$h_i = \|_{k=1}^{K} \sigma\left(\sum_{j \in N_i} \alpha_{ij}^k W^k h_j\right) \tag{3.5}$$

Thus, the Graph Attentional Layer (*GATLayer*) can be defined as applying eq. (3.5) across all nodes, simultaneously, which returns a list of hidden representations h of length |N|, where |N| is the number of nodes in the graph.

That being said, after each *GATLayer* pass we apply a LeakyReLU activation function, as defined previously in eq. (2.4).

3.5.2 Output layer

The output layer, as initially introduced in section 3.2 is a set of MLPs ($f(h) = Wh+b$), one for each task, where its output size is the same as the number of classes of its task. Each MLP is applied to the node embeddings of the last visit of each patient, returning $\hat{y}_i$, which represent either the probabilities of each class for the task for which it is trained - if a multi-label task - or the risk score - if binary task -, as illustrated in fig. 3.6. What is more, for multi-task scenarios we can chain the MLPs, such that we increase the correlation between tasks. Specifically, given an ordered set of tasks A, B, C, with the respective MLPs (MLP_A, MLP_B, MLP_C), we have that $MLP_A = W_A h + b_A$, $MLP_B = W_B(h \| MLP_A) + b_B$ and $MLP_C = W_C(h \| MLP_B) + b_C$, where $\|$ represents the operation of concatenation.

With that being said, for the multi-label tasks, with the goal of returning a set of probabilities that sum up to 1, we need first to apply a softmax function across the resulting array per patient, whereas for binary label tasks, we apply a sigmoid.

3.5.3 Hyperparameters

The proposed architecture, similarly to other DL architectures has a set of hyperparameters that impact the architecture performance, and can be either defined a priori or

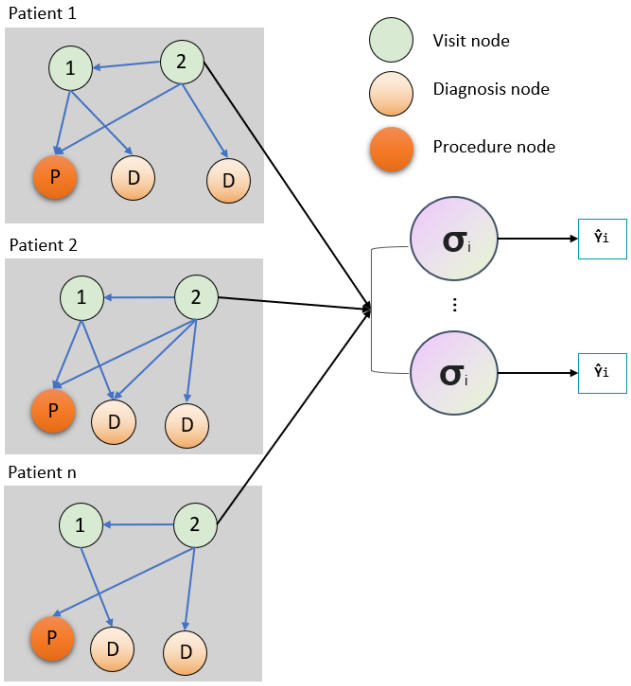

Figure 3.6: For the sake of simplicity we showcase all patients with two visits without any common diagnosis or procedure

fine-tuned with different datasets or portion of a dataset to avoid data leakage. With that said, the implemented architecture has the following five hyperparameters to be defined:

1. The number of *GATLayer*, which correlates to the depth of the model.
2. The embedding size of the nodes, which is the size of the latent space, thus correlating directly with the architecture capability to generalize.
3. The activation function between layers, which perform non-linear transformations, are fundamental to the design of any neural network.
4. The number of heads in the multi-head attention layer, which is the number of attention heads that the model will use.
5. The embedding initialization procedure, which can be either random or pre-trained.

All of which were tuned accordingly the validation set as further detailed in the following section.

3.6 Experimental Setup

In this section we describe the methodology and experimental setup used to evaluate the proposed architecture; namely the metrics and used datasets. We make use of weight and

biases[8] (Biewald, 2020) to visualize the performance of our models throughout training, logging a different variety of metrics to facilitate further analysis. However, in this chapter we focus on a small subset of configurations, thus for more in-depth experiments and discussion please refer to chapter 5.

With the goal of being as objective as possible to validate the proposed framework we attempt to follow as near as we can the experimental setup of the relevant related literature. However, some variations are to be expected, mostly because the studied experimental setups are not often documented or present several confusing and conflicting definitions of metrics and their respective meaning. For example visit-level precision or accuracy@k, which although are named differently are defined similarly in some of the studied literature.

Nevertheless, we follow the intuition of several works referenced throughout this thesis that use recall (or a similar metric) to evaluate the performance of the models, since it resembles how a physician evaluates their hypothesis. With that being said, the task of phenotype prediction, or any other health related multi-label task can be seen as a de facto information retrieval task where the goal is to rank documents according to a query. In this case, we rank health outcomes (the documents) based on the patient visit history (query). Thus, measuring the performance of the proposed model is equivalent to measuring the performance of a search engine, where the more relevant documents are ranked higher (i.e., the more likely diseases to occur, the patient with the most mortality risk, the most likely required procedure, and so on). Furthermore, within the context of FrailCare.AI, we are interested in ranking patients according their frailty score, thus the justification for the metric used still holds.

Most of the experiments were performed in shared instances with 64 GB RAM and 11GB GPU, thus the hardware was not always fully available, with Intel Core i7-3930K Processor (6 cores, 12 threads, 12M cache, 3.2 GHz, up to 3.80 GHz). We note that the CPU count was not that relevant since we do not parallelize any computation, while on the other hand, the available GPU and RAM constrained greatly the development of this work. In addition, later on we were forced to switch development enviroments due to some limitations, choosing paperspace[9], where we used instances with 16 GB GPU and 30 GB RAM with 8 CPU's.

3.6.1 Metrics

We mostly make use of Recall at K (R@K) to evaluate the performance of the proposed framework, as defined below.

$$Recall@K = \frac{|relevant\ documents@k \cap retrieved\ documents@k|}{relevant\ documents@k}$$

, where relevant documents@k represent the top k relevant documents (the real labels), and retrieved documents@k represent the top k documents that were retrieved by the

[8] https://wandb.ai/site

[9] https://www.paperspace.com/

system (the predicted outcomes). Starting from the basic definition of recall, we then transform it to better fit the task at hand, defining it similary to *visitlevel-precision@k* metric used in Y. Li, Qian, et al., 2020,

$$Recall@K = \frac{|\hat{y}_{correct}|k}{min(k, Y)}$$

where $|\hat{y}_{correct}|k$ is the list of the top k correctly predicted outcomes as defined in eq. (3.6), and Y the lenght of the real outcomes.

$$|\hat{y}_{correct}|k = \sum(\hat{y}_k \cap Y) \tag{3.6}$$

where $\hat{y}_k$ are top k predicted probabilities and Y the real outcomes - a multi-hot array -, thus the intersection of the top k predicted outcomes and the real outcomes yield the number of correctly predicted outcomes at k.

We note that the aforementioned definition can only be used to evaluate multi-label tasks. For instance, in the case of a binary classification problem, the definition of recall is equivalent to the definition of precision. Thus, in order to maintain the notion of recall, when p is a binary classification task, we define the metric across patients. Hence, we rank higher patients that have a higher probability of having a certain outcome, e.g mortality, unplanned re-admission or any other binary outcome. The remaining computation of the metric is similar. Nonetheless, we also evaluate binary tasks individually with metrics like accuracy, however due to the unbalanced nature of the data, we focus on sensitivity eq. (3.7) and specificity eq. (3.8)).

$$sensivity = \frac{TP}{TP + FN} \tag{3.7}$$

$$specificity = \frac{TN}{TN + TP} \tag{3.8}$$

These two metrics helps us understand how well the model identifies the positive class and the negative class, individually. Studying the sensivity of the model reasons with the fact that it is better to have a false positive than a false negative, thus the more sensible the better, and the more specific the worse.

3.6.2 Datasets

At the time of the implementation, and consequent writing of this work, we still had no access to the dataset that pertained to the FrailCare.AI project, thus we used the datasets that were both readily available and relatively frequent throughout the literature: MIMIC III and eICU, both ICU datasets. Here we overview both datasets, describing them briefly, showcasing in table 3.1 the distribution of the number of admissions (both absolute and normalized) and in table 3.2, the number of patients, admissions, diagnosis and their respective frequency. All of the described statistics are computed prior to any processing. We note that the category level codes are calculated retieving the first 3 digits of the

existing ICD-9 codes (e.g 539.01 - Infection due to gastric band procedure - category level is 539 - Complications of gastric band procedure). Additionally, we identify the rows with * that can not be directly compared since different coding schemes are used.

Table 3.1: Number of patients per admissions count

ADMISSIONS	MIMIC III	(%)	eICU	(%)
1	38.983	(84%)	100.884	(72%)
2	5.160	(11%)	26.554	(19%)
3	1.342	(3%)	6.612	(5%)
4	508	(1%)	2.899	(2%)
5	246	(0.5%)	1.094	(0.7%)
6	113	(0.2%)	603	(0.4%)
7	51	(0.1%)	305	(0.2%)
8	31	(0.07%)	153	(0.2%)
9	26	(0.06%)	88	(0.1%)
10+	60	(0.07%)	109	(0.4%)

In table 3.1 we can easily grasp that MIMIC III is both smaller in absolute size and in visit per patient. It proportionally has less visits per patient, for example 28% of the patients in eICU had 2 or more visits, whilst in MIMIC III only 16%.

Table 3.2: MIMIC III and eICU statistics

	MIMIC III	eICU
Patients	46.520	139.367
Patients with only one admission	38.983	100.884
Patients with two or more admissions [1]	7.537	38.483
Admissions	58.976	200.859
Avg admissions per patient	1,268	1,441
Unique diagnosis	6.984	1.072
Total diagnosis	651.047	4.550.787
Unique Procedures*	2.009	2.711
Total Procedures*	240.095	3.688.745
Avg diagnosis per admission	11,039	4,612
Avg diagnosis frequency per visit	1.1%	0.4%
Max diagnosis frequency per visit	36%	20%
Std diagnosis frequency per visit	3%	1%
Avg CCS diagnoses frequency per visit	3.9%	2.5%
Max CCS diagnoses frequency per visit	46%	32%
Std Dev CCS diagnoses frequency per visit	6.7%	4.6%
Unique category-level diagnosis	943	770
Unique category-level procedures*	711	611

However, table 3.2 showcases that MIMIC III albeit being smaller in size has a wider

diversity of diagnosis while compared to ICU, both in frequency (in average each diagnosis occurs more frequently), and in uniqueness (there are more unique diagnoses).

3.6.2.1 MIMIC III

MIMIC III , described in Johnson et al., 2016, 1, is an acronym to "Medical Information Mart for Intensive Care". MIMIC III is well-known dataset within the academia, where it encompasses data from 40,000 patients who stayed in critical care units (ICUs) of the Beth Israel Deaconess Medical Center between 2001 and 2012, with 58,976 admissions in total. MIMIC III has been used across the literature referenced on this manuscript. Thus, it acts as a standard dataset with some publicly available results to benchmark our architecture against. However, it is a rather small dataset compared to others which might difficult some tasks that typically require larger amounts of data.

MIMIC III , has exactly 46,520 patients, where 38,983 were admitted only 1 time. Thus, this dataset only has 7,537 patients with two or more visits. We arguet hat in the case of the trajectory prediction task, this dataset might be a challenge to work with, since there is not enough historical data to learn from. Furthermore, out of this 7,537 patients with more than 1 visit, more than half of them only had 2 visits. Whereas the maximum visits a single patient had was 42. We refer to Rodrigues-Jr, Spadon, Brandoli, and Amer-Yahia, 2019, where the authors explore the expected challenges of working with MIMIC III , for a more thorough review. What is more, MIMIC III uses the ICD ninth revision coding scheme to identify both the diagnosis and procedures.

3.6.2.2 eICU

eICU Collaborative Research Database is a multi-center database comprising deidentified health data associated with over 200,000 admissions to ICUs across the United States between 2014 and 2015, described in Pollard et al., 2018. eICU, has exactly 139,367 patients, where 100,884 were admitted only 1 time and 38,483 more than once. This dataset compared to MIMIC III favors more the good execution of a trajectory prediction task since there are more trajectories to learn from. Nonetheless, in similar fashion with the MIMIC III dataset, the vast majority of patients only with more than 1 visit, only had 2 visits. Whereas the maximum visits a single patient had was 26. eICU in contrast to MIMIC III uses both the ninth and the tenth revision of the ICD coding scheme.

3.6.3 Extract-Transform

Our architecture works within graph domain thus we need to transform the traditional relational tabular format to a graph format. Here we detail all the steps involved in this process, as illustrated in fig. 3.7. However, we note that for sake of simplicity the description and the actual implementation may differ slightly since we detail the implementation

conceptually and do not specify coding details. Nonetheless, the actual implementation can be found in our public github repository[10].

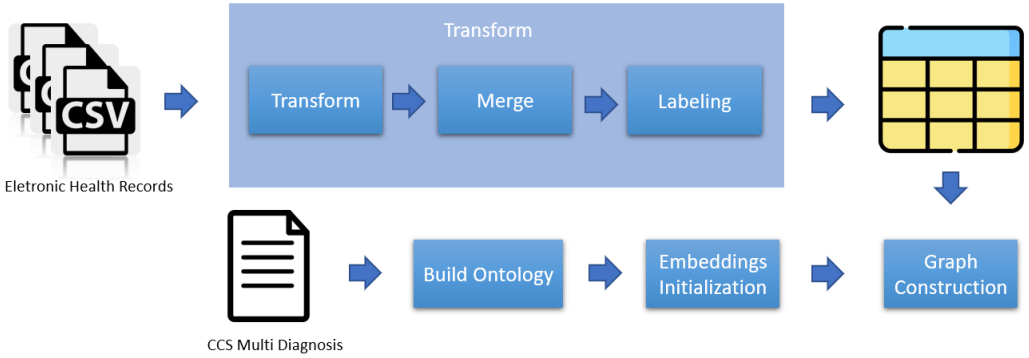

Figure 3.7: Extract-Transform illustration

Both datasets were extracted from their source at physionet as .csv files. We also note that the entire pipeline is highly configurable, but for the sake of simplicity we will focus the following discussion on tho the outcome prediction task as proposed in this chapter.

3.6.3.1 Transform

From MIMIC III we extract ICD9 diagnoses and procedures, spanning across visits for each patient. For that effect, the necessary data is stored in these 3 following tables: ADMISSIONS, DIAGNOSES_ICD, PROCEDURES_ICD. For eICU, we needed only admissionDx and diagnosis, since no procedures data was available in this dataset.

We used *ADMISSIONS* table as the pivot table since it correlates a patient to a visit. Thus, merging both *DIAGNOSES_ICD* and *PROCEDURES_ICD* with *ADMISSIONS* we are able to identify which diagnoses and procedures each patient had per visit. *ADMISSIONS* also defines the death time of a patient in a visit, or NaN if the patient is still alive. Afterwards, we create a pair (D,Pr) for each visit, where D is the list of diagnoses and Pr is the list of procedures. Then, we group each visit by patient yielding in a list of pairs (D, Pr) for each patient. For the eICU dataset we also made use of *patient* table, which jointly with the *admissionDx* table presents the same role as *ADMISSION*, whilst *diagnosis* shares similar information to *DIAGNOSES_ICD*. However, in this case, because eICU presents more than one diagnoses per row, we need first to unwrap it so that there is only one diagnosis per row. Furthermore, since eICU does not have procedures, a visit is simply defined by a list of diagnoses D. Afterwards, we collect all diagnoses per visit, and then all visits per patient, yielding in a list of lists for each patient, similarly to MIMIC III . At this point, both MIMIC III and eICU are in the following format:

[10]https://github.com/Tekaichi/HOPP-code

- $(D, Pr) \in V \in P$ which reads: a tuple of diagoses and procedures belongs to a visit, and a visit belongs to a patient;
- $D \in V \in P$, which reads: a list of diagnoses belongs to a visit, which belongs to a patient.

Additionally, since we are focusing on phenotype prediction, which is a supervised task, we need to retrieve labels for each patient. We use the information of the last visit for that effect, storing it separately. Thus, separating x - the training data - from y - the labes - in this step, where x is every visit except the last, and y the last visit, for each patient. From this point, we can configure the grouping procedure - ICD9-Chapters (19), CCS Group (283), 3-digit level (1778) or 2nd hierarchy category level (184), for ICD-9 diagnosis, and CCS Group (231), for procedures. The grouping is executed for each diagnosis and procedure in the last visit of every patient (y), where we map each ICD-9 Code to the configured group. Nevertheless, the implemented methods also allow the grouping procedure to be executed throughout the entire dataset (x), which on one hand reduces the dimensionality of the input, but on the other, it disables the use of G, since the medical ontology correlate specific ICD-9 codes. The described process is illustrated in fig. 3.8 for MIMIC III with CCS grouping.

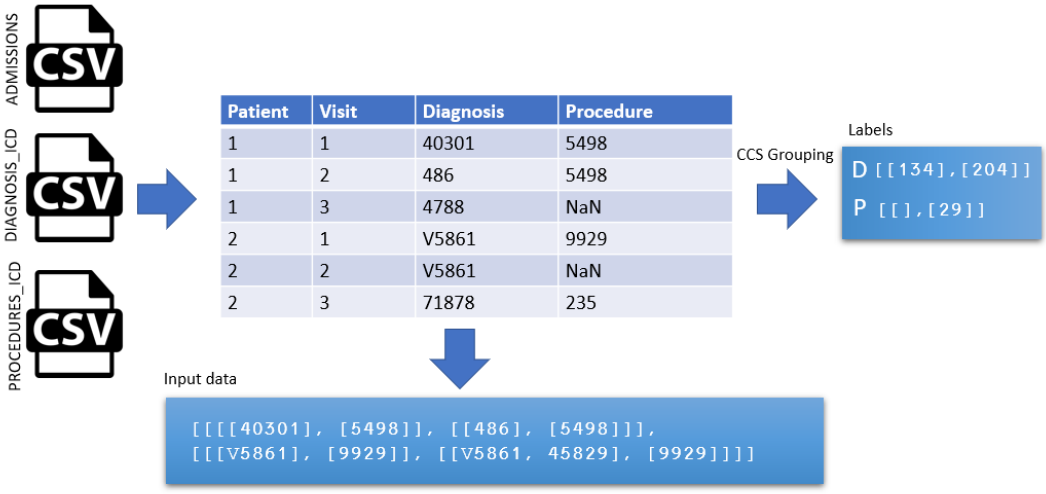

Figure 3.8: MIMIC III pre-process illustration

As the last step to prepare the dataset for a supervised task we need to encode the labels. With that goal, we unwrap the pairs (D,Pr) of y for each patient into two separate lists, $\hat{D}$ and $\hat{P}$, which correspond to y_d and y_t when task t = procedure suggestion, respectively, only for MIMIC III . For eICU we simply use D. We then perform multi-hot encoding using sklearn multi label binarizer[11], where we can configure if we want to use all of the existing classes, even if they do not occur, or if we want to use only the classes

[11] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html

that occur in the dataset. The default setup is to always use all classes, so that at end of the data pipeline both MIMIC III and eICU have the exact same format, thus allowing the proposed models to be trained and used on both, interchangably.

In addition, in order to pre-process the data for the in-hospital mortality, mortality @ 30 and 90 days, we make use of *ADMISSIONS*, for MIMIC III , and *patient*, for eICU, and define the labels for said tasks, by settting mortality at 30 or 90 days as 0 if the person did not die in up to 30 or 90 days after the previous visit, or else, as 1, respectively. For the case of in-hospital mortality, we simply assign the label 1 if the patient died in a visit, and 0 if not. We note that for eICU we are not able to compute mortality @ 30 and 90 days due to lack of proper timestamped death times.

We designed and implemented our own transform pipeline that met our needs. Nonetheless, we identify one publicly available work that maps MIMIC III to the standard FHIR (Franz, Shrestha, & Paudel, 2020), which we could have build upon. HL7 FHIR[12], an increasing standard within healthcare, could act as a common intermediate format. Thus, the deep learning researchers and developers would need only to process the FHIR format to fit their respective architectures, whilst every dataset would be mapped to FHIR, standardizing part of the process, promoting interoperability and reproducibility.

We implemented the aforementioned steps in two classes divided by MIMIC3.py and eICU.py, which can be found in our github repository [13].

3.6.3.2 Graph Construction

Afterwards the aforementioned step of data pre-processing, we have a list of lists of tuples with two items where each one is a list for MIMIC III and a list of lists of lists for eICU. The first dimension represents a patient and the second dimension represents a visit. For MIMIC III the third dimension is a tuple that has a list of the diagnoses given for that visit, and a list of procedures also given in that visit. Whereas for eICU, the third dimension is a list of diagnoses.

With that said, the final step in the Extract-Transform pipeline is to transform the processed data into graph format as previously described in section 3.4. We highlight that the graph topology is critical for the performance of the models, namely the direction of the edges, which decides how information propagates throughout the graph. The default graph EHR has the following relationships: (V,D) and (V_{t+1}, V_t), which translates to diagnoses feeding to visits and visit t feeding into visit $t+1$. Hence, diagnoses do not have access to visits, which prevents diagnoses to learn co-ocurrence based representations since a visit is the aggregation of diagnoses and past visits. We leave for chapter 5 further experiments regarding the impact of graph topology.

In fig. 3.9 we illustrate the input and the output of the process for a small number of patients and visits, where we showcase two distinct graphs, one for each patient for the

[12] https://www.hl7.org/fhir/
[13] https://github.com/Tekaichi/HOPP-code

sake of simplicity. However, they are in fact sub-graphs of a larger graph with several patients that are interconnected by their diagnoses and procedures, in addition to the medical ontology G that correlates all diagnoses and procedures.

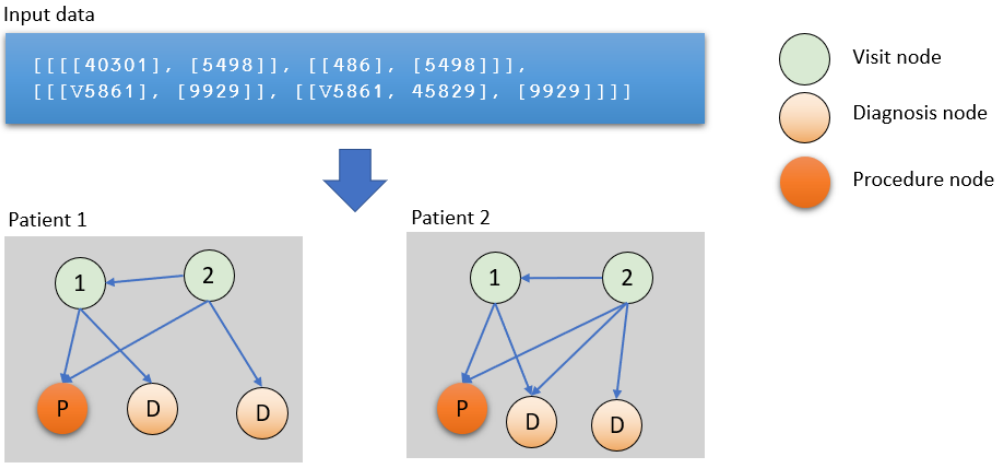

Figure 3.9: MIMIC III graph construction illustration.

3.6.3.3 Label processing

With the objective of comparing the developed architecture performance with several related works, we implemented a configurable label parser at the end of the transform step described above. This parser expects a diagnosis or procedure and the respective label option - None, Single CCS Grouping, 3-digit or 2nd hierarchy, for procedures we implemented only Single CCS Grouping and 3-digit. Here, we describe only the implemented parser for diagnoses, since the functionality is similar to procedures, only with a different source of truth.

None. This option uses the entire ICD-9 vocabulary available as extracted from the file *Version 32 Full and Abbreviated Code Titles – Effective October 1, 2014 (ZIP)* downloaded from cms.gov[14], which contains the latest version of the ICD-9 diagnosis vocabulary. Hence, the necessary label processing with this option is inserting zeroes in the multi-hot encoded vector resulting from applying sklearn multi label binarizer in the dataset, for each ICD-9 code that is present in the dataset.

Single CCS Grouping. GRAM (Choi, Bahadori, Song, et al., 2017) utilizes this processing where they map each ICD-9 code to one of 283 groups (for diagnoses) as defined in a file[15] availble by the Agency for Healthcare Research and Quality[16]. With that said, we first downloaded the necessary file and then apply regex to separate the groups from

[14] https://www.cms.gov/Medicare/Coding/ICD9ProviderDiagnosticCodes/codes
[15] https://www.hcup-us.ahrq.gov/toolssoftware/ccs/AppendixASingleDX.txt
[16] https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp

the specific ICD-9 codes that they represent, creating a dictionary for later use where the key is the specific ICD-9 code. In fig. 3.10 we display the first two groups as shown in the original file.

```
Appendix A - Clinical Classification Software-DIAGNOSES (January 1980 through September 2015)

Revised 03/24/2016

1    Tuberculosis
     01000 01001 01002 01003 01004 01005 01006 01010 01011 01012 01013 01014 01015 01016 01080 01081 01082 01083 01084 01085
     01086 01090 01091 01092 01093 01094 01095 01096 01100 01101 01102 01103 01104 01105 01106 01110 01111 01112 01113 01114
     01115 01116 01120 01121 01122 01123 01124 01125 01126 01130 01131 01132 01133 01134 01135 01136 01140 01141 01142 01143
     01144 01145 01146 01150 01151 01152 01153 01154 01155 01156 01160 01161 01162 01163 01164 01165 01166 01170 01171 01172
     01173 01174 01175 01176 01180 01181 01182 01183 01184 01185 01186 01190 01191 01192 01193 01194 01195 01196 01200 01201
     01202 01203 01204 01205 01206 01210 01211 01212 01213 01214 01215 01216 01220 01221 01222 01223 01224 01225 01226 01230
     01231 01232 01233 01234 01235 01236 01280 01281 01282 01283 01284 01285 01286 01300 01301 01302 01303 01304 01305 01306
     01310 01311 01312 01313 01314 01315 01316 01320 01321 01322 01323 01324 01325 01326 01330 01331 01332 01333 01334 01335
     01336 01340 01341 01342 01343 01344 01345 01346 01350 01351 01352 01353 01354 01355 01356 01360 01361 01362 01363 01364
     01365 01366 01380 01381 01382 01383 01384 01385 01386 01390 01391 01392 01393 01394 01395 01396 01400 01401 01402 01403
     01404 01405 01406 01480 01481 01482 01483 01484 01485 01486 01500 01501 01502 01503 01504 01505 01506 01510 01511 01512
     01513 01514 01515 01516 01520 01521 01522 01523 01524 01525 01526 01550 01551 01552 01553 01554 01555 01556 01560 01561
     01562 01563 01564 01565 01566 01570 01571 01572 01573 01574 01575 01576 01580 01581 01582 01583 01584 01585 01586 01590
     01591 01592 01593 01594 01595 01596 01600 01601 01602 01603 01604 01605 01606 01610 01611 01612 01613 01614 01615 01616
     01620 01621 01622 01623 01624 01625 01626 01630 01631 01632 01633 01634 01635 01636 01640 01641 01642 01643 01644 01645
     01646 01650 01651 01652 01653 01654 01655 01656 01660 01661 01662 01663 01664 01665 01666 01670 01671 01672 01673 01674
     01675 01676 01690 01691 01692 01693 01694 01695 01696 01700 01701 01702 01703 01704 01705 01706 01710 01711 01712 01713
     01714 01715 01716 01720 01721 01722 01723 01724 01725 01726 01730 01731 01732 01733 01734 01735 01736 01740 01741 01742
     01743 01744 01745 01746 01750 01751 01752 01753 01754 01755 01756 01760 01761 01762 01763 01764 01765 01766 01770 01771
     01772 01773 01774 01775 01776 01780 01781 01782 01783 01784 01785 01786 01790 01791 01792 01793 01794 01795 01796 01800
     01801 01802 01803 01804 01805 01806 01880 01881 01882 01883 01884 01885 01886 01890 01891 01892 01893 01894 01895 01896
     1370 1371 1372 1373 1374 V1201

2    Septicemia (except in labor)
     0031 0202 0223 0362 0380 0381 03810 03811 03812 03819 0382 0383 03840 03841 03842 03843 03844 03849 0388 0389
     0545 449 77181 7907 99591 99592
```

Figure 3.10: Single CCS Diagnosis ICD-9 file

3-digit. Some of the works cited in this document use this method albeit without detailing its implementation, thus we tried to reproduce the methodology based on the available descriptions. With that said, this option crops the ICD-9 codes to its first 3-digits e.g E920.9 → E920 and 480.8 → 480. Similarly to **None** we retrieve the entire ICD-9 vocabulary and apply the 3-digit method to its entirety to compute the 3-digit ICD-9 vocabulary.

2nd hierarcy. Category level or 2nd hierarchy is a method also employed by some cited works but similarly to the previous option, it is also not fully detailed in any of the respective works. Nonetheless, KAME (Ma et al., 2018) references the source[17] which they retrieve the 2nd hierarchy of each ICD-9 code from and exemplify that the 2nd hierarchy level of 250.1 is 249-259. Thus, we used the given source and implemented a web crawler which initially starts in the root level[18] and opens each ICD-9 chapter link, then opening another web-page with the respective 2nd hierarchy levels. We note that it can be the case that the 2nd hierarchy matches its 3-digit version. For each 2nd hierarchy level found, we store it in a pandas dataframe leveraging the digitize method of numpy to retrieve the respective group given the specific ICD-9 code.

ICD Chapters. This option was also implemented but we do not run any experiments with it since it yields only 19 classes, which we argue that is not sufficient granularity to be of any use in a real case setting. That being said, we extract the chapter information from icd9data.com [19] and map the ICD-9 code to the respective range retrieving its chapter.

[17] http://www.icd9data.com/

[18] http://www.icd9data.com/2015/Volume1/default.htm

[19] https://icd.codes/icd9cm

3.6.4 Embedding Initilization

An important step in the methodology used is the embedding initialization procedure. In the described setup, every data point - diagnosis, visit or procedure - is a node. Thus, each node has its own latent representation. Thus, we can leverage pre-trained embedding to initialize the representantions of each node in the graph instead of bootstrapping the architecture embeddings randomly. Intuitively, pre-trained embeddings are expected to yield better results, as it has been shown within the context of NLP with GPT-3 and other language models, where the models are pre-trained and then fine-tuned towards a downstream task. Nevertheless, we still explore the impact of the embedding initalization procedure further ahead.

Each *Diagnosis* and *Procedure* nodes are initialized with the Poincaré algorithm (Nickel & Kiela, 2017), using the implementation offered by the gensim[20] library. In order to apply the Poincaré algorithm, we extracted the medical ontology G as noted previously, and select only the relevant subsets of the ontology before applying the algorithm. We illustrate the process for diagnosis nodes in fig. 3.11. With that said, we use 40 epochs and run the algorithm for different embedding sizes as we do with our proposed architecure. Agarwal et al., 2019 showcases that Poincaré initialization yield better results than other initialization procedures like node2vec for the SNOMED-CT ontology, which is structured hierarchically, similarly to the CCS Multi-level ontology, albeit being more complex and heterogeneous. One of the advantages of Poincaré against node2vec is that the former can be applied directly to prior knowledge (an ontology) without any additional training data, whereas the latter requires the sufficient amount of training data to be provided, where sufficient is normally hard to quantify.

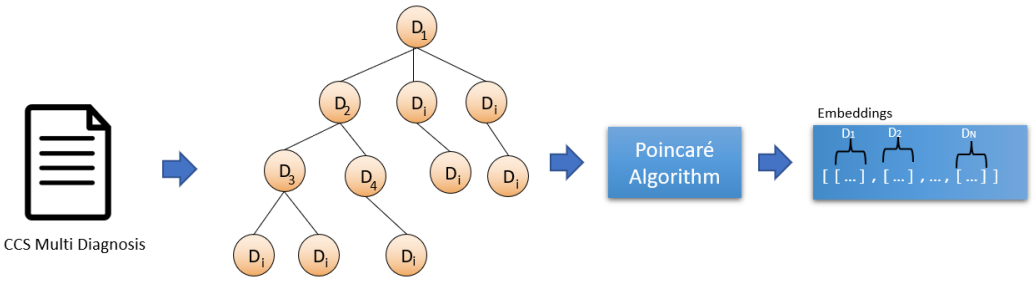

Figure 3.11: Embedding Initialization

The Poincaré algorithm objective is to learn similarities in the embedding space that reflects semantic similarity between two nodes. It re-defines a measure of distance, which projects points in hyperbolic space. This distance can be defined as:

[20] https://radimrehurek.com/gensim/models/poincaré.html

$$d(u,v) = cosh^{-1}(1 + 2\frac{\|u-v\|^2}{(1-\|u\|^2(1-\|v\|^2))}) \tag{3.9}$$

Where u and v are the representations of each point (the points coordinates in a high dimensional space). Thus, the algorithm learns how to position these points such that semantically similar objects are close in the embedding space according to their Poincaré distance. Whereas, the optimization function used for this effect maximizes the distance between unrelated samples.

$$\zeta = \sum_{(u,v)} log\frac{e^{-d(u,v)}}{\sum_{v_1 \in N(u)} e^{-d(u,v_1)}} \tag{3.10}$$

N(u) is the set of negative samples for an entity u.

However, for our specific setting where we initialize both procedures and diagnosis nodes similarly and independently we argue that the current implementation might limit the overall performance of the model. In the worst case scenario the learned embeddings of the different modalities can appear to be similar (nearby in the latent space), without a strong or even any correlation. With that being said, after the embedding initilizations, if a diagnosis is similar - nearby in the latent space - to a procedure, it objectively means nothing.

The visit nodes are initialized with zeroes, since they should hold no information at the start of the training procedure. Nonetheless, it is possible for the visit nodes to learn visit-level features e.g., the current age of the patient, the time delta from the last visit or other visit specific features. However, initial experiments did not show any significant improvement in performance when using the patient age as a feature thus we investigated no further.

We note that all nodes have the same embedding size and are represented in the same latent space. Thus, it follows that different modalities can implictly overlap in hyperdimensional space since they should represent the same concept or set of concepts. However we point to works like Cheerla and Gevaert, 2019, which have explictly attempted to do so by means of unsupervised learning techniques.

3.6.5 Other Implementation Details

In order to perform stochastic gradient descent we needed to implement a custom data-loader to perform data batching, which is the process of sequentially selecting non-repeating subsets of data in each epoch. The basic unit of our data is a patient, which in turn represents a diverse number of data points. Thus, we select subset of patients in each batch, which does not promote equal sized batches since one patient can have two visits whilst other patient ten or even more. At the beginning of each epoch we shuffle patients thus randomizing batches at each epoch. After each batch is selected we create the corresponding graph of patients and feed it into the model.

3.6.6 SOTA Reproducibility

We highlighted that there is a issue of reproducilibity in DL overall and here we discuss some of the discrepancies found in the literature that we used as the building blocks for this work. Namely, several works that tackle the phenotype prediction task do not do so within the same setup and not always state it clearly. Intuitively, an important detail in any ML setup is the number of classes used in the supervised task at hand, which is a detail that is tendencially missing in the studied literature - either the methodology used to compute the labels or even more troublesome, the number of classes effectively used. Thus, objectively comparing the cited works between themselves and our work is rather complicated, even if the employed datasets are the same. For example, Y. Li, Qian, et al., 2020, an architecture that highly inspired this work, showcases a really interesting R@30, however it is not clear on the number of classes used. Similarly, more recently Z. Liu et al., 2020 describes an interesting heterogeneous approach that leverages several modalities of EHR also showcasing competitive results. However, no explicit description is given on the number of classes used for the diagnoses prediction task. In addition, Panigutti, Perotti, and Pedreschi, 2020 which showcases Choi, Bahadori, Schuetz, et al., 2016 results against MIMIC III , uses CCS grouping to reduce the granularity of the prediction task, yielding in 272 labels out of 283 possible CCS Grouping labels. Thus, they do not add the missing labels - and do not give any comment on it - which prevents the model to be transferable to other datasets without re-training. Furthermore, it is unclear if the grouping pertains only to the ICD-9 codes present in the last visit of each patient, or to the whole dataset. Our analysis suggest the former, however we could not reach that value, since our analysis groups MIMIC III labels in 262 CCS groups, when only the last visits are used, or 281 for the whole dataset, but never 272. Thus, in both of these related works we do not have exaclty the same number of labels and thus can not objectively compare the results, even if we are also performing the phenotype prediction task.

Another work, KAME(Ma et al., 2018), states that it groups ICD-9 codes using the second-hierarchy of ICD-9 codes, however similarly to the previous work, they do not disclose if this grouping is done on the whole dataset or only on the last visit, and also do not state the exact number of classes. Nevertheless, we followed the indications and retrieved the second level of ICD-9 codes from the website[21] mentioned in this work which resulted in 184 classes.

Clinical Outcome Prediction from Admission Notes using Self-Supervised Knowledge Integration (van Aken et al., 2021) is a possible candidate to compare our method, albeit being dataset specific as the work stated above. The authors use 1266 diagnoses classes that are computed by generalizing the ICD-9 codes to their first 3 digits, following previous works like Choi, Bahadori, Schuetz, et al., 2016; Choi et al., 2018. However, it is also not clear where the original codes where retrieved from. We could not reach that number. In addition, As a final remark, the only work that explictly details the number

[21] http://www.icd9data.com/2015/Volume1/default.htm

of classes and how they were reached, as far as we know, is Choi, Bahadori, Song, et al., 2017. However, we were not able to replicate the evaluation methodology to objectively compare the results.

With that said, in table 3.3 we summarize both the number of classes used in that specific work, if available, the method used and the number of classes that we computed using ours interpretatino of specified method. We highlight that we always encode the labels of the datasets used with a fixed dimensionality, thus it is to be expected that several works differ for a small number if their encoding does not cover the entire expected vocabulary. We leave for chapter 5 further details of the performance of our framework with different setups and settings, and how to possibly combine the different levels of granularity.

Table 3.3: Label generation |MIMIC III

	Method	Classes	Expected
DoctorXAI[1]	Single CCS Grouping	272	283
KAME	2nd Hierarchy	–	184
GNDP	2nd Hierarchy	171	184
HSGNN[2]	3-digit	203	1778
Clinical Outcome Prediction	3-digit	1266	1778
GRAM	Single CCS Grouping	283	283

[1] uses DoctorAI model
[2] 3-digit is refered in the document but not explicitly stated as the label generation methodology used

3.6.7 Reproducibility

In the github repository[22] we make available the necessary files and recommendations to fully reproduce this work. We can not legally provide either MIMIC III or eICU dataset, so it is the responsibility of the user to download the necessary data.

With that said, here we describe the usage of said files in order to achieve the results stated above. We note that since the splitting between training and testing is random, slight differences between runs may occur. In addition, the smaller the dataset and/or the more diverse, the higher the chance of bigger deviations between runs.

The entire training pipeline can be executed directly from the file **phenotype.py**. In table 3.4 we describe the available parameters and their default values.

Following the parameters described in table 3.4, in order to reproduce the results in table 4.2, table 4.3 and table 4.4 using MIMIC III , we used:

- `python phenotype.py -causal True -task phenotype procedures -add_labels True -k-fold 5 -optimize R@5 R@5 -modalities diagnoses procedures`
- `python phenotype.py -add_labels True -dataset eICU`

[22] https://github.com/Tekaichi/HOPP-code

Parameter	Description	Default
n_layers	Number of layers to train on	[0,1,2,3,4]
embeddings	Embedding Sizes to train on	[50,128,256,512]
batch	Batch Size	256
epochs	Epochs	25
dataset	Dataset	mimic
grouper	ICD9 code grouper for label codes	CCS
replicate	Use target replication whenever possible	False
add_labels	Add missing labels	False
causal	Defines if tasks are sequentially dependent	False
task	Defines tasks to train on	phenotype
override	Override existing results	False
k-fold	Select K for K-fold cross-validation	0
optimize	Hyperparameters to optimize when k-fold > 0	None
modalities	Modalities to use	diagnoses
masking	Masking probability	0
dropout	Use dropout	True
ancestry	Type of ancestry to use	full

Table 3.4: phenotype.py parameters

- `python phenotype.py -add_labels True`
- `python phenotype.py -add_labels True -dataset eICU -task phenotype mortality -causal True`
- `python phenotype.py -add_labels True -task phenotype mortality -causal True`

3.7 Framework

A framework is traditionally defined as a set of software libraries or platforms to aid in developing other software. In order to develop this work we implemented a set of complementary software with the goal of assisting similar implementations. Thus, as a byproduct of our research, we implemented a modular and extensible framework as previously illustrated in fig. 3.1. The implemented framework goal was to facilitate data handling and quickly iterate and experiment with variations of the proposed architecture. Even so, the developed framework can stand as its own contribution. It offers the following features, in no specific order:

- Dataset parser interface
- Networkx based EHR graph constructor
- Instantiate models with any number of GATLayers for any number of tasks with arbitrary number of classes.

- Similarity methods applied to both learned representations and raw data
- Exploratory 'what if' interface

Dataset parser interface. An abstraction that dictates that a dataset parser needs to implement two methods: *to_self_supervised* and *get_label*. The first method is responsible for returning the data in a tabular format with two columns. One column is the aggregation of modalities throughout visits (except the last) - $(diagnoses, procedures) \in Visit$, where diagnoses and procedures are lists of said modalities. The second, are the labels - the diagnosis (and/or procedures) of the last visit. Whereas, *get_label* is responsible for computing other labels like mortality, mortality@30 and mortality@90.

Networkx based EHR graph constructor. A set of classes responsible for transforming the output of the dataset parser interface in a graph using the networkx library. In our work, we implemented two distinct graph constructors - for one and for two modalities of data.

Parameterized model. A pytorch model that receives as parameter the number of GATLayers and a list of integers representing the output size of each task e.g., for the case of phenotype prediction (CCS) and mortality prediction the output size parameter would be $[283, 1]$. The model would consequently be instantiated with N GATLayers and two output layers with the respective output sizes.

Similarity methods. A set of methods used to rank patients according to their similarity. The current implementation of framework offers cosine distance between the learned representations of any visit and Poincaré distance between any visit. Intuitively, Poincaré distance can only used in real data, since the output, in the case for the multi-label tasks of our system is a set of probabilities per patient.

Exploratory 'what if' interface. A set of methods to interact with the model. Namely, add diagnosis and manually boost prediction probabilities. This feature can be seen in System for Visualization and Decision Support based on Health Trajectories(Rebelo, 2021), another Msc. Thesis within the context of Frailcare, as shown in fig. 3.12.

Here given a patient, two tables showcase the top-5 prediction for both diagnosis and procedures using the CCS labeling strategy, given the patient health history until that point. Afterwards, the end-user can highlight diagnoses, which manually increases their score within the model,changing the suggested procedures. Furthermore, the end-user can also add ICD-9 diagnoses and explore their impact on the predictions.

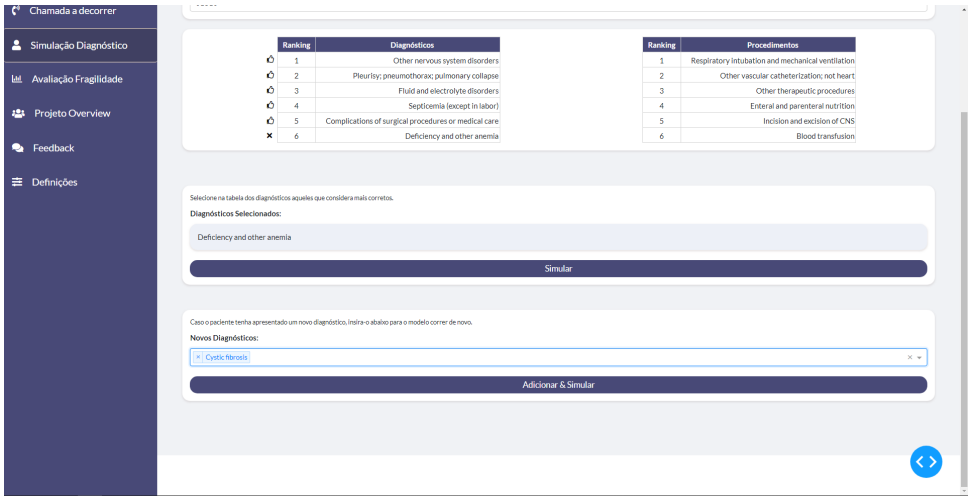

Figure 3.12: 'What if' UI integration

4

Results

4.1 Introduction

Here we present the evaluation of the proposed architecture using the aforementioned hyperparameters for different setups. We train the model with 75% of the data and test with the remaining 25% for both MIMIC III and eICU. We display in table 4.1 the number of patients, admissions (that can be used in training) and diagnosis codes for both datasets after the pre-processing steps described in section 3.6.3. Furthermore, we also compare both the performance and representations learnt with related work.

Table 4.1: Pre-processed dataset statistics

	MIMIC III	eICU
Patients	7.499	15.530
Admissions	12.412	22.182
Diagnoses	164.835	868.879

4.2 Hyperparameter tuning

We follow the standard procedure of machine learning practice of performing k-fold cross-validation to select the best hyperparameters. We use 5 folds and train for 50 epochs using 256 as the batch size. We do not stratify the data, since one of the goals of the proposed architecture is to handle unbalanced and noisy data, which is common within the healthcare domain. With that said, we perform all combinations of the following hyperparameters: *embedding size* (50,128,256,512), *number of GATLayers* (1,2,3,4,5) and the *embedding initialization procedure* (Poincaré, random), yielding in 40 different combinations to be tested. However, due to hardware constraints, it was not possible to run 5 layers with 256 as the embedding size, and 3 or more layers with 512 as the embedding size. Thus, the search was reduced to 32 possible combinations. Furthermore, we performed k-fold cross validation towards the multi-task of pathway prediction - phenotype prediction and procedure prediction -, with CCS Grouping and generalize

the found hyperparameters to the remaining of the tasks. In table 4.2 and table 4.3, we present the top 5 averages of validation errors for each combination of hyperparameters, sorted by R@5 and R@10, for the phenotype prediction task and suggested procedure tasks respectively. We select the top performing combination of hyperparameters - 2 GAT layers, embedding size of 512 with poincaré initialization.

Table 4.2: 5-fold validation diagnoses task | MIMIC 3

Initialization	layers	embeddings	R@5	R@10	R@30
poincare	2	512	**0.648**	**0.577**	0.748
	4	256	0.439	0.417	0.613
	5	256	0.476	0.455	0.660
random	2	512	0.592	0.526	0.718
	4	256	0.619	0.566	**0.760**

Table 4.3: 5-fold validation procedures task | MIMIC 3

Initialization	layers	embeddings	R@5	R@10	R@30
poincare	1	512	**0.575**	0.705	0.915
	2	256	0.441	0.556	0.797
random	1	512	0.549	0.687	0.915
	2	256	0.437	0.538	0.781
	3	256	0.566	**0.711**	**0.924**

4.3 Evaluation

Only MIMIC III dataset has procedures, thus for eICU we showcase only the performance of phenotype prediction and mortality tasks. For the phenotype prediction task, we evaluate the three different labeling strategies, where we make use of recall at different thresholds to validate their performance, showcasing the resulting metrics in table 4.4.

Intuitively, the difference in performance between models that use the entire vocabulary or just the existing subset is minimal. However, the difference appear to widen with the size of the vocabulary - on R@30 is 1.5% while with CCS and 3-digit, is only 0.3%. Nonetheless, we argue that a good metric does not always correlate with a good real world performance of the model. Thus, the label granularity needs to be further evaluated in real world scenarios to study the trade-offs between the different strategies.

In addition, since MIMIC III and eICU present different distribuitions of ICD-9 codes, it is not possible to conclude if the model when trained with more data yields better results, even if intuitively we would argue that is the case. Specifically, models trained and validated with eICU have a higher R@30, however, a significant lower R@5, and that is due the fact that eICU only has $\tilde{4}$ diagnoses per visit whilst MIMIC III has $\tilde{1}1$.

Table 4.4: Test set evaluation | phenotype prediction

	MIMIC III			eICU		
	R@5	R@10	R@30	R@5	R@10	R@30
Phenotype Prediction (CCS)	0.598	0.548	0.716	0.474	0.583	0.823
Phenotype Prediction (CCS)*	0.587	0.541	0.713	0.470	0.575	0.815
Phenotype Prediction (3-digit)*	0.533	0.479	0.607	0.456	0.562	0.793
Phenotype Prediction (3-digit)	0.540	0.477	0.603	0.462	0.572	0.797
Phenotype Prediction (2nd Hierarchy)	0.665	0.637	0.853	0.574	0.706	0.925
Phenotype Prediction (2nd Hierarchy)*	0.669	0.642	0.853	0.580	0.710	0.922
Phenotype Prediction (ICD-9)	0.390	0.333	0.420	0.351	0.430	0.627
Phenotype Prediction (ICD-9)*	0.380	0.324	0.405	0.344	0.423	0.611

* No missing classes where added in these methods.

Table 4.5: Phenotype prediction comparison | MIMIC III

	R@5	R@10	R@30
Phenotype w/ Mortality	0.49	0.462	0.652
Phenotype w/ Mortality@30	0.395	0.368	0.567
Phenotype w/ Procedures	0.586	0.532	0.702
Phenotype	0.598	0.548	0.716

Table 4.6: Mortality prediction comparison | MIMIC III

	sensivity	specificity	accuracy
Mortality w/ Phenotype	0.319	0.872	0.763
Mortality	0.409	0.953	0.884

4.3.1 Multi-task evaluation

As far as we know there are no benchmarks which we can compare our multi-task results. Hence, here our goal is to evaluate how the multi-task performance fares against the single task counterpart i.e., phenotype, procedures and mortality prediction. For the sake of simplicity we will showcase only the results pertaining to MIMIC III , since with eICU we arrive to similar conclusions.

In tables 4.5 and 4.6 we empirically demonstrate that in our setup any combination of tasks performs worse than any of the single tasks individually. We argue that for the multi-task settings a higher number of epochs might be required. We sustain this claim by analysing the graphs in fig. 4.1, where we denote that the model performance keeps improving throughout epochs. However, in fig. 4.2 we can see a reverse trend, where the performance actually worsens throughout epochs even if the loss keeps decreasing. This follows that the treatment task (procedure prediction) is more complex and our architecture not suited for this setting, as we detail further in chapter 5. In addition, if we compare our architecture performance against mortality prediction to the results

displayed in Sadeghi, Banerjee, and Romine, 2018, our architecture under-performs by a wide margin (55%). Nonetheless, we highlight that the context is slightly different since the cited work uses features engineered from vital signs data and its goal is to predict mortality risk within the first hours of ICU admission, whereas our mortality prediction task is not limited in time.

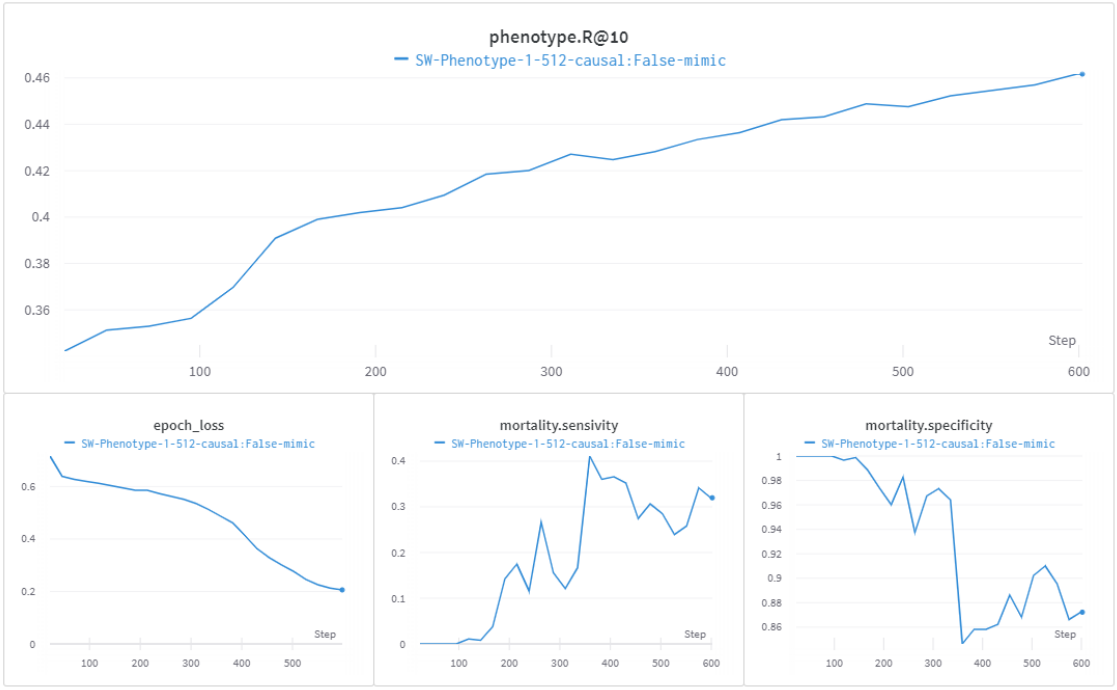

Figure 4.1: Phenotype and Mortality performance | MIMIC III

We also point out that the variance of sensitivity and specificity metrics for the mortality task suggest that the stochastic gradient descent is performing too big of a steps for this task.

4.3.1.1 Few-shot prediction

Few-Shot Learning is a type of machine learning problems where there are only a limited number of examples with supervised information for the given target. In this context, this definition translates to class imbalance where the examples of common diseases outnumber greatly in quantity examples of rare diseases in the data. That being said, in order to evaluate the impact of the medical ontology G on the few-shot performance of the proposed architecture, we follow a similar procedure as GRAM (Choi, Bahadori, Song, et al., 2017). We group each CCS code in one of 5 percentiles (0-20, 20-40, 40-60, 60-80, 80-100), which represent their frequencies in the entire dataset in non-decreasing order, where 0-20 are the rarest diagnoses while 80-100 are the most common ones. With the groups created, we can match the predicted labels with the respective percentile and

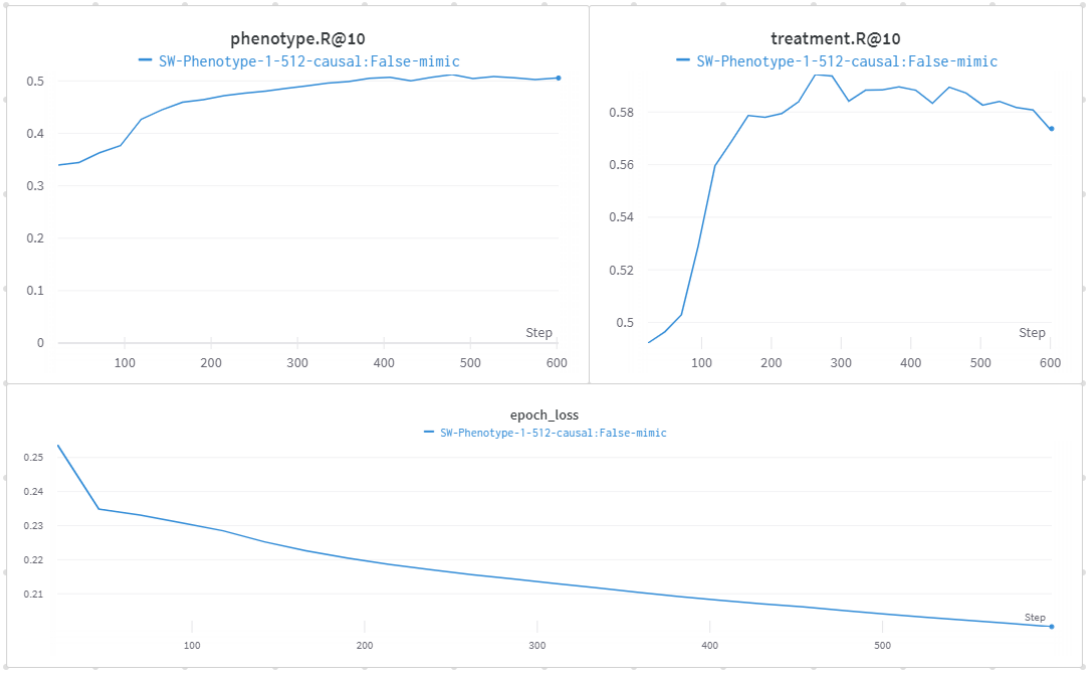

Figure 4.2: Phenotype and Procedure performance | MIMIC III

then compute the corresponding R@20 for each group. We add an index representing the percentile, changing slightly the definition of recall to:

$$Recall@K_i = \frac{|\hat{y}_{i\,correct}|k}{min(k, Y_i)}$$

We display the results in table 4.7, where we sample 2500 patients from MIMIC III and then compare the results against GRAM as displayed in the original paper. GRAM also gathers the predicted labels frequency in 5 groups. However, we do note that the methodology used by GRAM is not detailed, which jointly with the disparity of the results suggest that they do not appear to be directly comparable.

Table 4.7: Few-shot evaluation using R@20| MIMIC III

	0-20	20-40	40-60	60-80	80-100
HOPP	0.52	0.74	0.93	0.98	1.0
GRAM+	0.067	0.179	0.264	0.249	0.627

In table 4.8, we showcase the number of CCS groups in each percentile, where we highlight that in the 80-100 group, there is only one CCS group, which occurs in $\tilde{4}7\%$ of the visits. Thus, it is reasonable to expect that this code is constant throughout predictions.

Table 4.8: Percentile

Percentile	CCS
0-20	241
20-40	30
40-60	6
60-80	5
80-100	1

4.3.1.2 Transfer Learning

It is known that datasets within the same domain can have different distributions and several models that work in one dataset do not work that well on others, even if the goal of a machine learning model is to generalize beyond what it is fed to. With that said, we have shown that MIMIC III differs from eICU in size and distribution, thus in table 4.9 we evaluate how well a model trained on one dataset generalizes to the other for the phenotype prediction task using CCS grouping. That is, we train the model on one dataset and test it on another. We sample 2500 patients from each dataset to evaluate the models, and we use the same models that were trained above. In the columns we represent the dataset which the model - the row - was tested against.

Table 4.9: Transfer Learning | CCS

	mimic			eICU		
	R@5	R@10	R@30	R@5	R@10	R@30
mimic	0.652	0.591	0.769	0.394	0.48	0.693
eICU	0.382	0.36	0.531	0.544	0.658	0.894

4.3.1.3 Clusters

A traditional method to evaluate if a DL architecture is learning properly the latent representations of the data is to project said representantions to 2 dimensions and visualize the result. With that said, we follow this idea and project the embeddings of the specific ICD-9 codes to 2 dimensions using sklearn TSNE[1]. Furthemore, we color each point according to their first level respective to the multi-level CCS ICD-9 ontology - 18 colors in total. In fig. 4.3 we showcase the resulting clusters of ICD-9 codes for diverse methods for comparison, taken from KAME (Ma et al., 2018), and in fig. 4.4, we employ the same methods and showcase the clusters pertaining to our architecture, after training and before training.

Similarly to GRAM and KAME, our method appear to properly cluster points in the relevant groups, however the groups appear to be split. Hence, our method only succeeds a partial representation of medical codes. Furthermore, the actual training seem to

[1] https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

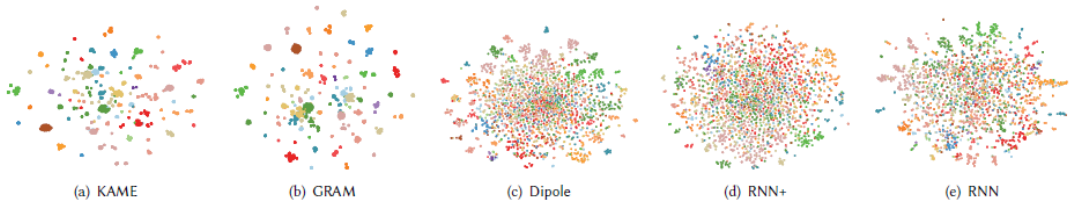

Figure 4.3: Baseline clusters taken from KAME(Ma et al., 2018)

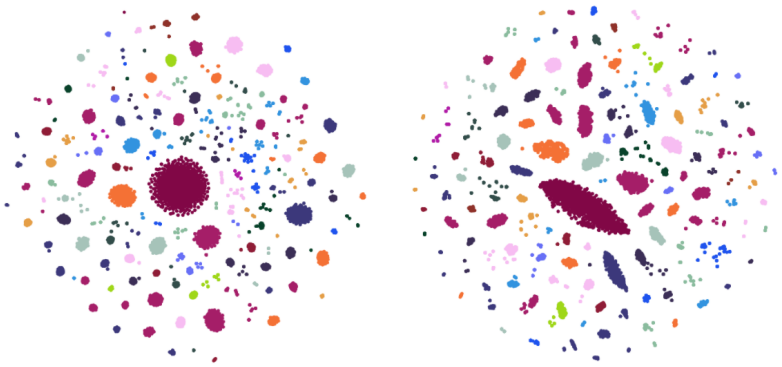

Figure 4.4: HOPP ICD-9 clusters.
left - poincaré only. right - after training

dissipate the clusters when compared to the original Poincaré representations which the ICD-9 code embeddings are initialized with.

4.3.2 SOTA

In table 4.10 and table 4.11, we compare our method for the phenotype prediction task to related ones as objectively as possible. Nonethess, as noted previously, some implementation details are omitted thus we are not able to assert if our experimental settings fully matches the related work. We make use of visit-level precision@k, the same metric as the works we compare. However, in table 4.10, according to the source of the results, the metric is equivalent to Recall@K, as already described above. Whilst in table 4.11 is equivalent to Precision@K, also according to definition from the source of results, with the exception of the last row. In table 4.10, all of the results were taken from the table of results shown in Y. Li, Qian, et al., 2020 while in table 4.11, the first three rows were taken from the table of results in Z. Liu et al., 2020 and the last row from Panigutti et al., 2020. We identify our architecture as HOPP.

In table 4.10, our results are near the current SOTA when K = 30 and K = 10, however, when K = 5 our method seems to fall short with a 10% difference. However, in table 4.11 our method is nowhere near the SOTA. Nonetheless, we would like to point out that if the metrics displayed are in fact equivalent to precision @ k according ot the authors, and

Table 4.10: MIMIC III with 2nd Hierarchy

	Visit-level precision@k		
	5	10	30
Dipole	0.622	0.584	0.802
KAME	0.710	0.657	0.847
GRAM	0.700	0.645	0.842
GNDP	**0.743**	**0.677**	**0.875**
HOPP	0.665	0.637	0.853

Table 4.11: MIMIC III with 3-digit

	Visit-level precision@k		
	5	10	20
Dipole	0.593	0.743	0.754
KAME	0.611	0.748	0.756
HSGNN	**0.642**	**0.766**	**0.774**
Doctor AI	—	0.350	0.521
HOPP	0.519	0.406	0.287

knowing that the dataset used is MIMIC III , which has around 11 diagnoses per visit, consequently when k >11 the results should always be lower than k <= 11, which is not the case in the displayed table.

4.3.3 Weights and biases

Here we leverage the Weight and Biases logs to understand the inner working of our models throughout training (25 epochs). In fig. 4.5, we showcase several runs of the model with the selected hyperparameters using MIMIC III , with CCS Single Grouping (283 classes).

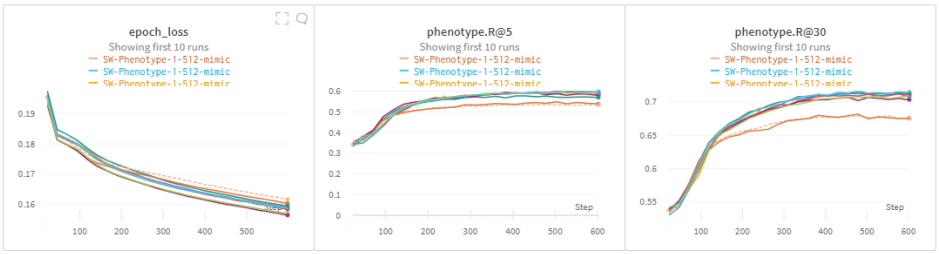

Figure 4.5: W&B loss vs metrics

We can see that the loss keeps decreasing for all of the runs, however, both R@5 and R@30 appear to stabilize around 12 epochs, where R@5 stabilizes sooner around 0.59 whilst R@30 around 0.71. Thus, we conclude that the model at some point stops learning the positive classes, and just ranks lower the negative classes, which does not impact the performance of the model using the proposed metric. Furthermore, we can also see some variation between the models, which correlates to how the dataset split was done since, hence we can safely say that the model is rather sensitive to the data that it was fed, varying as much as 4% in R@30.

In fig. 4.6, we showcase the performance throughout the training of the model with the same hyperparameters as above, but using the 2nd hierarchy method to compute the classes. Here we are interested to see what happens if we train the model through more than 25 epochs, since we have shown that the model appears to stabilize.

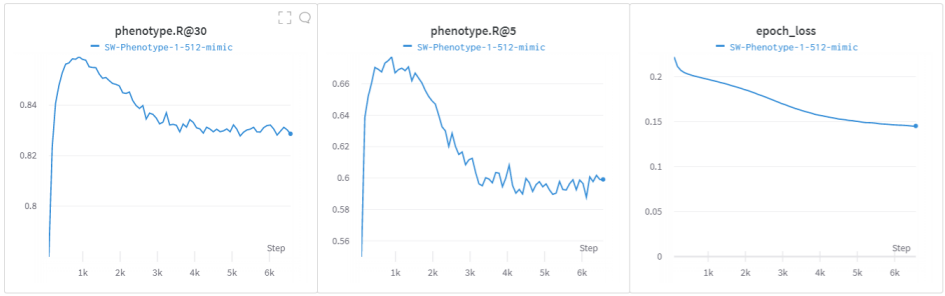

Figure 4.6: W&B extended training

In the showcased run, the model reaches its peak value of R@5 and R@30 in epoch 9, and the more it trains the worse it gets even if the loss of each epoch continues to decrease. This behavior follows the analysis done in fig. 4.5, which leads us to conclude that the model is actually learning what diseases a patient does not have, i.e focusing on the negatives whilst it should focus on the positives.

5

Ablation Studies

5.1 Introduction

In this chapter, we describe all experiments that have been carried out within the scope of this work, evaluating them and discussing the results comprehensively, and whenever possible, comparing them against the results stated in the literature. With that said, we do not describe the data processing pipeline used since most of the experiments share it, nevertheless, when relevant, we highlight the differences in the pre-processing step. In the following section we describe several experiments, showcasing the impact of number of layers, embedding size, how the graph is constructed, target replication, multi-task learning and node masking. Afterwards, we conclude the chapter with an analysis and discussion of the findings. For all the following experiments, if nothing said in contrary, we train for 25 epochs with 256 as the batch size.

5.2 Ablation

Here we empirically demonstrate the results of several concepts that aim to improve the results according to the literature studied within the context of this work. These mild variants of the proposed framework can range between leveraging target replication during training, how the knowledge graph is built to changes to the architecture itself. Here, we evaluate the results against the phenotype prediction task as it is easier to compare among variants and datasets.

5.2.1 Target Replication

Inspired by Lipton, Kale, Elkan, and Wetzel, 2017, we employ a similar strategy of target replication, common within RNNs, for the proposed architecture within the context of phenotype prediction using diagnosis. We experiment with what we define as *paralell* target replication, where we apply the output layer in each and every visit node simultaneously, and compute the loss using the predicted labels and the CCS Grouped codes of the next visit, summing all of them. According to the literature, target replication should

improve the generalization capability of the model since we are essentially augmenting the dataset used during training.

Conceptually, each visit node is now considered as an output node, thus being passed onto the Output Layer as described previously. We implemented target replication by unfolding each patient $P_p = \{x_1^p, x_2^p, ..., x_T^p\}$ into tuples x_t^p, y_{t+1}^p, where y_{t+1}^p are the 283 possible diagnoses at timestep $t+1$. With this approach, the aforementioned architecture and pipeline can be throughly used without any modification, with the addition of being possible to paralelize the output layer across all visit nodes as mentioned. In fig. 5.1 we illustrate target replication process for one patient, and we ommit the ontology related nodes for sake of simplicity.

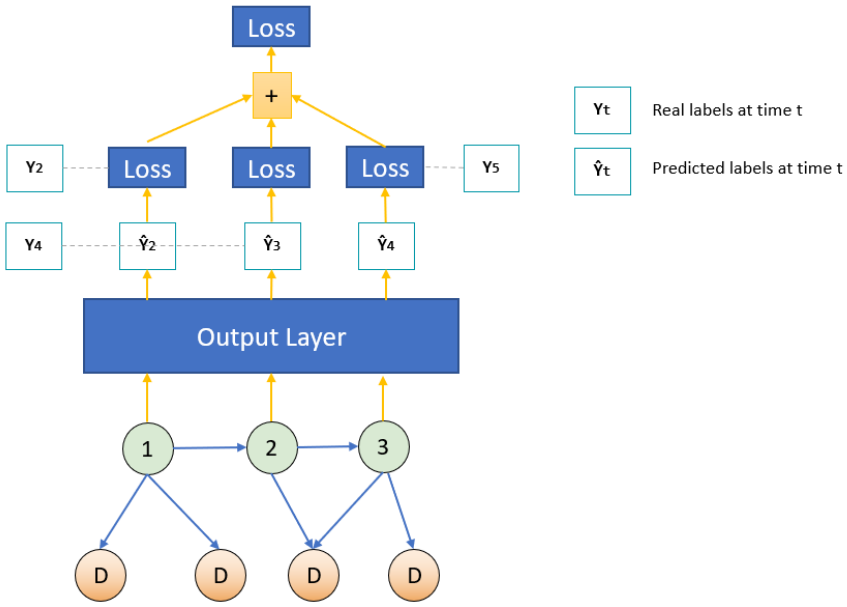

Figure 5.1: Target replication illustration

We note that we weight each timestep with the same value during training, in constrast with some work referenced in Lipton et al., 2017, which weight each target differently depending on when they occur in time - for example the older the less it should weight. We leave this experiment for future work. With that said, we follow a similar setup as described in chapter 3, 25 epochs and 256 batch size and train the model with the strategy of target replication with 75% of the data and validate the results with the remaining 25%. In table 5.1, we showcase the resulting metrics of the test set for MIMIC III for several hyperparameters combinations of the proposed architecture. Intuitively, we can see that the replicated variants take 5% more time training, which scales with the amount of visits each patient in the training set has. We consider the variant 3-256 an outlier, and justify its time complexity due to the shared instance nature of the of the hardware used. We can not identify constant improvements in any of the displayed metrics, thus concluding that the experimented target replication method is not a good option to boost

the performance.

Table 5.1: Target replication strategy vs normal strategy |MIMIC III - 283 classes

		R@5	R@10	R@30	time(s)
1-512	replicate	0.598	0.544	0.715	1712.902
	normal	0.606	0.552	0.717	1620.333
1-256	replicate	0.604	0.539	0.706	1023.782
	normal	0.600	0.539	0.703	951.213
3-256	replicate	0.565	0.518	0.692	1058.954
	normal	0.560	0.518	0.694	1098.759
3-128	replicate	0.553	0.507	0.687	893.153
	normal	0.545	0.503	0.683	682.241

5.2.2 Knowledge Graph

In chapter 3, we briefly showcased that initializing the embeddings of the nodes with the Poincáre algorithm versus a random initialization, in average, yields slightly better results in equivalent models. Nonetheless, the difference is minimal, thus we do not find it significant and we argue that further investigation is needed. With that in mind, here we experiment the impact of the structure of the knowledge graph during training, posterior to the Poincaré initialization. More specifically, we are interested in validating or disproving the claims of cited previous works, (Choi, Bahadori, Song, et al., 2017; Y. Li, Qian, et al., 2020; Ma et al., 2018), which embed a medical knowledge graph in their architectures. Thus, for that effect we employ three experiments where the structure of the knowledge graph differs slightly, namely:

- **Ancestry Partially Connected**, each node is only directly connected to its direct parent;
- **Ancestry Fully Connected**, the knowledge graph topology used in chapter 3;
- **No Ancestry**, the knowledge graph is removed after the embedding are initialized.

We showcase the results for the selected hyperparameters in chapter 2, in table 5.2 for MIMIC III , where the rows are the ancestry type employed and the columns the metrics. The results indicate that a full connected approach, as introduced in Choi, Bahadori, Song, et al., 2017 yields better results across metrics.

Table 5.2: Ancestry evaluation |MIMIC III

	R@5	R@10	R@15	R@30
full	**0.598**	**0.544**	**0.573**	**0.717**
partial	0.571	0.525	0.557	0.708
no-ancestry	0.539	0.493	0.527	0.675

In addition, following the setup in section 4.3.1.1, we also compare the few-shot performance of each topology. In fig. 5.2, we showcase R@20 for each percentile group and for each topology.

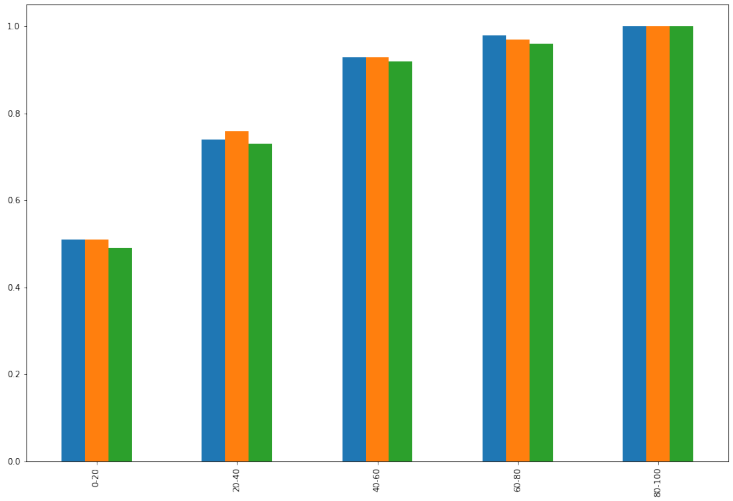

Figure 5.2: blue - full connected, orange - partial connected, green - no ancestry

Intuitively, performance of the model in few-shot scenarios is similar to the results shown in table 5.2. That being said, we reach the same conclusion that the full connected topology yields better results in most cases - only with the exception of the "20-40"percentile.

5.2.3 Multi-task Learning

In chapter 3, we introduced the impact of multi-task learning, namely how learning simultaneously phenotypes and procedures or mortality impacts the performance of the individual tasks. Here, we explore the impact of multi-task learning during training, studying how the performance changes during the 25 epochs which the models is trained on. Intuitively, multi-task learning should yield improved results since its resembles the way humans learn - several tasks at the same time that share characteristics. In this context, learning the next likely diseases share characteristics to the mortality risk, since the predicted diseases can be the cause of mortality. Nonetheless, there is no go-to framework to employ multi-task models, and good results do not tendentially come with simply training a neural network with several tasks simultaneously, as we showcase below in fig. 5.3.

We argue that if the tasks are dependent, the performance of the multi-task models is higher or the same than the individual tasks.

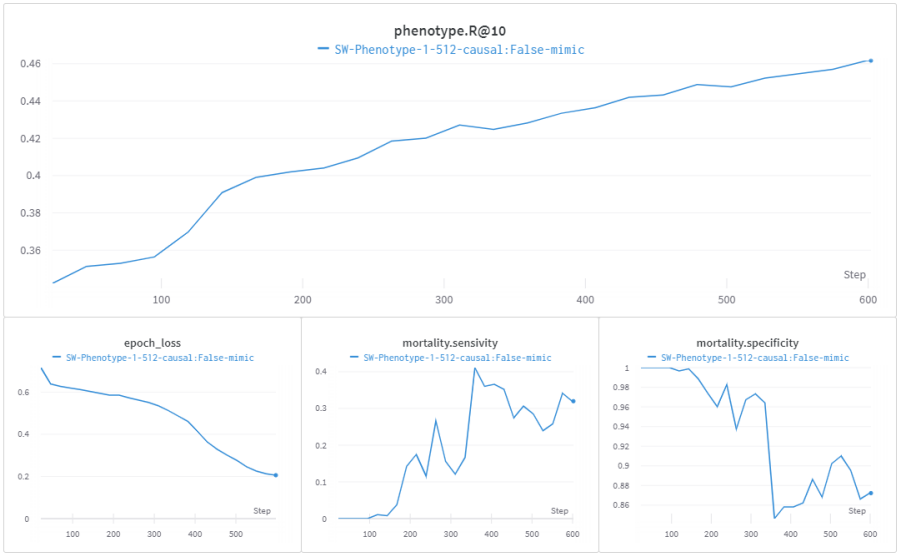

Figure 5.3: W&B Multi-task phenotype and mortality |MIMIC III

5.2.4 Node Masking

Inspired by Devlin et al., 2019; Y. Li, Rao, et al., 2020, 1; Mishra, Piktus, Goossen, and Silvestri, 2021, masking certain elements promotes an improved generalization of the model. Thus, with that goal, we implement a similar masking scheme and evaluate it within the context of this work. The implementation is conceptually similar to Mishra et al., 2021, however we simplify by following the strategy in Devlin et al., 2019 where in this case, the tokens (the nodes), are masked a priori per visit. In practice, we mask the edges that connect a diagnosis to a visit, as illustrated in fig. 5.4. As Devlin et al., 2019 claim, the proposed masking procedure yields better results than the standard dropout. Hence, we experiment using dropout with the same probabilities than the masking method.

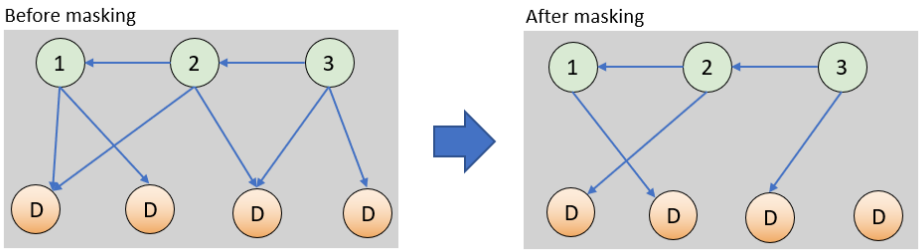

Figure 5.4: Node Masking illustration

We grouped R@5, R@10 and R@30 for both dropout and masking variant for the best combination of hyperparameters defined prior and display them as a bar graph in fig. 5.5. With that said, the X axis represents the metrics, the Y the corresponding value of the

metric, and the colors distinct the dropout variant from the masking one.

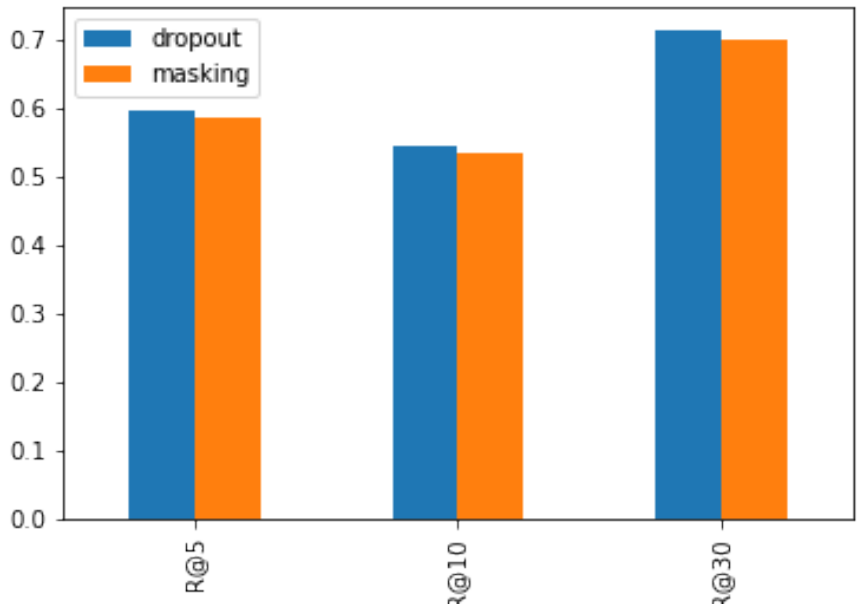

Figure 5.5: Dropout vs Masking

The fig. 5.5 demonstrates that the dropout scheme constantly yields better results for all metrics, thus going against Devlin et al., 2019. However, the masking scheme employed was just an approximation of the one used in the cited work. Furthemore, there can be some variance introduced by the train-test split, thus we can not safely conclude anything from this experiment.

5.2.5 Unidirected

Intuitively, we follow the notion that direction matters, visits are made of diagnoses, a future visit depends on past visits, not the other way around. Nevertheless, intuition might be wrong as architectures like Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) showcase. Hence, we implemented an unidirected version of the proposed architecture in order to benchmark it against the default directed version. That is, given the directed graph $G = (V, E)$, where V are the vertices and E the edges, we simply replace E with E', where E' = $\{\{V_i, V_j\} | (V_i, V_j) \in E\}$. In table 5.3, we display the results where we can see that the unidirected version actually yields slight improvements in several hyperparamenters combinations of the architecture across metrics.

What is more, the developed framework offers both directed and unidirected graph versions, handling the entire necessary transformation pipeline. However, for future research we would also like to attempt a combination of unidirected and directed. For example, the pairs (Diagnoses, Visit) would be unidirected (or directed both ways), whilst

Table 5.3: Unidirected vs Directed

		R@5	R@10	R@30
1-512	unidirected	**0.612**	**0.552**	0.711
	directed	0.598	0.544	**0.715**
1-256	unidirected	0.604	0.547	**0.710**
	directed	**0.604**	0.539	0.706
3-256	unidirected	**0.574**	**0.519**	**0.696**
	directed	0.565	0.518	0.692
3-128	unidirected	**0.567**	**0.516**	0.687
	directed	0.553	0.507	0.687

(Visit,Visit) unidirected, since it seems intuitive. Nonetheless, as shown here, being intuitive is not enough.

5.3 Summary

In this work, we found a plethora of small details that can impact the performance of the proposed model, exploring just a small subset of them here. Namely, the somewhat famous concept of target replication, the presence of the knowledge graph throughout training with different topologies, learning multiple tasks simultaneously and as well the standard method of dropout with a twist. We showcase that some results might be counter-intuitive, as the unidirected experiment shows. In addition, even within DL architectures, the context and topology of the data and some methodologies which improve the performance in some cases might not hold in this specific context, as was the case of employing target replication and leveraging node masking. However, we highlight that here we were able to sustain the claim that inserting knowledge, introducing prior biases, improves the models' performance across settings. Nonetheless, we do note that the difference between most of the variations in the real world amount to nothing, thus the faster option should be chosen. It has been shown throughout DL research that several small details impact the performance the most. That being said, it is impossible to exhaustively experiment all possible combinations of architectural changes, hence further fundamental research is required both within the overarching context of DL and the domain of healthcare.

6

Discussion and Future work

In this work, with the objective of tackling the supervised task of predicting health outcomes, we introduced a novel graph and attention-based deep learning architecture that readily accepts an arbitrary number of modalities - diagnoses, procedures, medications, etc. Where each modality can be initialized with pre-trained embeddings - Poincaré, ClinicalBERT, GloVe, or others. We experiment with random and Poincaré embeddings initialization procedures, and leave the remaining for future work. Furthermore, our model is designed with the goal of being extensible, so that it is easily adapted to a plethora of downstream supervised tasks. We evaluated our architecture against phenotype prediction, procedure prediction and mortality prediction tasks, where we performed several experiments across diverse settings. Namely, we evaluated the learned representations, the performance in few-shot scenarios, transfer learning capabilities and the impact of multi-task scenarios. We validate our results using Recall @ K metric, a metric that resembles how a human doctor would evaluate possible diagnosis. Whereas for the case of the binary mortality prediction task, we measure the performance of positive predictions - how good it predicts the patients that actually died, that is, sensitivity. We compare our methods against previous methods, however possible, showcasing competitive results. Nonetheless, this was proven difficult due to reproducilibity issues present in several works, which we specifically pinpointed, namely the lack of details regarding the number of classes used for the studied tasks. This omission prevents us from accurately concluding how our method fares against SOTA in the several settings it was evaluated against. With that in mind, we open-source our code[1], and display all of our results in order to improve reproducibility and promote more baselines within the context of this research. In addition, we also offer a simple interface to interact with the model with the goal of allowing 'what-if' scenarios, where the user can add or remove diagnoses of any given patient, exploring its impact on the predictions.

[1] https://github.com/Tekaichi/HOPP-code

6.1 Limitations

Due to some of our design choices throughout the development of this work, we can highlight some limitations. The homogeneous nature of our architecture, albeit simple, can be seen as limitation within the context of this work. In fact, the proposed architecture tries to learn representation of different modalities, simultaneously, in the same feature space, which implies that the feature space is big enough to accomodate all modalities. With that said, the larger the feature space, the larger the space and time complexity, which in turn would tendencially require a larger amount of data for the architecture to learn more accurate underlying representations of the data. Whereas, in our work, we argue that the feature space - the embedding size - is not large enough (512) to accomodate different modalities. What is more, training for more epochs, with the goal of learning better representations, do not necessarily improves the performance of the model as shown in previous section. That being said, some of the cited work explicitly tackle this limitation by employing specially crafted heterogeneous graph neural networks (Z. Liu et al., 2020), which requires further research. As a matter of fact, in this work, we attempted to leverage HGNN (Hu et al., 2020) as the fundamental layer in our architecture, however due to memory constraints we were not able to transform a significant amount of data into heterogeneous graphs, which require a feature matrix for each node type and an adjacency matrix for each relationship type. Thus, we could not empirically test our architecture with this setup.

Furthermore, both MIMIC III and eICU are relatively small datasets - 46000 and 140000 patients - when compared to the real world. Hence, any performance or metric shown in this work is not guaranteed to be sustained within real-world settings, where the distribution of the data might be completely different. The goal of any DL model is to generalize to unseen data, but as shown in this document, learning on MIMIC III and using it on eICU, or vice-versa, does not showcase consistent results.

6.2 Conclusion

Our initial goal of developing predictive models for frailty under the umbrella of the project FrailCare.AI was not possible due to constraints out of our control regarding data availability. Nevertheless, we pivoted to a more common task within healthcare - namely phenotype prediction - with the goal of generalizing the initial objective so that our research could be easily translated for its original purpose. Our pivot directed us to uncover reproducibility issues within the literature for the studied task which we detail. Having said that, we still successfully implemented a modular and extensible attention-based graph neural network has originally planned. We argue that the research and work developed can be easily translated to the original goal of FrailCare.AI due to the developed framework built to agilize our experiments. That being said, in order to understand the scalability of our modular architecture, we experimented with two other family of tasks -

mortality prediction and procedure suggestion - where one is binary and other multi-label, kin to the phenotype prediction task. We showcase the results for these tasks and also experiment combinations of all of the studied tasks in a multi-task setting, concluding that in our specific context, multi-task is not desirable performance-wise. However, we direct the readers to (Fifty et al., 2021), which study techniques to optimize multi-task learning by finding optimal combinations of tasks. We argue it could be further studied within the context of this work. Furthermore, we perform ablation studies comparing the developed architecture against itself with different settings - leveraging target replication, changing the graph topology, among others - with the goal of validating our findings against the literature, which showed that our work agrees with most of the claims of the studied literature. That being said, we argue that we still managed competitive results, 72% or 82% for R@30 with CCS grouping, for MIMIC III and eICU respectively, which as far as can ascertain, near SOTA. To conclude, geometric deep learning is a research field increasing in popularity with innumerous aplications, where GNNs are part of. The complex structures that graphs allow to model and the representation learning capabilities of GNN are of the utmost interest and relevance within a context where data is sparse and multi-modal, which is the case of healthcare. In this work, we argue that we successfully studied its application laying out foundations for future work. With that said, healthcare is a multi-area research field where artifical inteligence is only but one of the several branches that needs to be further developed. Namely, without software design, human-computer-intrection and digital literacy, any prediction model is of no use. What is more, specifically for the context of healthcare, solutions need to be designed and developed as close as possible to the real problems, which is as close as possible to the end-users. That is, outside academia, and closer to clinics and hospitals which are the only entities that can actually validate if this or any other developed research within the domain is of any value.

6.3 Future work

We separate our suggestions in two groups: architecture, which relates to improving our proposed DL architecture and system, which covers all the necessary work and research to employ the developed models in the real world.

6.3.1 Architecture

In the researched context, data quality and quantiy was an issue. Hence, intuitively we point to data augmentation research, specifically for the case of imbalanced classes and noisy data. Namely, research the application of SMOTE(Chawla, Bowyer, Hall, & Kegelmeyer, 2002) in this context or extend variational graph auto-encoders(Kipf & Welling, 2016b) - which have generative capabilities - to heterogeneous and complex graphs. Furthermore, the work showcased in Panigutti et al., 2020, where the authors

perform ontological perturbation to generate fake data points to help train their interpretable model. Which, jointly with J. Li, Zhang, Xu, Dickerson, and Ba, 2020, which purposely uses noisy labels, points to interesting directions. Within this data-centric approach, we also suggest the release of more datasets and consequent standardization among them to assist further research.

More research towards scalable heterogeneous graph architectures within this context is also a natural next step. For example, translate concepts in NLP like masked language model and T5 based architectures into the graph domain. Furthermore, the further development of heterogeneous approaches will then allow the full use of SNOMED-CT, a more complex and extensive ontology which correlates a plethora of diverse concepts, in contrast to the one used in this work. What is more, more fundamental researchlike how node classification, where our methods fits in, compares against link prediction and graph classification architectures, will help shed light into the application os GNN within healthcare. We argue that link prediction is more intuitive, and given the change we would follow this direction.

In addition, in this work we decided not to discuss Temporal GNNs(Rossi et al., 2020; Rozemberczki et al., 2021) or Memory Networks (Gao et al., 2019) since our architecture differs from these concepts, however we think that it deservers further research as alternative architectures within this context.

Finally, we are also interested in applying information retrieval techniques like learn2rank to re-order the predicted diagnoses given configurable risk factors defined by domain experts. For example, tune the predictions so that the model tends to predict the more likely diseases that are also more deadly.

6.3.2 System

We argue that the most impactfull research within this context is the one closer to real world applications - improving Clinical Decision Support Systems (CDSSs), improve user experience for all stakeholders (interfaces and explainability) and promote data interoperability. With that in mind, the work done in the receiving institution of this research, VOH.CoLAB, goes in line with this direction. We developed and researched four different components, besides this master thesis, that could easily work together and be integrated in a CDSS, moving towards a more digital, patient centered and cost beneficial healthcare. The research developed for TREC, both 2021 Precision Medicine and TREC 2022 Clinical Trials editions are an important part of an improved precision medicine model, which requires more research. Whereas, EH4C[2], CoaguBot are real world applications that were built side by side with the team of clinicians that is going to use it but also the patients which these tools target. In this context, software design and human-computer interaction are the fundamental pillars to be researched further within healthcare, specifically within hospitals and clinics.

[2]https://easyhealth4covid.vohcolab.org/

Our Health Outcome Pathway Prediction (HOPP) framework ties nicely between the research for TREC and the remaining developed digital health applications. The proposed architecture and consequent framework aim to predict health conditions, whereas both developed digital health software aims to gather and serve data. On the other hand, the research for TREC can be leveraged to augment these models, using biomedical literature, or finding alternative treaments in clinical trials, if the prediction are not satisfactory enough for the domain experts.

6.3.2.1 Data Ingestion

The datasets used are in the traditional tabular format, which loosely map the ER Schema of the healthcare systems which these datasets were originally extracted from. Thus, due to our graph approach, we first needed to implement the necessary tools to transform the data into the expected format, which we argue to be a bottleneck in scaling this solution to real-world scenarios. As a matter of fact, for each dataset used, we had to implement a specific parser as described in chapter 3, which is not scalable if every EHR database is different. However, this can, to some detail, be avoided due to the existence of graph databases like Neo4J and Tiger Graph, which allow the data layer to be modeled in a more flexible fashion that maps directly to any Graph Neural Network architecture. Furthermore, Neo4J offers the graph data science library[3], which can perform several graph algorithms (including graph neural networks) out-of-the-box, without any custom-made implementations - node classification, link prediction or graph classification. Even so, neo4j's graph data science library is still in a alpha stage for machine learning methods, so some work is still needed to fully leverage this or similar technologies.

6.3.2.2 Training procedure

Training graphs can be expensive memory wise, however there have been successes in the field of distributed learning which studies how to train a DL model across servers, distributing the load among them. That beings said, following the graph approach to model the data, it is then rather intuitive to partition the data throughout several servers using the patient as the central unit of partition. Then, given each partition perform the same algorithm simultaneously, sharing the learned information with a centralized server, as NVIDIA showcase in the 2019 blog post "Federated Learning powered by NVIDIA Clara"[4]. Hence, future work would be to adapt our proposed framework to be ready for a distributed or federated learning approach to properly scale with massive datasets to bring the research closer to the real world. What is more, all of the aforementioned suggestions and developed work need to be coupled with proper evaluation methodologies to assess the performance of any deployed models, as well their impact on peoples' lifes.

[3] https://neo4j.com/docs/graph-data-science/current/algorithms/
[4] https://developer.nvidia.com/blog/federated-learning-clara/

Bibliography

Agarwal, K., Eftimov, T., Addanki, R., Choudhury, S., Tamang, S., & Rallo, R. (2019, July 19). Snomed2Vec: Random Walk and Poincar\'e Embeddings of a Clinical Knowledge Base for Healthcare Analytics. arXiv: 1907.08650 [cs, stat]. Retrieved November 27, 2020, from http://arxiv.org/abs/1907.08650. (Cit. on pp. 23, 29, 48)

Biewald, L. (2020). Experiment tracking with weights and biases. Retrieved from https://www.wandb.com/. (Cit. on p. 39)

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... Amodei, D. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, *33*. Retrieved January 28, 2021, from https://proceedings.neurips.cc//paper_files/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html. (Cit. on p. 2)

Cardoso, M. D., & Martins, F. (2020). VOH.CoLAB at TREC 2020 Precision Medicine Track, 9. (Cit. on p. 3).

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. doi:10.1613/jair.953. arXiv: 1106.1813. (Cit. on p. 73)

Cheerla, A., & Gevaert, O. (2019). Deep learning with multimodal representation for pancancer prognosis prediction. *Bioinformatics (Oxford, England)*, *35*(14), i446–i454. doi:10.1093/bioinformatics/btz342. eprint: https://academic.oup.com/bioinformatics/article-pdf/35/14/i446/28913346/btz342.pdf. (Cit. on p. 49)

Cheng, Y., Wang, F., Zhang, P., & Hu, J. (2016, June 30). Risk Prediction with Electronic Health Records: A Deep Learning Approach. In *Proceedings of the 2016 SIAM International Conference on Data Mining* (pp. 432–440). Proceedings of the 2016 SIAM International Conference on Data Mining. doi:10.1137/1.9781611974348.49. (Cit. on p. 24)

Choi, E., Bahadori, M. T., Kulas, J. A., Schuetz, A., Stewart, W. F., & Sun, J. (2017, February 26). RETAIN: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism. arXiv: 1608.05745 [cs]. Retrieved January 18, 2021, from http://arxiv.org/abs/1608.05745. (Cit. on p. 24)

Choi, E., Bahadori, M. T., Schuetz, A., Stewart, W. F., & Sun, J. (2016). Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine learning for healthcare conference* (pp. 301–318). PMLR. (Cit. on pp. 24, 50).

Choi, E., Bahadori, M. T., Searles, E., Coffey, C., & Sun, J. (2016, February 17). Multi-layer Representation Learning for Medical Concepts. arXiv: 1602.05568 [cs]. Retrieved January 6, 2021, from http://arxiv.org/abs/1602.05568. (Cit. on pp. 22, 23)

Choi, E., Bahadori, M. T., Song, L., Stewart, W. F., & Sun, J. (2017, August 4). GRAM: Graph-based Attention Model for Healthcare Representation Learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 787–795). KDD '17: The 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. doi:10.1145/3097983.3098126. (Cit. on pp. 7, 23, 24, 26, 29, 31, 34, 46, 51, 58, 66)

Choi, E., Biswal, S., Malin, B., Duke, J., Stewart, W. F., & Sun, J. (2018, January 11). Generating Multi-label Discrete Patient Records using Generative Adversarial Networks. arXiv: 1703.06490 [cs]. Retrieved September 27, 2021, from http://arxiv.org/abs/1703.06490. (Cit. on p. 50)

Choi, E., Xu, Z., Li, Y., Dusenberry, M., Flores, G., Xue, E., & Dai, A. (2020). Learning the graphical structure of electronic health records with graph convolutional transformer. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 34, pp. 606–613). (Cit. on pp. 27–29, 35).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, May 24). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv: 1810.04805 [cs]. Retrieved January 11, 2021, from http://arxiv.org/abs/1810.04805. (Cit. on pp. 18, 68, 69)

Dong, Y., Chawla, N. V., & Swami, A. (2017, August 4). Metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 135–144). doi:10.1145/3097983.3098036. (Cit. on p. 19)

Esteva, A., Chou, K., Yeung, S., Naik, N., Madani, A., Mottaghi, A., ... Socher, R. (2021). Deep learning-enabled medical computer vision. *NPJ Digital Medicine*, *4*, 5. doi:10.1038/s41746-020-00376-2. pmid: 33420381. (Cit. on p. 5)

Evans, R. S. (2016). Electronic Health Records: Then, Now, and in the Future. *Yearbook of Medical Informatics*, S48–S61. doi:10.15265/IYS-2016-s006. pmid: 27199197. (Cit. on pp. 5, 9)

Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., & Finn, C. (2021, October 25). Efficiently Identifying Task Groupings for Multi-Task Learning. arXiv: 2109.04617 [cs]. Retrieved November 2, 2021, from http://arxiv.org/abs/2109.04617. (Cit. on p. 73)

Franz, L., Shrestha, Y. R., & Paudel, B. (2020, June 23). A Deep Learning Pipeline for Patient Diagnosis Prediction Using Electronic Health Records. arXiv: 2006.16926 [cs]. Retrieved May 20, 2021, from http://arxiv.org/abs/2006.16926. (Cit. on p. 45)

Gao, J., Wang, X., Wang, Y., Yang, Z., Gao, J., Wang, J., … Xie, X. (2019). CAMP: Co-Attention Memory Networks for Diagnosis Prediction in Healthcare. *Proceedings - 19th IEEE International Conference on Data Mining, ICDM 2019*, 1036–1041. doi:10.1109/ICDM.2019.00120. (Cit. on p. 74)

Gobbens, R. J. J., van Assen, M. A. L. M., Luijkx, K. G., Wijnen-Sponselee, M. T., & Schols, J. M. G. A. (2010). The tilburg frailty indicator: Psychometric properties. *J. Am. Med. Dir. Assoc.*, *11*(5), 344–355. doi:10.1016/j.jamda.2009.11.003. (Cit. on p. 3)

Gonçalves, S. (2022). *Uncertainty-aware deep learning for prognosis modelling*. (Cit. on p. 4).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. (Cit. on p. 12).

Gori, M., Monfardini, G., & Scarselli, F. (2005, July). A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* (Vol. 2, 729–734 vol. 2). Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005. doi:10.1109/IJCNN.2005.1555942. (Cit. on p. 18)

Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 855–864). (Cit. on p. 19).

Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., … Webster, D. R. (2016). Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. *JAMA*, *316*(22), 2402–2410. doi:10.1001/jama.2016.17216. (Cit. on p. 1)

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 1025–1035). (Cit. on p. 19).

Harutyunyan, H., Khachatrian, H., Kale, D. C., Ver Steeg, G., & Galstyan, A. (2019). Multitask learning and benchmarking with clinical time series data. *Scientific data*, *6*(1), 1–18. (Cit. on p. 7).

Hu, Z., Dong, Y., Wang, K., & Sun, Y. (2020). Heterogeneous graph transformer. In *Proceedings of the web conference 2020* (pp. 2704–2710). (Cit. on pp. 20, 72).

Johnson, A. E. W., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., … Mark, R. G. (2016). MIMIC-III, a freely accessible critical care database. *Scientific Data*, *3*(1), 160035. doi:10.1038/sdata.2016.35. (Cit. on pp. 10, 42)

Kawamoto, K., Houlihan, C. A., Balas, E. A., & Lobach, D. F. (2005). Improving clinical practice using clinical decision support systems: A systematic review of trials to identify features critical to success. *BMJ*, *330*(7494), 765. doi:10.1136/bmj.38398.500764.8F. (Cit. on p. 11)

Kipf, T. N., & Welling, M. (2016a). Semi-Supervised Classification with Graph Convolutional Networks. Retrieved January 30, 2021, from https://arxiv.org/abs/1609.02907v4. (Cit. on pp. 19, 20)

Kipf, T. N., & Welling, M. (2016b, November 21). Variational Graph Auto-Encoders. arXiv: 1611.07308 [cs, stat]. Retrieved September 4, 2021, from http://arxiv.org/abs/1611.07308. (Cit. on p. 73)

Kwon, B. C., Choi, M.-J., Kim, J. T., Choi, E., Kim, Y. B., Kwon, S., ... Choo, J. (2019). RetainVis: Visual Analytics with Interpretable and Interactive Recurrent Neural Networks on Electronic Medical Records. *IEEE Transactions on Visualization and Computer Graphics*, *25*(1), 299–309. doi:10.1109/TVCG.2018.2865027. arXiv: 1805.10724. (Cit. on pp. 25, 32)

Landi, I., Glicksberg, B. S., Lee, H.-C., Cherng, S., Landi, G., Danieletto, M., ... Miotto, R. (2020). Deep representation learning of electronic health records to unlock patient stratification at scale. *npj Digital Medicine*, *3*(1), 1–11. doi:10.1038/s41746-020-0301-z. (Cit. on pp. 22, 24)

Li, J., Zhang, M., Xu, K., Dickerson, J. P., & Ba, J. (2020, December 23). Noisy Labels Can Induce Good Representations. arXiv: 2012.12896 [cs, stat]. Retrieved September 29, 2021, from http://arxiv.org/abs/2012.12896. (Cit. on p. 74)

Li, Y., Qian, B., Zhang, X., & Liu, H. (2020). Graph Neural Network-Based Diagnosis Prediction. *Big Data*, *8*(5), 379–390. doi:10.1089/big.2020.0070. (Cit. on pp. 7, 23, 27, 29, 30, 32, 36, 40, 50, 61, 66)

Li, Y., Rao, S., Solares, J. R. A., Hassaine, A., Ramakrishnan, R., Canoy, D., ... Salimi-Khorshidi, G. (2020). BEHRT: Transformer for Electronic Health Records. *Scientific Reports*, *10*(1), 7155. doi:10.1038/s41598-020-62922-y. (Cit. on pp. 23, 25, 29, 68)

Lipton, Z. C., Kale, D. C., Elkan, C., & Wetzel, R. (2017, March 21). Learning to Diagnose with LSTM Recurrent Neural Networks. arXiv: 1511.03677 [cs]. Retrieved July 5, 2021, from http://arxiv.org/abs/1511.03677. (Cit. on pp. 64, 65)

Liu, L., Liu, Z., Wu, H., Wang, Z., Shen, J., Song, Y., & Zhang, M. (2020). Multi-task learning via adaptation to similar tasks for mortality prediction of diverse rare diseases. In *Amia annual symposium proceedings* (Vol. 2020, p. 763). American Medical Informatics Association. (Cit. on p. 7).

Liu, Y., Gadepalli, K., Norouzi, M., Dahl, G. E., Kohlberger, T., Boyko, A., ... Stumpe, M. C. (2017, March 7). Detecting Cancer Metastases on Gigapixel Pathology Images. arXiv: 1703.02442 [cs]. Retrieved January 28, 2021, from http://arxiv.org/abs/1703.02442. (Cit. on p. 1)

Liu, Z., Li, X., Peng, H., He, L., & Philip, S. Y. (2020). Heterogeneous similarity graph neural network on electronic health records. In *2020 ieee international conference on big data (big data)* (pp. 1196–1205). IEEE. (Cit. on pp. 27, 50, 61, 72).

Ma, F., Chitta, R., Zhou, J., You, Q., Sun, T., & Gao, J. (2017, August 13). Dipole: Diagnosis Prediction in Healthcare via Attention-based Bidirectional Recurrent Neural Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1903–1911). doi:10.1145/3097983.3098088. (Cit. on p. 25)

Ma, F., You, Q., Xiao, H., Chitta, R., Zhou, J., & Gao, J. (2018, October 17). KAME: Knowledge-based Attention Model for Diagnosis Prediction in Healthcare. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (pp. 743–752). CIKM '18: The 27th ACM International Conference on Information and Knowledge Management. doi:10.1145/3269206.3271701. (Cit. on pp. 7, 24, 26, 29, 31, 47, 50, 60, 61, 66)

Miotto, R., Li, L., Kidd, B. A., & Dudley, J. T. (2016). Deep Patient: An Unsupervised Representation to Predict the Future of Patients from the Electronic Health Records. *Scientific Reports*, *6*(1), 26094. doi:10.1038/srep26094. (Cit. on p. 22)

Mishra, P., Piktus, A., Goossen, G., & Silvestri, F. (2021, May 16). Node Masking: Making Graph Neural Networks Generalize and Scale Better. arXiv: 2001.07524 [cs, stat]. Retrieved June 25, 2021, from http://arxiv.org/abs/2001.07524. (Cit. on p. 68)

Murdoch, T. B., & Detsky, A. S. (2013). The Inevitable Application of Big Data to Health Care. *JAMA*, *309*(13), 1351–1352. doi:10.1001/jama.2013.393. (Cit. on p. 5)

Nickel, M., & Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, *30*, 6338–6347. (Cit. on pp. 23, 48).

Panigutti, C., Perotti, A., & Pedreschi, D. (2020, January 27). Doctor XAI: An ontology-based approach to black-box sequential data classification explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (pp. 629–639). FAT* '20: Conference on Fairness, Accountability, and Transparency. doi:10.1145/3351095.3372855. (Cit. on pp. 50, 61, 73)

Pathak, J., Kho, A. N., & Denny, J. C. (2013). Electronic health records-driven phenotyping: Challenges, recent advances, and perspectives. *Journal of the American Medical Informatics Association : JAMIA*, *20*(e2), e206–e211. doi:10.1136/amiajnl-2013-002428. pmid: 24302669. (Cit. on p. 5)

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). DeepWalk: Online Learning of Social Representations. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710. doi:10.1145/2623330.2623732. arXiv: 1403.6652. (Cit. on p. 19)

Pham, T., Tran, T., Phung, D., & Venkatesh, S. (2017). DeepCare - Predicting healthcare trajectories from medical records: A deep learning approach. *Journal of Biomedical Informatics*, *69*, 218–229. doi:10.1016/j.jbi.2017.04.001. (Cit. on p. 24)

Pollard, T. J., Johnson, A. E. W., Raffa, J. D., Celi, L. A., Mark, R. G., & Badawi, O. (2018). The eICU Collaborative Research Database, a freely available multi-center database for critical care research. *Scientific Data*, *5*(1), 180178. doi:10.1038/sdata.2018.178. (Cit. on pp. 10, 42)

Rebelo, B. (2021). *System for visualization and decision support based on health trajectories*. (Cit. on pp. 4, 53).

Rodrigues-Jr, J. F., Spadon, G., Brandoli, B., & Amer-Yahia, S. (2019, November 28). Patient trajectory prediction in the Mimic-III dataset, challenges and pitfalls. arXiv: 1909.04605 [cs, stat]. Retrieved November 27, 2020, from http://arxiv.org/abs/1909.04605. (Cit. on p. 42)

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., & Bronstein, M. (2020, October 9). Temporal Graph Networks for Deep Learning on Dynamic Graphs. arXiv: 2006.10637 [cs, stat]. Retrieved January 11, 2021, from http://arxiv.org/abs/2006.10637. (Cit. on pp. 20, 74)

Rozemberczki, B., Scherer, P., He, Y., Panagopoulos, G., Riedel, A., Astefanoaei, M., ... Collignon, N., et al. (2021). Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models. In *Proceedings of the 30th acm international conference on information & knowledge management* (pp. 4564–4573). (Cit. on p. 74).

Sadeghi, R., Banerjee, T., & Romine, W. (2018). Early Hospital Mortality Prediction using Vital Signals. *Smart Health (Amsterdam, Netherlands)*, *9–10*, 265–274. doi:10.1016/j.smhl.2018.07.001. pmid: 30873427. (Cit. on p. 58)

Scarselli, F., Gori, M., Ah Chung Tsoi, Hagenbuchner, M., & Monfardini, G. (2009). The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80. doi:10.1109/TNN.2008.2005605. (Cit. on p. 18)

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., ... Silver, D. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, *588*(7839), 604–609. doi:10.1038/s41586-020-03051-4. (Cit. on p. 2)

Schuster, M., & Paliwal, K. (1997). Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, *45*, 2673–2681. doi:10.1109/78.650093. (Cit. on p. 16)

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., ... Hassabis, D. (2020). Improved protein structure prediction using potentials from deep learning. *Nature*, *577*(7792), 706–710. doi:10.1038/s41586-019-1923-7. (Cit. on p. 2)

van Aken, B., Papaioannou, J.-M., Mayrdorfer, M., Budde, K., Gers, F. A., & Löser, A. (2021, February 8). Clinical Outcome Prediction from Admission Notes using Self-Supervised Knowledge Integration. arXiv: 2102.04110 [cs]. Retrieved September 21, 2021, from http://arxiv.org/abs/2102.04110. (Cit. on pp. 26, 50)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008). (Cit. on pp. 16, 17, 36).

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018, February 4). Graph Attention Networks. arXiv: 1710.10903 [cs, stat]. Retrieved December 28, 2020, from http://arxiv.org/abs/1710.10903. (Cit. on pp. 19, 27, 35, 36)

Vuik, S. I., Mayer, E. K., & Darzi, A. (2016). Patient Segmentation Analysis Offers Significant Benefits For Integrated Care And Support. *Health Affairs*, *35*(5), 769–775. doi:10.1377/hlthaff.2015.1311. (Cit. on p. 5)

Wang, L., Wang, Q., Bai, H., Liu, C., Liu, W., Zhang, Y., … Zhou, Y. (2020). EHR2Vec: Representation Learning of Medical Concepts From Temporal Patterns of Clinical Notes Based on Self-Attention Mechanism. *Frontiers in Genetics*, *11*, 630. doi:10.3389/fgene.2020.00630. (Cit. on pp. 22, 23)

Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., & Yu, P. S. (2019). Heterogeneous graph attention network. In *The world wide web conference* (pp. 2022–2032). (Cit. on p. 20).

Wenming, C., Yan, Z., He, Z., & He, Z. (2020). A Comprehensive Survey on Geometric Deep Learning. *IEEE Access*, *PP*, 1–1. doi:10.1109/ACCESS.2020.2975067. (Cit. on p. 19)

Xue, Q.-L. (2011). The frailty syndrome: Definition and natural history. *Clinics in Geriatric Medicine*, *27*(1), 1–15. doi:10.1016/j.cger.2010.08.009. (Cit. on p. 3)

Yan, S., Xiong, Y., & Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. (Cit. on pp. 20, 27).

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., … Sun, M. (2020). Graph neural networks: A review of methods and applications. doi:https://doi.org/10.1016/j.aiopen.2021.01.001. (Cit. on p. 19)

Zhu, W., & Razavian, N. (2019, December 8). Graph Neural Network on Electronic Health Records for Predicting Alzheimer's Disease. arXiv: 1912.03761 [cs, stat]. Retrieved November 27, 2020, from http://arxiv.org/abs/1912.03761. (Cit. on p. 27)