The Pennsylvania State University
The J.Jeffrey and Ann Marie Fox Graduate School

ADVANCING PREDICTIVE MODELING ON ELECTRONIC HEALTH RECORDS: FROM HANDCRAFTED TO AUTOMATED METHODS

A Dissertation in
Informatics
by
Suhan Cui

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2025

The dissertation of Suhan Cui was reviewed and approved by the following:

Dongwon Lee
Professor of College of Information Sciences and Technology
Dissertation Advisor
Chair of Committee

Minhao Cheng
Assistant Professor of College of Information Sciences and Technology

Sharon Huang
David Reese Professor of College of Information Sciences and Technology

Qiushi Chen
Assistant Professor of Industrial and Manufacturing Engineering

Prasenjit Mitra
Affiliated Professor of L3S Research Center, Leibniz University Hannover
Special Member

Carleen Maitland
Professor of College of Information Sciences and Technology
Associate Dean for Research and Graduate Affairs

Abstract

Electronic Health Record (EHR) systems are widely adopted across healthcare institutions, collecting vast amounts of patient data and serving as a foundation for healthcare research. These systems enable exploratory and predictive analytics, facilitating advancements in medical applications such as disease diagnosis, treatment recommendations, and patient monitoring.

In recent years, researchers and clinicians have increasingly leveraged machine learning (ML) techniques to analyze EHR data. However, developing effective ML models for EHR data remains a significant challenge, primarily due to the reliance on human experts to handcraft these models. This process demands expertise in both ML and medical domains and involves labor-intensive efforts to design and optimize model architectures, which often results in models tailored to specific datasets or tasks, limiting their generalizability. Consequently, there is a pressing need for innovative approaches to streamline ML model development for EHR data, minimizing the reliance on domain expertise and manual effort while enhancing model performance across different scenarios.

This dissertation focuses on advancing predictive modeling for EHR data, transitioning from *handcrafted* to *automated* methodologies. In the first part, we propose two handcrafted frameworks, namely `MedPath` and `MedRetriever`, which enhance predictive models using external medical knowledge sources, including knowledge graphs (KG) and medical texts. These frameworks improve model performance and interpretability while reducing reliance on domain-specific expertise. Notably, they can integrate seamlessly with any existing predictive models, offering improved performance across diverse datasets and tasks.

In the second part, we introduce automated frameworks designed to address predictive modeling challenges in multi-modal and multi-task learning for EHR data. These include `AutoMed`, `AutoFM`, and `AutoDP`, which leverage data-driven approaches to automatically design model architectures. By eliminating the need for manual intervention, these methods enhance performance and generalizability across various datasets and tasks. What is more, these methods further reduce the labor efforts to design the model architectures compared to handcrafted methods.

Overall, the methodologies presented in this dissertation—both handcrafted and automated—provide general solutions for EHR modeling, improving model applicability and performance across diverse scenarios.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my deepest gratitude to everyone who has supported me throughout my PhD journey. This research would not have been possible without the guidance, encouragement, and contributions of many individuals.

First and foremost, I extend my heartfelt thanks to my advisors, Prof. Dongwon Lee and Prof. Prasenjit Mitra, for their invaluable insights, patience, and unwavering support. Their expert guidance and constructive feedback have been instrumental in shaping this dissertation. I am especially grateful for their support during the challenging moments of my PhD journey, for which I am deeply thankful.

I am also sincerely grateful to my committee members, Prof. Sharon Huang, Prof. Minhao Cheng, and Prof. Qiushi Chen, for their time, expertise, and thoughtful suggestions, which have greatly enhanced this dissertation.

Additionally, I would like to express my appreciation to my former advisor, Prof. Fenglong Ma, whose mentorship inspired me to pursue a PhD and helped shape my research interests and foundational knowledge.

My heartfelt appreciation goes to my collaborators, whose dedication, insights, and contributions have significantly enriched this work. Their support and expertise have been invaluable in refining key aspects of this study.

Finally, I would like to express my deepest appreciation to my family for their unconditional love, patience, and unwavering belief in me. Their support has been my greatest source of strength.

This dissertation is partially supported by the National Science Foundation (NSF) under Grant No. 1951729, 2119331, 2212323, and 2238275, and the National Institutes of Health (NIH) under Grant No. R01AG077016. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation and the National Institutes of Health.

Chapter 1
Overview

Electronic Health Records (EHRs) are digital systems designed to store comprehensive patient health information, including medical history, diagnoses, medications, test results, and treatment plans. By replacing traditional paper charts, EHRs enhance healthcare efficiency by providing real-time, secure access to accurate and up-to-date data. They facilitate better coordination among healthcare providers, reduce medical errors, and increase patient engagement. In recent years, the widespread adoption of EHR systems has also enabled researchers and clinicians to conduct data-driven healthcare research, with applications spanning disease diagnosis, treatment recommendations, drug discovery, and personalized medicine.

Increasingly, researchers and clinicians are leveraging machine learning (ML) techniques to analyze EHR data, capitalizing on ML's strong representation learning capabilities. Among these applications, predictive modeling refers to forecast future health conditions by analyzing patients' historical EHR data. However, building effective machine learning models for predictive tasks is highly challenging due to the complex nature of EHR data. Specifically, the unique challenges for EHR data are summarized as follows:

- **Temporal Dynamics:** Patient records are longitudinal and irregularly sampled, complicating the capture of time-based trends and sequential events.
- **Heterogeneity of Input Data:** EHR combine structured data (e.g., lab results, medication orders) with unstructured data (e.g., clinical notes), requiring models that can handle diverse modalities.
- **Heterogeneity of Prediction Tasks:** Many EHR applications aim to predict multiple outcomes (e.g., diagnoses, readmission, mortality) simultaneously. This necessitates models that can share representations effectively across tasks while managing potential conflicts.

- **Interpretability:** In the healthcare context, it is crucial for models to be interpretable. Clinicians need to understand the rationale behind model predictions to build trust and ensure that decision-making is transparent and evidence-based.

For addressing the challenges specific to EHR data, most current approaches rely on human experts to design task- or dataset-specific models for EHR-based predictive modeling. These approaches, however, has several significant limitations:

- **C1:** It requires human experts to possess deep expertise in both the medical and machine learning domains—an intersection of skills that is rare. Most of the current predictive models are designed by ML experts, so the common limitation is the lack of medical domain knowledge during modeling EHR data.

- **C2:** It is challenging for human experts to design the appropriate models for EHR data. The development process is labor-intensive, demanding substantial effort to design and optimize model architectures. Also, due to the complex nature of modeling EHR data, it is extremely challenging to find the suitable model architectures, so there is a big room for performance improvement over the current methods.

- **C3:** Most of the current methods is to handcraft the models on a specific feature (i.e. diagnosis codes) or task, which will result in models lacking generalizability to other scenarios. Oftentimes, EHR data may contain diverse types of feature modalities and prediction tasks that makes handcrafting models less effective and efficient.

To address these challenges, we propose a series of methodologies aimed at advancing predictive modeling on EHR data. Our approach focuses on developing more general and automated solutions that reduce reliance on human expertise and labor efforts while improving performance across a variety of scenarios. Specifically, for **C1**, we design frameworks that can incorporate medical knowledge graphs and texts to augment the prediction models, which solves the problem of lacking domain knowledge. Then, for **C2**, we further propose automated machine learning (AutoML) solutions for designing EHR models, that can reduce human efforts and improve performances. Moreover, we extend our AutoML solutions to multi-modal and multi-task scenarios for handling **C3**.

The overall structure of the dissertation is shown in Figure 1.1. We transition from *handcrafted* to *automated* methods for improving prediction models on EHR data. Next, we provide the overview for each Chapter.

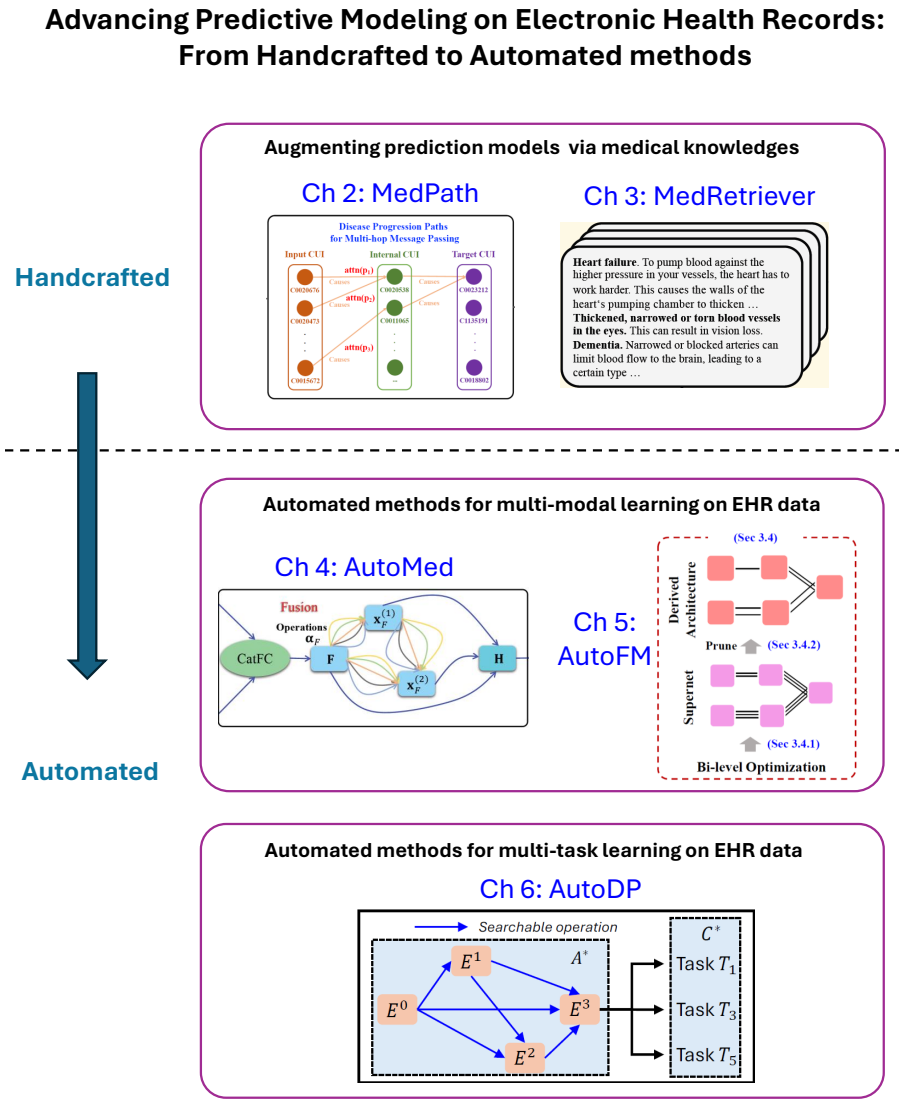


Figure 1.1: Overview of the Dissertation

In Chapter 2 and Chapter 3, we introduce two handcrafted methods—`MedPath` and `MedRetriever`—which utilize external knowledge to address the limitations of insufficient domain knowledge in modeling EHR data. `MedPath` leverages medical knowledge graphs (KG) to improve prediction models, enhancing both their performance and interpretability. In addition to improving predictive accuracy, `MedPath` generates human-readable explanations for prediction results by presenting paths from the KG. Similarly, `MedRetriever` employs medical texts to augment existing prediction models, achieving improved performance. Furthermore, it enhances interpretability by providing natural language explanations for prediction results.

Then, in Chapter 4 and Chapter 5, we move on to the automated methods for multi-modal learning on EHR data. We propose two frameworks - `AutoMed` and `AutoFM`, that leverage neural architecture search (NAS) methods to automatically design the model architectures for multi-modal fusion of EHR data. Specifically in `AutoMed`, we focus on fusing diagnosis information and time information from EHR data, while in `AutoFM`, we propose a more general framework that can handle more EHR modalities. In both works, we develop automated machine learning framework that utilize data-driven methods to design the multi-modal architectures, which large reduce the human intervention for predictive modeling.

Finally, in Chapter 6, we introduce an automated framework for multi-task learning (MTL) on EHR data. We propose `AutoDP`, a framework designed to address the challenges of joint disease prediction using EHR data. This framework not only automates the design of MTL architectures but also performs automatic task grouping for multiple prediction tasks, thereby further enhancing automation and generalizability.

Ethical Concerns. Machine learning (ML) on electronic health record (EHR) data raises significant ethical concerns, particularly regarding patient privacy, data security, and bias. EHR data used to train ML models is highly sensitive, necessitating the implementation of robust anonymization techniques to protect individual identities. Moreover, breaches in data security could expose sensitive health information, resulting in potential misuse or harm. Bias in EHR data, stemming from historical disparities or incomplete records, poses a risk of perpetuating or amplifying inequities in healthcare outcomes when used to train ML models. There is also the danger of over-reliance on algorithms, which may lead to decisions that lack human empathy and contextual understanding. To address these challenges, transparency in model design, equitable representation in datasets, and strict ethical oversight are essential to ensure ML applications in healthcare are both effective and fair.

The datasets used in this dissertation were acquired following established procedures and adhering to all ethical requirements. For example, access to the MIMIC-III [1] and MIMIC-IV [2] datasets was granted only after completing the CITI training [1]. Furthermore, all datasets utilized in this dissertation were properly anonymized and contained no private patient information. The methodologies and models developed in this dissertation enhance the interpretability and reliability of ML models, thereby mitigating concerns related to the "black-box" nature of such systems.

[1] https://about.citiprogram.org/

Chapter 2
Augmenting Health Risk Prediction via Medical Knowledge Paths

2.1 Introduction

The wide adoption of EHR systems has enabled researchers and clinicians to conduct data-driven healthcare research in recent years. It has been used in various applications such as disease diagnosis, treatment recommendations, drug discovery, and personalized medicine. Among those applications, health risk prediction is a classical problem in the medical domain [3]. The goal of it is to predict a patient's future health condition based on the historical EHR of the patient. However, the intrinsic nature of EHR data poses great challenges for the researcher to design effective and accurate predictive models for this task. Since EHR data is often encoded by medical code systems such as International Classification of Diseases (ICD) codes [1], it has features that are high-dimensional, sparse, discrete, temporal and noisy, which largely limits the model's predictive power in terms of both performance and interpretability.

In recent years, many researchers and clinicians start to apply deep learning techniques to solve the challenging task of health risk prediction for its strong representation learning ability. Therefore, various risk prediction models are proposed to help predict the future conditions for patients. The problem of health risk prediction is often formulated as a binary classification task, which means that the goal of prediction is a specific target disease or condition. Based on the historical EHR data of a patient which is typically an ordered sequence of medical visits, the goal is to predict whether this patient will suffer from the target disease or condition in the near future. There are multiple

[1] `https://www.cdc.gov/nchs/icd/icd9.htm`

lines of work that are proposed. Firstly, most existing works aim to apply recurrent neural network(RNN) [4–9] and Transformers [10–12] to modeling the EHR data as sequences and capture the uniques features of EHR. Since the EHR data is sequence data by nature, RNN and Transformer are naturally very suitable model architectures for processing EHR data. Although these methods can achieve remarkable performance, they ignore the importance of incorporating external knowledge to augment health risk prediction. Medical knowledge is widely presented in medical Knowledge Graphs (KG) and medical texts, which could be applied to augment the current EHR-based models for risk prediction. Such valuable information has great potential to improve the current risk prediction models in terms of both performance and interpretability.

To address this issue, researchers have proposed to incorporate prior medical knowledge or knowledge graph (KG) on the web to enhance the representation learning of medical codes, which augments health risk prediction task [13–17]. For example, GRAM [13] learns the latent embedding of a clinical code (e.g., diagnosis code) as a convex combination of the embeddings of the code itself and its ancestors on the ontology graph. KAME [14] is built upon GRAM and tries to use high-level knowledge to further improve the performance. DG-RNN [15] introduces a dynamic attention mechanism to enhance the embedding learning of medical codes with a medical KG named KnowLife [18]. Although these approaches are effective for risk prediction task, they still suffer from the following issues.

Necessity of Incorporating Personalized Knowledge Graph. All the aforementioned approaches need to encode the entire ontology or knowledge graph, which contains a large number of medical codes and corresponding relations. However, the number of overlapping medical codes between individual patients' EHR data and the entire KG is very small. Thus, using the whole KG for individual patient prediction may introduce noise information and further hurt the performance. Moreover, the leading causes of a specific target disease for different patients vary a lot, which indicates the necessity of inferring disease causes for different patients individually. Therefore, it is essential and reasonable to extract personalized knowledge graph for each patient and harness it with deep learning models for achieving personalized prediction.

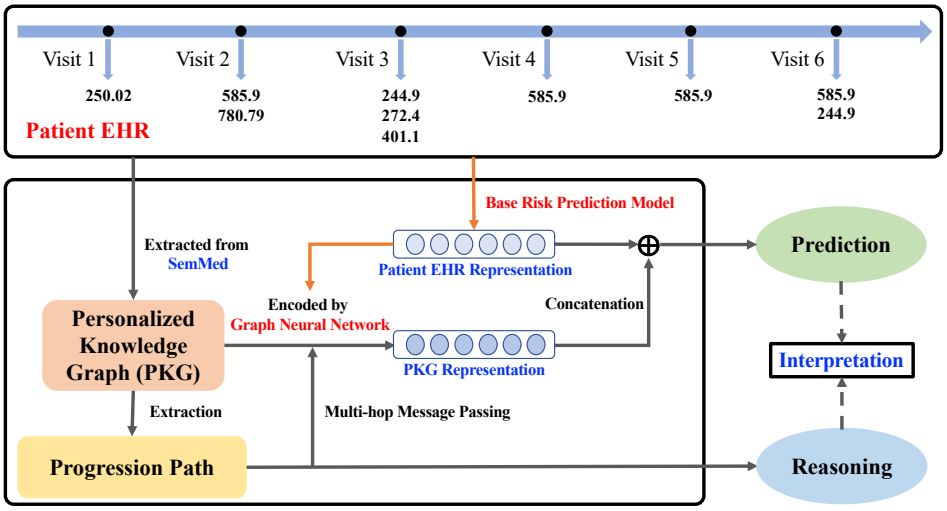
Explicit Reasoning over Disease Progression Paths. Existing approaches aim to incorporate medical KGs to enhance the representation learning of medical codes with the help of relations among them and conduct *implicit reasoning* on the prediction results. However, there exist *explicit disease progression paths in KG* from the observed symptoms (i.e., medical codes) to the target disease, which are ignored by existing studies.

In addition, existing approaches often exploit attention weights to identify important medical codes for the predictions, which can be considered as *implicit one-hop paths*, i.e., from the selected codes to the target disease. However, some one-hop paths may not be in accord with current medical knowledge, which leads to the untrustworthiness of existing models of both patients and doctors. Therefore, explicit reasoning and reliable explanation generation are equally important and challenging for health risk prediction task.

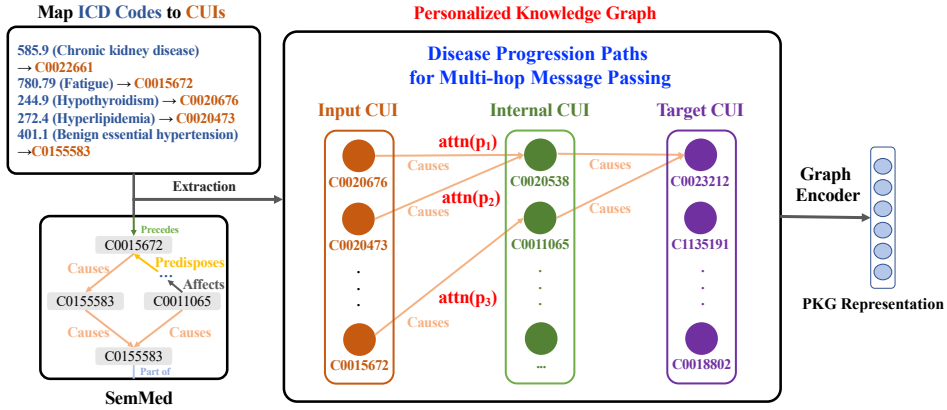
Our Approach. To address the aforementioned challenges, we propose a model-agnostic and ready-to-use framework, named `MedPath` as shown in Figure 2.1(a), which enables any existing risk prediction models to provide personalized prediction and explicit reasoning. During the **prediction** stage, `MedPath` takes both medical codes in a patient's EHR data and the target disease codes as inputs to extract a personalized knowledge graph (PKG) from a large, complex, and noisy medical knowledge graph for each patient. With the guidance of the patient representation obtained from EHR data by existing risk prediction approach, `MedPath` then conducts explicit reasoning via learning personalized knowledge representations over the extracted PKG. Finally, `MedPath` makes predictions based on both patient representations and personalized knowledge representations. During the **reasoning** stage, `MedPath` finds all possible paths within the extracted graph linking the nodes of observed symptoms and the target disease, and uses the learned weights of different paths to explicitly explain the prediction results.

Contributions. Our main contributions are listed as follows:

- We propose a general, model-agnostic and ready-to-use framework `MedPath`, which enables any existing risk prediction models to work with both EHR data and personalized knowledge graphs simultaneously, for providing personalized prediction and explicit reasoning.
- We propose a novel graph attention network to calculate weights for different paths within the extracted personalized knowledge graph, which allows `MedPath` to provide explicit, reliable, and trustworthy medical knowledge path-based explanations for the predictions.
- We conduct experiments on three real-world medical claim datasets to demonstrate the effectiveness of `MedPath` framework in improving the risk prediction performance of eight state-of-the-art risk prediction models. We also use case studies to demonstrate that `MedPath` can provide reasonable explanation for risk prediction.



(a) Framework



(b) PKG Processing

Figure 2.1: (a) The framework of `MedPath`, which augments risk prediction models by learning a PKG embedding extracted from SemMed for prediction. Disease progress can be reasoned from PKG in (b), which is helpful for explicit interpretation.

2.2 Literature Review

2.2.1 Health Risk Prediction

Health risk prediction task is a developing area that attracts significant attention due to its great potential in real-world medical application. Existing risk prediction works mainly focuses on how to better utilize deep learning techniques, which adopt RNNs like gated recurrent units (GRUs) [19], long-short-term memory (LSTM) [20], and self attention-based networks [21]. Built upon the backbone models, some approaches consider

to use other information such as visit time information and extra knowledge as well as advanced attention mechanisms to further improve the prediction performance. Next, we will survey those state-of-the-art risk prediction models.

2.2.1.1 Basic Attention

Built upon the naive recurrent neural network and Tranformer, attention based enhancements are first proposed to improve the performance of medical risk prediction tasks. There are three existing work that are built upon RNN models including Retain [4], Dipole [5] and AdaCare [8]. They all propose different attention mechanisms to further improve the model's representation learning ability. Specifically, Retain [4] is the first interpretable model for risk prediction. It learns visit-level weights and feature-level weights together with two independent RNNs, and then aggregates the features together with the learned weights, which achieves better results than vanilla RNNs. Then, Dipole [5] tries to model longitudinal EHR data using bidirectional Gated Recurrent Unit(GRU) which have a stronger feature extraction ability. In addition, it applies attention mechanisms over the output sequence of bi-directional GRU and aggregate the features of all time steps togehter to make prediction. Finally, AdaCare [8] applies multi-scale convolutions cores to capture the EHR features of different time scales. After the feature extraction process of convolution cores, the output features are further processed by normal RNNs to get the comprehensive representation. The work built on Transformers [21] includes SAnD [10] and LSAN [12]. They combine it with different mechanisms to better capture the patterns of EHR data. SAnD [10]is the first work that apply Transformer to model EHR data and it uses dense interpolation to fuse the embeddings of Transformer outputs, which can better capture the intrinsic patterns of EHR data. Then, LSAN [12] is a recent Transformer based model that incorporates Convolution Neural Network (CNN) and advanced hierarchical attention mechanism that can capture both the long term dependencies and short term correlations of the EHR data.

2.2.1.2 Using Time Information

Besides aforementioned work, some other literature further try to improve the risk prediction model by incorporating time information into the modeling. With the help of additional time information, the model is able to better capture the temporal features and determine the useful signals from the EHR data. Next, we will review the relevant works that use this kinds of approach. Most work focus on adapting RNN models to better handle irregular time intervals, such as T-LSTM [22] and Timeline [7]. T-LSTM [22] is

the first work proposed in this direction, which assumes that the patient information may decay as the time gap increases. They propose a novel model to handle irregular time intervals in longitudinal patient records that can learn a subspace decomposition of the cell memory which enables time decay to discount the memory content according to the elapsed time. And Timeline [7] develops a mechanism that learns time decay factors for every medical code which allows the Timeline to learn that chronic conditions have a longer lasting impact on future visits than acute conditions. By analyzing the attention weights and disease progression functions of Timeline, it is possible to interpret the predictions and understand how risks of future visits change over time. Besides them, RetainEX [6] is an extend work of Retain, which attends the time information into the EHR input and process the combined data in reversed order based on the assumption that more recent visits have more importance during risk prediction. What's more, HiTANet [11] is recent Transformer based model that achieves state-of-the-art performance, which proposes a time-aware attention mechanism built upon Transformer to enable the model to automatically learn the time varying patterns for each patient. The advantage of this work is to allow the model to learn the internal features of time information with little assumptions compared to previous work.

2.2.1.3 Using External Knowledge

On the other hand, some studies focus on the incorporation of the external knowledge on the prediction to improve the interpretation of the model. Beside time information, there are other types of external information that help the model to better make risk prediction results, such as ICD hierarchy structure, medical description, medical knowledge graph, medical text and multi-sourced knowledge. Besides our work that incorporate medical KGs and medical texts, there are several works that apply ICD hierarchy [13, 14], medical code description [17] and multi-sourced data [23]. Specifically, GRAM [13] is an early work that first apply ICD hierarchy to improve the risk prediction model with RNN as the backbone. It utilizes convex combination to obtain the ICD codes' embedding based on the hierarchy of ICD codes family trees, which can augment the risk prediction with the relations among ICD codes. And KAME [14] also apply the ICD hierarchy as the external knowledge. It design a novel knowledge based attention mechanism to better exploits the general knowledge contained in the ICD hierarchy, which can further improve the prediction performance. Then, in [17], the author propose a general framework for diagnosis prediction via incorporating medical code description, which utilize CNN models to extract meaningful embedding from the medical description of the ICD codes

and then the extracted feature can be fed into any predictive model to enhance the representation learning. Unite [23] leverages multi-sourced data (public health data, demographics) to make health risk prediction and provide uncertainty estimations for the prediction results.

2.2.2 Graph Neural Network

Graph neural network (GNN) is a special kind of neural network used for graph data, which learns the features of nodes by aggregating information passed from neighboring ones. The GNN technique is first fueled by the work of Graph Convolutional Network (GCN) by Kipf et al. [24], which builds up the foundation of how to expand the convolution operation on graph data. After that, new types of GNN like Graph Attention Network (GAT) [25] and R-GCN [26] are proposed to propagate neighboring information with graph attention and efficiently handle multi-relational data, respectively.

However, the message-passing process in GCN and R-GCN is single-hop, which cannot handle graphs like the knowledge graph in SemMed to capture more complex relations between nodes. Our work `MedPath` [27] is motivated by a more up-to-date GNN called multi-hop graph relation network (MHGRN) [28] that can provide the multi-hop inference between symptom nodes and target disease nodes. Specifically, the `MedPath` variant called `MedPath`-TA include MHGRN as the graph encoder to learn the embeddings of personalized graphs through multi-hop messages. In addition, we develop on the idea of `MedPath`-TA and provide another variant called `MedPath`-SA to handle multi-relation learning in different hops.

2.3 Data & Task

In this section, we will introduce the format of electronic health records (EHR) and medical knowledge graph data. In addition, we will present how to preprocess those data as the inputs and then formally define our task.

2.3.1 Electronic Health Records

EHR is a special kind of data that is extensively used in health risk prediction task, which contains the complete medical history of a patient and provides rich information on their future status. The EHR $\mathbf{X}$ of a specific patient normally consists of records of multiple visits $[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T]$ where T is the number of visits, and each visit $\mathbf{x}_t$ $(1 \leq t \leq T)$ is

described by several medical (ICD-9[2]) codes where each code represents a symptom, an abnormal finding, or a treatment.

The goal of this work is to predict the probability of a patient suffering from a given target disease in the future by analyzing the historical EHR data $\mathbf{X}$. Though this is a binary classification task, a specific target disease can be denoted by a set of ICD-9 codes due to its various types. Take the heart failure disease as an example. "428.0" represents *congestive heart failure*,[3] "428.1" denotes *left heart failure*,[4] and "428.2" means *systolic heart failure*.[5] For a specific target disease, we let $\mathcal{Y}$ denote the set of the target disease codes. Note that there is no overlapping code between $\mathbf{X}$ and $\mathcal{Y}$.

2.3.2 Medical Knowledge Graph

As mentioned before, our method relies on an external knowledge graph for explicit interpretability. The source of the medical knowledge graph that we use is SemMed [29, 30],[6] which is a huge multi-relational medical knowledge graph with more than 150,000 entities and 64 relation types. The knowledge in SemMed is described in the form of triples, which are extracted from abstract sentences of medical publications on PubMed.[7] Each triple consists of three elements: the **head entity**, the **tail entity**, and the **relation**. The head entity is the subject concept, the tail entity is the object concept, and the relation describes the relationship between the subject concept and the object concept. For example, *<Hypertensive disease, CAUSES, Left heart failure>* is a triple in SemMed. *Hypertensive disease* is the **head entity**, *Left heart failure* is the **tail entity**, and *CAUSES* is the **relation**. Such triples not only augment the risk prediction feature learning but can also be used to explicitly interpret the main reason why the patient will suffer the target disease in the future.

2.3.3 Personalized Graph Extraction

Intuitively, the leading causes of a specific target disease for different patients vary a lot. Thus, it is essential to infer such reasons for different patients individually. To achieve this goal, since we are given a knowledge graph containing giant medical knowledge, we

[2] https://www.cdc.gov/nchs/icd/icd9.htm
[3] http://www.icd9data.com/2015/Volume1/390-459/420-429/428/428.0.htm
[4] http://www.icd9data.com/2015/Volume1/390-459/420-429/428/428.1.htm
[5] http://www.icd9data.com/2015/Volume1/390-459/420-429/428/428.2.htm
[6] https://skr3.nlm.nih.gov/SemMed/
[7] https://pubmed.ncbi.nlm.nih.gov/

can extract a personalized knowledge graph $\mathcal{G}$ from the whole SemMed based on their EHR. The process of obtaining $\mathcal{G}$ with patient EHR data $\mathbf{X}$ and the target disease code set $\mathcal{Y}$ are in the following three steps.

Unification of ICD Codes and SemMed Entities To get the personalized graph, the first problem we need to consider is that SemMed uses Concept Unique Identifiers (CUIs) to represent entities,[8] while EHR data uses ICD codes. To unify them, we first map the ICD codes to CUIs using SNOMED CT [31] for convertion. In particular, all the unique codes in $\mathbf{X}$ are mapped to a new CUI set $\mathcal{E}_x$. Also, the target code set $\mathcal{Y}$ is mapped to another CUI set $\mathcal{E}_y$. Note that an ICD code may have multiple corresponding CUIs. In this case, we map such ICD codes to the first CUIs, which have the highest matching scores. Besides, there are some ICD codes not having corresponding CUIs, we then ignore them.

Path Extraction To construct the personalized graph for reasoning the causes of disease, we need to extract paths between the input CUIs $\mathcal{E}_x$ and the target CUIs $\mathcal{E}_y$. Specifically, given a CUI entity $e_x \in \mathcal{E}_x$ and a target CUI $e_y \in \mathcal{E}_y$, we use depth-first search [32] to generate all possible paths with multiple hops that link e_x and e_y. There are 64 types of relations in SemMed, but only a few of them are related to disease progression. Here, we only keep the paths with the following nine relations in the set of $\mathcal{R}$, namely *AFFECTS*, *AUGMENTS*, *CAUSES*, *DIAGNOSES*, *INTERACTS_WITH*, *PART_OF*, *PRECEDES*, *PREDISPOSES*, and *PRODUCES*. On the generated paths from $\mathcal{E}_x$ to $\mathcal{E}_y$, there are several internal CUIs on the paths, and the set of the internal CUI nodes are represented by $\mathcal{E}_i$.

Personalized Graph Using the outputs from the previous two steps, we can finally obtain the personalized knowledge graph $\mathcal{G} = \{(h, r, t) \mid h, t \in \mathcal{E}, r \in \mathcal{R}\}$ with each triple (h, r, t) describing a relation r between entity h and entity t, where $\mathcal{E} = \{\mathcal{E}_x, \mathcal{E}_i, \mathcal{E}_y\}$, and $\mathcal{R}$ is the set of the selected relations.

2.3.4 Health Risk Prediction Task

The proposed method for the risk prediction task is formulated as follows. For each patient, given the EHR $\mathbf{X}$ and the extracted personalized knowledge graph $\mathcal{G}$, we aim to predict whether the patient would suffer from the target disease in the future. In the meantime, our method also decodes paths from the knowledge graph $\mathcal{G}$ that starts from an input EHR CUI (i.e., $e_x \in \mathcal{E}_x$) and end in a target disease CUI (i.e., $e_y \in \mathcal{E}_y$) to help

[8] `https://www.nlm.nih.gov/research/umls/new_users/online_learning/Meta_005.html`

explain the possible disease development process.

2.4 Methodology

As Figure 2.1 shows, `MedPath` is a flexible framework that can be built on any existing risk prediction model F_e to provide explicit interpretation for the prediction process via the graph neural network F_g. The first part F_e uses state-of-the-art deep learning models as the EHR encoder, which takes the EHR data $\mathbf{X}$ as the input and encode $\mathbf{X}$ into a representation vector $\mathbf{s}$. Then for the graph encoder F_g, which is a multi-relational graph neural network, it takes $\mathbf{s}$ and the extracted personalized knowledge graph $\mathcal{G}$ as inputs to perform multi-relational reasoning. With these modules working together, the proposed `MedPath` framework incorporates external knowledge $\mathcal{G}$ to augment risk prediction representation $\mathbf{s}$ for final prediction. Moreover, we can have explicit interpretability from $\mathcal{G}$ on the relations between symptoms $\mathbf{X}$ and target diseases $\mathcal{Y}$ when we need to explain how `MedPath` makes prediction. The aforementioned process in pseudocode is shown in Algorithm 1.

Algorithm 1: `MedPath` for Health Risk Prediction

Inputs: EHR input $\mathbf{X}$, Personalized Knowledge Graph $\mathcal{G}$
Output: Predicted label y', medical paths $\{p_1, ..., p_{k^*}\}$
Encode EHR into feature $\mathbf{s}$ by Eq. (2.1)
Update node embeddings in in $\mathcal{G}$ by Eq. (2.2), (2.4), (2.5) and (2.6) and obtain $[\mathbf{h}'_1, \cdots, \mathbf{h}'_n]$
Output predicted label y' by following Section 2.4.3 given $\mathbf{s}$ and $[\mathbf{h}'_1, \cdots, \mathbf{h}'_n]$
Output medical paths by following Section 2.4.4

2.4.1 EHR Encoder

As mentioned before, `MedPath` is a general framework which can use any existing risk prediction model as its EHR encoder to represent the data $\mathbf{X}$. A number of existing methods [4–7,11,22,33,34] do not exploit external knowledge for risk prediction. However, they are still good at learning a comprehensive representation of $\mathbf{X}$ since they adopt effective and state-of-the-art deep learning techniques like recurrent neural networks and attention mechanisms. Considering that these representations are good for the training of graph encoder F_g and the final prediction, we simply remove the final classification

layer of any given risk prediction model and use it as the EHR encoder F_e. So we have

$$\mathbf{s} = F_e(\mathbf{X}), \tag{2.1}$$

where $\mathbf{s} \in \mathbb{R}^{d_s}$ is the representation vector learned from EHR encoder, and d_s is the dimension size of $\mathbf{s}$. Feature $\mathbf{s}$ contains the abstract information of EHR extracted by advanced deep learning models, and it can aid the training of graph encoder later on.

2.4.2 Graph Encoder

Using the personalized graph $\mathcal{G}$ extracted for the EHR $\mathbf{X}$ in Section 2.3.3, we introduce the graph encoder module F_g to synthesize the graph knowledge $\mathcal{G}$ into a comprehensive vector $\mathbf{g}$. Suppose that the personalized graph $\mathcal{G}$ has n entities (nodes), the embedding of each entity $\mathbf{h}_j \in \mathbb{R}^{d_h}$ $(1 \leq j \leq n)$ is initialized by pre-training triples in $\mathcal{G}$ with the TranE [35] algorithm. We will conduct type-specific transformation and multi-hop message passing to embed the correlation with other entities to learn a final entity embedding $\mathbf{h}'_j$ for each entity e_j. Eventually, we get the graph representation $\mathbf{g}$ by performing the attentive pooling over all the target disease entity features $\{\mathbf{h}'_j | e_j \in \mathcal{E}_y\}$.

2.4.2.1 Type-Specific Transformation

There are three types of entities in the personalized knowledge graph $\mathcal{G}$: input CUIs $\mathcal{E}_x$, target CUIs $\mathcal{E}_y$, and internal CUIs $\mathcal{E}_i$. To learn a good personalized graph embedding, it is helpful to distinguish the types of different entities and embed the node type information into the node embeddings. Therefore, we apply type-specific transformation on the initialized node embeddings to embed the node type information,

$$\mathbf{v}_j = \mathbf{U}_t \mathbf{h}_j + \mathbf{b}_t, \tag{2.2}$$

where $\mathbf{h}_j \in \mathbb{R}^{d_h}$ is the pretrained node embedding of node j, $\mathbf{v}_j \in \mathbb{R}^{d_x}$ is the transformed feature with type-specific information, and $\mathbf{U}_t \in \mathbb{R}^{d_x \times d_h}$, $\mathbf{b}_t \in \mathbb{R}^{d_x}$. There are different sets of $\{\mathbf{U}_t, \mathbf{b}_t\}$ for input CUIs $\mathcal{E}_x$, target CUIs $\mathcal{E}_y$, and internal CUIs $\mathcal{E}_i$, which are $\{\mathbf{U}_x, \mathbf{b}_x\}$, $\{\mathbf{U}_y, \mathbf{b}_y\}$ and $\{\mathbf{U}_i, \mathbf{b}_i\}$ respectively, and only one of them is used in Eq. (2.2) according the type of $\mathbf{h}_j$.

2.4.2.2 Multi-hop Message Passing

In addition to the node type information, another important information that a node embedding should include is the correlation between the node itself and its neighboring entities. To incorporate this information, the first step is to find out the message passing paths that connect input CUI entities $\mathcal{E}_x$ and target CUI entities $\mathcal{E}_y$. For computational efficiency, we define a hyper-parameter K, which denotes the maximum number of hops in a message passing path from $\mathcal{E}_x$ to $\mathcal{E}_y$. The set of valid K-hop paths is defined as:

$$\begin{aligned} P_k = \{(e_s, r_1, \cdots, r_k, e_d) | (e_s, r_1, e_1), \cdots, \\ (e_{k-1}, r_k, e_d) \in \mathcal{G}\}, (1 \leq k \leq K) \end{aligned} \tag{2.3}$$

where e_s is the source node, e_d is the destination node, and $r_j (1 \leq j \leq k)$ is the j-th relation in the path that connects entities e_{j-1} and e_j. Note that e_s and e_d can be any entities in $\mathcal{E}$, which are not limited to $e_s \in \mathcal{E}_x$ and $e_d \in \mathcal{E}_y$. Thus, the set P_k contains all paths from an arbitrary node in input CUIs $\mathcal{E}_x$ to an arbitrary node in target CUIs $\mathcal{E}_y$ with hops equal to k.

Given the path set P_k, which tracks all possible diseases development progress, we now need to use this information to update the embedding of each node e_j in graph $\mathcal{G}$. For different relation r_l, we use a transformation matrix $\mathbf{W}_{r_l}^t \in \mathbb{R}^{d_x \times d_x}$ $(1 \leq t \leq K)$ to denote how this relation passes the information from source node e_s to e_d. The value of $\mathbf{W}_{r_l}^t$ depends on the distance t from e_s to e_d in any given path $p = (e_s, r_1, \cdots, r_k, e_d) \in P_k$ that passes through e_d. Note that the number of hops k in some path p is smaller than K, we need to introduce padding matrices $W_0^{k+1}, ..., W_0^K \in \mathbb{R}^{d_x \times d_x}$ to maintain the matrix multiplication scale as K for parallel training. Thus, for all paths with k hops in P_k that pass through node e_d, the total updated information they have for e_d is embedded as follows

$$\mathbf{z}_d^k = \sum_{p \in P_k} \text{attn}(p) \cdot W_0^K \cdots W_0^{k+1} W_{r_k}^k \cdots W_{r_1}^1 \mathbf{v}_s, (1 \leq k \leq K), \tag{2.4}$$

where $\mathbf{v}_s \in \mathbb{R}^{d_s}$ is the embedding of source node e_s in path $p = (e_s, r_1, \cdots, r_k, e_d)$. Note that we introduce a structured relational attention mechanism $\text{attn}(\cdot)$, which automatically generates a constant to distinguish the contribution of different path p. The introduction of relational attention mechanism is useful in flexibly selecting important paths of disease development, which is helpful for the prediction interpretation and will be detailed in Section 2.4.2.3.

In Eq. (2.4), we define separate transformations for different positions and different

relations and also pay attention to the lengths of different paths, which allows us to better learn the correlation between node e_j and others. Now we need to aggregate all $\mathbf{z}_j^k$ with different k to get the final embedding $\mathbf{h}'_j$ for each node e_j in $\mathcal{G}$. Remember that we have learned an EHR representation $\mathbf{s}$ from the EHR encoder, so we use $\mathbf{s}$ to guide the aggregation process with a weighted sum calculated by the bilinear attention mechanism,

$$\mathbf{z}_j = \sum_{k=1}^{K} \text{Softmax}(\text{bilinear}(\mathbf{s}, \mathbf{z}_j^k)) \cdot \mathbf{z}_j^k, \tag{2.5}$$

where $\mathbf{z}_j \in \mathbb{R}^{d_x}$. Softmax$(\cdot)$ is used to normalize the attention score corresponding to each $\mathbf{z}_j^k$ calculated by function bilinear$(\cdot)$, and function bilinear$(\mathbf{s}, \mathbf{z}_j^k) = \mathbf{s}^\top \mathbf{B} \mathbf{z}_j^k \in \mathbb{R}$ where $\mathbf{B} \in \mathbb{R}^{d_s \times d_x}$ is a learnable matrix.

Now we have learned a feature $\mathbf{z}_j$ containing the information between the node e_j and the rest entities in the personalized graph $\mathcal{G}$. The final step we need to take is using $\mathbf{z}_j$ to update the node embedding $\mathbf{h}_j$,

$$\mathbf{h}'_j = \sigma(\mathbf{T} \cdot \mathbf{h}_j + \mathbf{T}' \cdot \mathbf{z}_j), \tag{2.6}$$

where $\mathbf{h}'_j \in \mathbb{R}^{d_h}$, $\mathbf{T} \in \mathbb{R}^{d_h \times d_h}$ and $\mathbf{T}' \in \mathbb{R}^{d_h \times d_x}$ are learnable transformation matrix, and σ is an activation function.

2.4.2.3 Structured Relational Attention

Before moving onto how to use $\mathbf{h}'_j$ for the final prediction, we want to discuss the design of **attn$(\cdot)$** function in Eq. (2.4). Recall that function attn$(\cdot)$ is used to calculate the contribution of different paths $p = (e_s, r_1, \cdots, r_k, e_d)$ with the same hop number k. Thus, function attn$(\cdot)$ is useful in finding out important relations between input CUIs $\mathcal{E}_x$ and target CUIs $\mathcal{E}_y$ and interpreting the prediction process.

`MedPath` is flexible in the choice of function attn$(\cdot)$, and we provide two different choices of attn$(\cdot)$ in this paper, i.e. transition matrix-based attention and relational self-attention. These two variants of `MedPath` are referred as `MedPath`-TA and `MedPath`-SA, respectively. The transition matrix-based attention is introduced in [28], which computes the attention weights by a probabilistic graphical model. Although transition matrix-based attention is effective and efficient for distinguishing the importance of different paths, this approach has an assumption that the probability of transition between two relations is fixed. However, in healthcare, different patients usually have different disease development progression, and thus, the transition probability should be dynamically

adjusted. To achieve this goal, we propose a new relational self-attention to dynamically model more complex correlation between different relations in different paths, which further helps improve the information flow of multi-hop message passing in `MedPath`.

Transition Matrix-based Attention. Transition matrix-based attention regards the attention score attn(p) of path $p = (e_s, r_1, \cdots, r_k, e_d)$ as the probability of the path p conditioned on $\mathbf{s}$:

$$\text{attn}(p) = \text{probability}(p|\mathbf{s}), \tag{2.7}$$

which is modeled by a probabilistic graphical model such as conditional random field [36]:

$$\begin{aligned} &\text{probability}(p|\mathbf{s}) \\ &\propto \exp\Big(\mu(\phi(e_s), \mathbf{s}) + \sum_{t=1}^{k} \delta(r_t, \mathbf{s}) + \sum_{t=1}^{k-1} \tau(r_t, r_{t+1}) + \nu(\phi(e_d), \mathbf{s})\Big) \\ &\triangleq \underbrace{\beta(r_1, \cdots, r_k, e_d)}_{\text{Relation Type Attention}} \cdot \underbrace{\gamma(\phi(e_s), \phi(e_d), \mathbf{s})}_{\text{Node Type Attention}}, \end{aligned} \tag{2.8}$$

where function $\phi(\cdot)$ outputs the node type of the input node. In implementation, functions $\mu(\cdot)$, $\nu(\cdot)$ and $\delta(\cdot)$ are learned by two-layer multilayer perceptrons (MLPs) and $\tau(\cdot)$ by a transition matrix $\in \mathbb{R}^{m \times m}$, where m is the number of relations.

Relational Self-Attention. From Eq. (2.8), we can see that the probability is defined as the product of relation type attention $\beta(\cdot)$ and node type attention $\gamma(\cdot)$. Function $\beta(\cdot)$ models the importance of a k-hop relation, while function $\gamma(\cdot)$ models the importance of messages from source CUIs to destination CUIs based on $\mathbf{s}$. Motivated by this idea, we propose an improved attn($\cdot$) by incorporating self-attention mechanism [21].

For modeling the differences among different patients, instead of using a fixed relation transition matrix $\tau(\cdot)$ as Eq. (2.8) does, we consider dynamically generating an $m \times k$ score matrix for every relation type at each hop conditioned on $\mathbf{s}$. First, the model takes the EHR representation $\mathbf{s}$ as the input and performs hop-specific transformation for the j-th hop by $\mathbf{a}_j = \mathbf{M}_j\mathbf{s}$, where $\mathbf{M}_j \in \mathbb{R}^{m \times d_s}$ and $\mathbf{a}_j \in \mathbb{R}^m$. We pack all $\mathbf{a}_j$ together and have a matrix $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_k] \in \mathbb{R}^{m \times k}$. We use the self-attention to capture relation-type correlations among different hops within a path,

$$\text{SelfAttention}(\mathbf{A}) = \text{Softmax}(\frac{\mathbf{A}_q\mathbf{A}_k^\top}{\sqrt{d}})\mathbf{A}_v, \tag{2.9}$$

where matrices $\mathbf{A}_q = \mathbf{M}_q\mathbf{A}$, $\mathbf{A}_k = \mathbf{M}_k\mathbf{A}$, and $\mathbf{A}_v = \mathbf{M}_v\mathbf{A}$ are the query, key and value matrices transformed from $\mathbf{A}$ respectively, where $\mathbf{M}_q$, $\mathbf{M}_k$, $\mathbf{M}_v \in \mathbb{R}^{d \times m}$, and d is the

number of transformed features.

The output of SelfAttention($\mathbf{A}$) is then mapped back to the original space $\mathbb{R}^m$ by a linear transformation $\mathbf{M}_l \in \mathbb{R}^{m \times d}$ and passed to softmax activation function to generate attention scores for all possible k-hop relations:

$$\beta(r_1, \cdots, r_k, \mathbf{s}) = \text{Softmax}(\mathbf{M}_l \cdot \text{SelfAttention}(\mathbf{A})). \tag{2.10}$$

For node type attention $\gamma(\phi(e_s), \phi(e_d), \mathbf{s})$, functions $\gamma(\cdot)$ and $\phi(\cdot)$ are both modeled by two-layer MLPs, and node type attention function $\phi(\cdot)$ is different for the source node and the destination node. Given $\beta(r_1, \cdots, r_k, \mathbf{s})$ and $\gamma(\phi(e_s), \phi(e_d), \mathbf{s})$, we can multiply them together and finally have the attention score attn(p) by taking the normalization of the multiplication score.

2.4.3 Prediction

Following previous sections, we now have the comprehensive EHR representation $\mathbf{s}$ by EHR encoder F_e and the updated personalized knowledge graph with node embeddings $[\mathbf{h}'_1, \cdots, \mathbf{h}'_n]$ obtained by multi-hop message passing. For the final prediction, we firstly apply attentive pooling [37] over all the target CUI entity features to obtain graph embeddings $\mathbf{g}$. Then we concatenate $\mathbf{g}$ and $\mathbf{s}$ to compute the final output by FC($\mathbf{s} \oplus \mathbf{g}$), where FC is the fully connected layer for classification, and $\oplus$ is the concatenation operation. The whole model is trained jointly end-to-end by minimizing the cross-entropy loss between FC($\mathbf{s} \oplus \mathbf{g}$) and the ground truth label.

2.4.4 Interpretation

One of the benefits of the proposed `MedPath` is that we can decode all k-hop paths ($1 \leq k \leq K$) and select the path with high attention scores during feed-forward process for model interpretation. From the attentive pooling layer, we can identify the target CUI entity e_y with the highest attention score. Then, using Eq. (2.5), we can get the optimal k^* with the highest attention scores. Among all k^*-hop paths, the most possible way that the patient will get the target disease is the path p with the maximum attn(p).

2.5 Experiments

In this section, we will first provide the details of our experimental settings and then discuss the results of comparison experiments and ablation studies. Finally, case studies are included to demonstrate how `MedPath` interprets the prediction results from the attention weights and personalized knowledge graph.

2.5.1 Experimental Setup

2.5.1.1 Datasets

Our experiments are conducted on three EHR datasets collected from real-world claim data. The statistics of these datasets are shown in Table 2.1. The target disease in these datasets is heart failure, chronic obstructive pulmonary disease (COPD), and kidney disease, respectively. Patients of these diseases normally experience a chronic and progressive condition for a long period.

2.5.1.2 Baselines

We consider the following baseline models, including LSTM [20], Dipole [5], [4], SAnD [38], RetainEx [6], Timeline [22], LSAN [12], and HiTANet [11], which also can be used as base models of the proposed `MedPath`-TA and `MedPath`-SA approaches.

2.5.1.3 Implementation

`MedPath` is implemented by PyTorch framework on an NVIDIA Tesla P100 GPU and Intel Xeon E5-2680 CPUs. The parameters are trained by Adam optimizer with the learning rate of 10^{-4} and the mini-batch size is 64. The hidden state numbers in LSTM

Table 2.1: Statistics of the used datasets.

Dataset	Heart Failure	COPD	Kidney Disease
Positive Cases	3,080	7,314	7,810
Negative Cases	9,240	21,942	8,430
Average Visits per Patient	30.39	38.74	39.09
Average Codes per Visit	4.24	3.50	4.40
Unique ICD-9 Codes	8,692	10,053	8,802

Table 2.2: Performance Comparison (with the p-values of significance test) in terms of AUC. The average AUC scores of our `MedPath` variants `MedPath`-TA and `MedPath`-SA for each dataset are followed by the percentage improvement (↑) over Vanilla models.

Dataset	Heart Failure			COPD			Kidney Disease		
Method	Vanilla	MedPath-TA	MedPath-SA	Vanilla	MedPath-TA	MedPath-SA	Vanilla	MedPath-TA	MedPath-SA
LSTM	0.708	0.716 (1e-10)	**0.739** (6e-10)	0.693	0.703 (4e-9)	**0.707** (7e-9)	0.739	0.762 (4e-10)	**0.774** (1e-10)
Dipole	0.687	0.744 (2e-8)	**0.751** (2e-8)	0.704	0.714 (2e-10)	**0.728** (1e-10)	0.755	0.765 (3e-7)	**0.768** (2e-7)
Retain	0.689	0.733 (2e-8)	**0.735** (5e-8)	0.699	0.723 (6e-10)	**0.730** (6e-10)	0.732	**0.766** (1e-7)	0.764 (3e-7)
SAnD	0.686	0.733 (1e-7)	**0.745** (1e-7)	0.692	0.736 (7e-10)	**0.737** (9e-11)	0.748	0.769 (2e-7)	**0.790** (5e-8)
RetainEx	0.688	0.738 (6e-9)	**0.751** (2e-9)	0.707	**0.746** (2e-9)	0.743 (2e-9)	0.728	0.772 (2e-8)	**0.786** (3e-9)
Timeline	0.705	**0.735** (3e-9)	0.729 (2e-8)	0.698	**0.713** (4e-9)	0.704 (1e-9)	0.756	0.761 (6e-9)	**0.769** (7e-9)
LSAN	0.738	0.729 (9e-8)	**0.745** (1e-7)	0.723	**0.728** (4e-6)	0.720 (2e-6)	0.766	0.765 (9e-7)	**0.782** (5e-8)
HiTANet	0.750	**0.785** (4e-8)	**0.785** (3e-8)	0.752	0.787 (7e-11)	**0.799** (1e-10)	0.792	0.800 (8e-8)	**0.810** (4e-7)
Average	0.706	0.739 (↑4.7%)	**0.748 (↑5.9%)**	0.709	0.731 (↑3.1%)	**0.734 (↑3.5%)**	0.752	0.770 (↑2.4%)	**0.780 (↑3.7%)**

Table 2.3: Performance Comparison (with the p-values of significance test) in terms of F1 Score. The average F1 scores of our `MedPath` variants `MedPath`-TA and `MedPath`-SA for each dataset are followed by the percentage improvement (↑) over Vanilla models.

Dataset	Heart Failure			COPD			Kidney Disease		
Method	Vanilla	MedPath-TA	MedPath-SA	Vanilla	MedPath-TA	MedPath-SA	Vanilla	MedPath-TA	MedPath-SA
LSTM	0.561	0.582 (2e-10)	**0.611** (1e-9)	0.548	**0.554** (7e-9)	0.550 (1e-8)	0.616	0.646 (3e-9)	**0.661** (3e-9)
Dipole	0.542	0.625 (6e-8)	**0.631** (8e-8)	0.562	0.560 (1e-9)	**0.584** (5e-10)	0.656	**0.657** (1e-6)	0.645 (1e-6)
Retain	0.549	**0.613** (4e-8)	0.612 (1e-7)	0.555	0.570 (2e-9)	**0.575** (1e-9)	0.614	**0.654** (4e-7)	**0.654** (6e-7)
SAnD	0.544	0.605 (2e-7)	**0.609** (3e-7)	0.539	0.568 (1e-9)	**0.590** (3e-10)	0.636	0.633 (9e-7)	**0.670** (3e-7)
RetainEx	0.546	0.625 (4e-9)	**0.640** (2e-9)	0.570	**0.613** (1e-9)	0.612 (1e-9)	0.612	0.665 (4e-8)	**0.682** (2e-8)
Timeline	0.574	**0.614** (5e-9)	0.609 (1e-8)	0.550	**0.570** (6e-9)	0.555 (2e-9)	0.648	0.647 (1e-8)	**0.657** (3e-9)
LSAN	0.623	0.607 (2e-7)	**0.626** (1e-7)	0.574	**0.586** (1e-5)	0.573 (4e-6)	0.661	0.651 (4e-6)	**0.668** (1e-6)
HiTANet	0.647	**0.678** (1e-7)	0.668 (1e-7)	0.637	**0.670** (8e-10)	**0.670** (4e-10)	**0.702**	0.687 (5e-7)	0.688 (8e-7)
Average	0.574	0.619 (↑7.8%)	**0.626 (↑9.1%)**	0.567	0.586 (↑3.4%)	**0.589 (↑3.9%)**	0.643	0.655 (↑1.9%)	**0.666 (↑3.6%)**

and GRU are both 256, while they are 128 in the rest methods. As for the graph encoder, the embedding size of each CUI code is 100 and the layer number is 1.

2.5.1.4 Evaluation Metrics

The evaluation metrics in our task include F1 score and area under the receiver operating characteristic curve (AUC). AUC is the probability that a model ranks a randomly chosen positive case higher than a randomly chosen negative case. As for F1 score, it is the harmonic mean of precision and recall. Precision penalizes false positives while recall penalizes false negatives. Thus, F1 score is a better evaluation for it takes these two aspects into consideration. The higher these scores are, the better performance risk prediction frameworks have.

2.5.2 Experimental Results

2.5.2.1 Performance Comparison

The comparison results are shown in Tables 2.2 and Table 2.3. For each baseline method, we first test the vanilla model and get the AUC and F1 scores. Then we build on the vanilla model and add a graph neural network layer to obtain the performance of `MedPath`-TA and `MedPath`-SA using either $k = 2$ or 3, where k is the length of knowledge path.

Firstly, we can observe the effectiveness of using personalized knowledge by comparing the AUC and F1 scores between the vanilla models and the `MedPath` variants. Take `MedPath`-TA as an example. By comparing the results of vanilla models and `MedPath`-TA from Table 2.2, we can see that in almost all the settings, `MedPath`-TA achieves higher AUCs. While from Table 2.3 in 75% settings, `MedPath`-TA has higher F1 scores. Notably, the largest improvement in AUC takes place in Dipole, SAnD and RetainEX on the heart failure, COPD and kidney disease datasets, respectively. The largest improvement in F1 scores is taken place in Dipole, RetainEx and RetainEx for heart failure, COPD and kidney disease, respectively. Existing base models emphasize interpreting prediction through attention weights, so `MedPath` is especially good at coping with existing interpretable frameworks. We also take the average of AUC and F1 score over all base methods for different datasets. The average AUC increment that `MedPath`-TA has over the vanilla model is 0.033, 0.022 and 0.018 in heart failure, COPD and kidney disease, respectively. The average F1 score increment is 0.045, 0.019 and 0.012 in heart failure, COPD and kidney disease, respectively.

After determining the effectiveness of graph encoder, the next step is finding out how a better structural relational attention design in graph neural network layer can improve the performance of `MedPath`. In Section 2.4.2.3, we have discussed two types of structural relational attentions, i.e. transition matrix-based attention (TA) and relational self-attention (SA), and we distinguish these two `MedPath` varients by `MedPath`-TA and `MedPath`-SA, respectively. Specifically speaking, `MedPath`-SA can achieve higher AUCs than `MedPath`-TA in 79.2% settings in Table 2.2, and it can achieve higher F1 scores than `MedPath`-TA in two-thirds separate settings in Table 2.3. Among them, the highest improvement in AUC is achieved in the case of LSTM, HiTANet and SAnD for heart failure, COPD and kidney disease, respectively. The highest F1 score improvement is achieved in the case of LSTM, LSTM and SAnD for heart failure, COPD and kidney disease, respectively. These models cover a broad range of risk prediction models,

including RNN-based, attention-based and time aware-based ones. In addition, `MedPath`-SA achieves higher average AUC and F1 scores on all three datasets. These results show that the use of relational self-attention contributes to better risk prediction performance in various types of risk prediction, and the designed relational self-attention is a better choice than the transition matrix-based attention.

To sum up, the improvement is brought by the personalized knowledge graph feature encoded by the graph neural network module. The new feature embeds the professional medical knowledge in SemMed, which makes the reasoning of the neural networks closer to the reasoning of physicians in real life. With the guidance of the personalized knowledge graph, our model could identify important ICD codes more easily in the records which have plausible relations with the target diseases and help improve the performance.

2.5.2.2 Significance Test

We also conduct significance test to compare the differences between all the results obtained by vanilla models and `MedPath` variants. To achieve this goal, we use the Bootstrap method to randomly choose 1,000 testing data and then calculate the values of AUC and F1 five times. The hypothesis is that the AUC or F1 means of baseline approaches are the same as those of the proposed `MedPath` variants. Student's T-test is used with significance level α as 1% to calculate the p-values. From the p-values listed in Tables 2.2 and 2.3, we reject the null hypothesis and accept the alternative hypothesis, i.e., true means are totally different. These results confirm that `MedPath` is significantly better than vanilla baselines.

2.5.2.3 Model Interpretability

In addition to performance improvement, the other reason that we incorporate medical paths is to provide explicit explanations to interpret the prediction results and highlight the influential relationship between different symptoms that will eventually lead to the target disease. In this part, we verify our claim by giving a case study on how to interpret the risk prediction with `MedPath` as shown in Table 2.4. The target disease in this case study is heart failure, and the first row shows the EHR data of the patient, which consist of 6 visits in total. Each visit is described with several diagnosis codes. For the ease of illustration, we only show the 2-hop paths in the personalized knowledge graph. In Table 2.4, we select three 2-hop paths with the highest attention weights and one 2-hop path with the lowest attention weight learned by `MedPath`-SA with LSTM. Here we can see how `MedPath` explicitly illustrates the correlation between the EHRs and the target

Table 2.4: Case study results of heart failure for showing the explicit interpretability that `MedPath` has.

EHR Data		**Visit 1**: 250.02 (Diabetes mellitus); **Visit 2**: 585.9 (Chronic kidney disease) and 780.79 (Fatigue); **Visit 3**: 244.9 (Hypothyroidism), 272.4 (Hyperlipidemia), and 401.1 (Benign essential hypertension); **Visit 4**: 585.9 (Chronic kidney disease); **Visit 5**: 585.9 (Chronic kidney disease); **Visit 6**: 585.9 (Chronic kidney disease) and 244.9 (Hypothyroidism)
1st Highest Attention Weighted Path	Weight: 0.0189	Hypothyroidism $\xrightarrow[E1]{CAUSES}$ Hypertensive disease $\xrightarrow[E2]{CAUSES}$ Left heart failure
	Evidence E1	*Animal studies suggest that hypertension leads to cardiac tissue hypothyroidism a condition that can by itself lead to heart failure.*
	Evidence E2	*Left ventricular failure in some SA/OHS patients may be the result of hypertensive cardiac disease.*
2nd Highest Attention Weighted Path	Weight: 0.0178	Hyperlipidemia $\xrightarrow[E3]{CAUSES}$ Hypertensive disease $\xrightarrow[E4]{CAUSES}$ Left heart failure
	Evidence E3	*A literature search indicates that Anglo-Saxon countries report alarming hyperplastic changes particularly in the liver blood clots hyperlipidemia leading to high blood pressure porphyria atypical leiomyomas and cervical hyperplasia.*
	Evidence E4	*Left ventricular failure in some SA/OHS patients may be the result of hypertensive cardiac disease.*
3rd Highest Attention Weighted Path	Weight: 0.0150	Fatigue $\xrightarrow[E5]{CAUSES}$ Cessation of life $\xrightarrow[E6]{CAUSES}$ Left heart failure
	Evidence E5	*In light of the magnitude of this sleep debt it is not surprising that fatigue is a factor in 57% of accidents leading to the death of a truck driver and in 10% of fatal car accidents and results in costs of up to 56 billion dollars per year.*
	Evidence E6	*Though rare death due to myocardial stunning and LV power failure can occur during ICD insertion.*
One of the Lowest Attention Weighted Path	Weight: 0.0000	Heart failure $\xrightarrow[E7]{CAUSES}$ Hypertensive disease $\xrightarrow[E8]{CAUSES}$ Left heart failure
	Evidence E7	*These findings suggest that the ATF3 activator tBHQ may have therapeutic potential for the treatment of pressure-overload heart failure induced by chronic hypertension or other pressure overload mechanisms.*
	Evidence E8	*Left ventricular failure in some SA/OHS patients may be the result of hypertensive cardiac disease.*

disease. Take ICD-9 code "244.9" (hypothyroidism) as an example, which has the largest attention weight. As the second row shows, the relation between hypothyroidism and heart failure disease is that it causes hypertensive disease and that can further lead to left heart failure. If we want to have supportive medical findings on how hypothyroidism can lead to hypertensive disease and how hypertensive disease can finally lead to left heart failure, we can see Evidence E1 and Evidence E2 in Table 2.4. They are the abstract sentences extracted by SemMed from reliable medical publications. Let us move to the third path, which is from fatigue to left heart failure and is not highly related to the target disease. Thus, the weight of this path is much lower than those of the top two paths. As for the path with the lowest attention weight, we can see that the source entity and the destination entity are all heart failure, which does not provide any useful information on the relation between the EHR example and target disease. Thus, it is correct for `MedPath` to give it a zero weight. This case study can clearly show the reasonableness of the proposed `MedPath` with reliable explanations on prediction results.

2.5.2.4 Ablation Study

Now we investigate our framework design by the ablation study. We break down our `MedPath` layer into type-specific transformation (Eq. (2.2)), node-type attention (i.e., $\gamma(\cdot)$

Table 2.5: Ablation study results of removing each component in MedPath-SA using LSTM as the encoder.

Dataset	Heart Failure	COPD	Kidney
Methods	AUC	AUC	AUC
MedPath-SA (k = 3)	**0.739**	**0.707**	**0.774**
- type specific transformation	0.707	0.701	0.759
- node type attention	0.714	0.704	0.746
- relational self attention	0.724	0.701	0.755

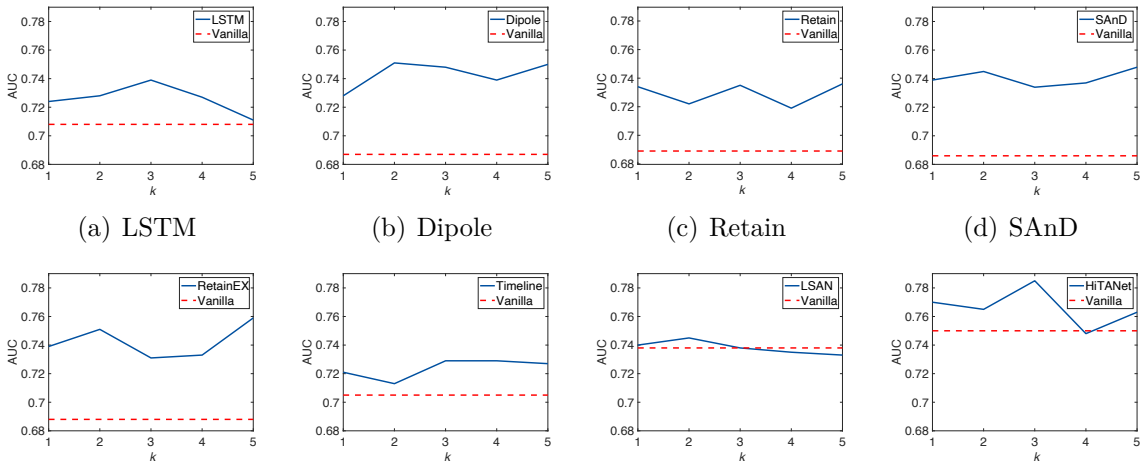
function in Eq. (2.8)), and relational self-attention components to see how they influence the model performance. Without loss of generalization, we use LSTM as the base model of MedPath-SA for analysis in this part. After removing each component individually, from Table 2.5 we can see that the model performance in AUC decrease for at least 1.5%, 0.3% and 1.5% in heart failure, COPD and kidney disease, respectively. These results conclude our ablation study and show that each component is essential for the design of MedPath.

2.5.2.5 Discussion for k Selection

As shown in Figure 2.2, Figure 2.3 and Figure 2.4, for each base model with relational self-attention, we increase k from 1 to 5 and compare the AUC scores in all test datasets. These figures shows that among 24 different testing settings, we can obtain the best AUCs when we choose k to be either 2 or 3 in 16 (two-thirds) cases. In addition to better AUC performance, selecting a relatively smaller k also provide straightforward interpretation for risk prediction. Therefore, we recommend tuning k to be 2 or 3.

2.5.2.6 Another Two Case Studies

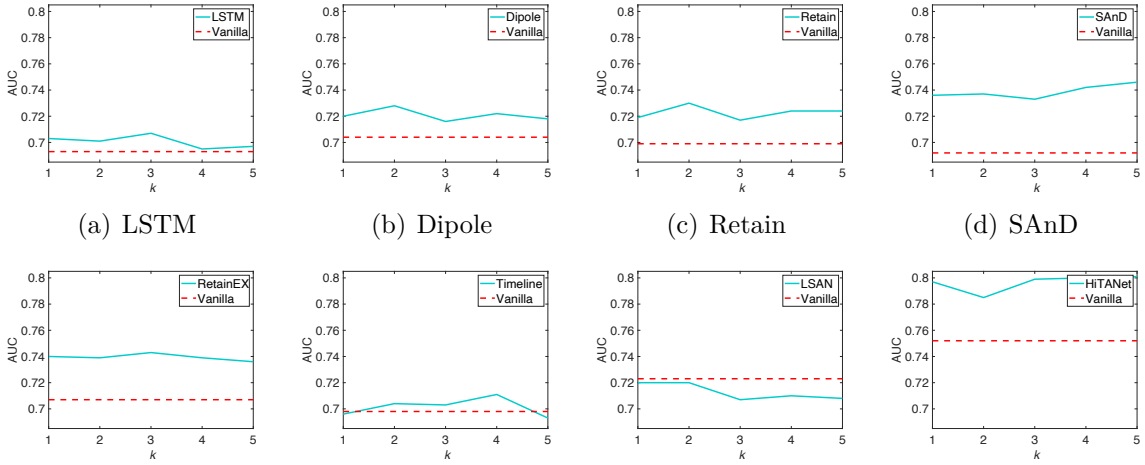
Here we provide two more examples to verify the explicit interpretability that MedPath has. For the ease of analysis, we only extract the 2-hop medical progression paths. We first analyze the positive COPD case as shown in Table 2.6. In this case, the patient has 6 visits, and each visit is described by at least 3 ICD-9 codes. As shown in the second part of Table 2.6, MedPath finds out three disease progression paths that are most likely lead to the COPD disease. The path having the highest attention weights starts from the symptom of gastroesophageal reflux disease ("530.81") in the first visit, which will affect the aspiration action of the patient and finally lead to the bronchitis disease. Another two possible paths also end with the bronchitis disease, a common condition of



(a) LSTM (b) Dipole (c) Retain (d) SAnD

(e) RetainEX (f) Timeline (g) LSAN (h) HiTANet

Figure 2.2: Comparison of AUC values with different k in different `MedPath`-SA methods on the heart failure dataset.

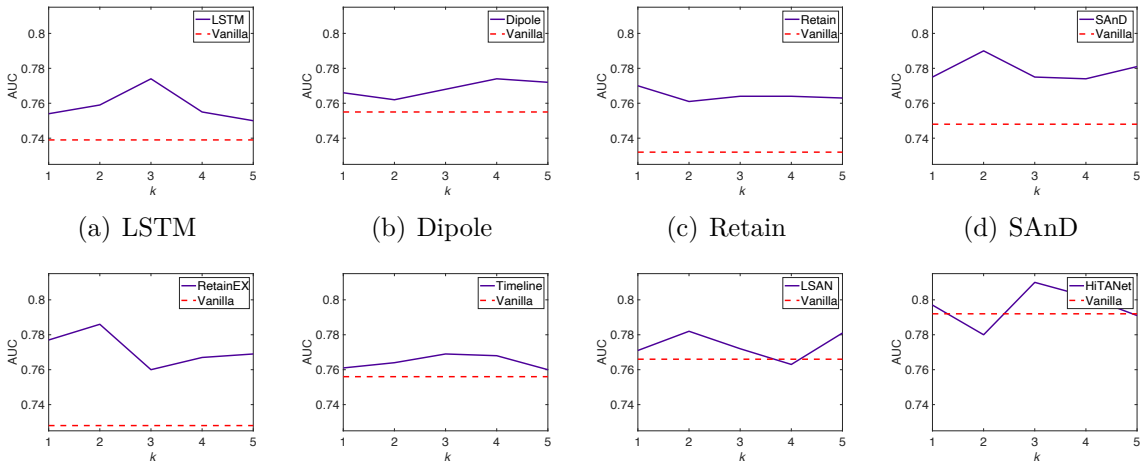


(a) LSTM (b) Dipole (c) Retain (d) SAnD

(e) RetainEX (f) Timeline (g) LSAN (h) HiTANet

Figure 2.3: Comparison of AUC values with different k in different `MedPath`-SA methods on the COPD dataset.

the COPD disease. One of them starts from the coronary arteriosclerosis ("414.00" in Visit 2, 5 and 6) that reduces blood flow in the heart and affects inflammation, while the other starts from the symptom of fatigue ("780.79" in Visit 5) which affects the Ammonia inside the body. The case study example of kidney disease is shown in Table 2.7. The patient in the case study has 8 visit records. The first thing we should note is that the symptom of hyperlipidemia ("272.4") is recorded in 5 visits, and the path starting from hyperlipidemia is assigned with the highest weight by `MedPath`. The path



(a) LSTM (b) Dipole (c) Retain (d) SAnD

(e) RetainEX (f) Timeline (g) LSAN (h) HiTANet

Figure 2.4: Comparison of AUC values with different k in different `MedPath`-SA methods on the kidney disease dataset.

with the second highest attention weight starts from rotator cuff syndrome ("726.1" in Visit 1) which causes the atrophic symptom and eventually the kidney failure, and we can have similar analysis on the path with the third highest attention weights. From the weight assignment results, we can see that `MedPath` can assign high weights to symptoms that occur repeatedly if they are correlated to the target disease, as evidenced by the hyperlipidemia symptom.

2.5.2.7 Performance Comparison with GRAM as the base model

In additional to the base models in Table 2.2, we also use GRAM as the baseline. From Table 2.8, we can see that in terms of AUC and F1 score, using `MedPath` framework can help us achieve improvement in 2 out 3 datasets. These results show that `MedPath` is model-agnostic and can also bring improvements to models using knowledge graph.

2.6 Conclusion

In this paper, we introduce a new general framework, called `MedPath`, to enable existing health risk prediction methods to incorporate personalized information and provide explicit interpretation for predictions. To find out the correlations between symptoms and target diseases, `MedPath` first extracts a personalized knowledge graph (PKG) for each patient from the SemMed web which contains a giant medical knowledge graph.

Table 2.6: Case study results of COPD for showing the explicit interpretability that `MedPath` has.

EHR Data	**Visit 1**: 719.41 (Shoulder joint pain), 530.81 (Gastroesophageal reflux disease) and 272.4 (Hyperlipidemia); **Visit 2**: 708.0 (Allergic urticaria), 272.4 (Hyperlipidemia) and 414.00 (Coronary Arteriosclerosis); **Visit 3**: 413.9 (Angina Pectoris), 272.4 (Hyperlipidemia), 786.50 (Chest Pain) and 425.4 (Cardiomyopathies); **Visit 4**: 426.3 (Left bundle branch block), 401.9 (Essential Hypertension) and 413.9 (Angina Pectoris); **Visit 5**: 786.50 (Chest Pain), 414.00 (Coronary Arteriosclerosis) and 780.79 (Fatigue); **Visit 6**: 300.00 (Anxiety state), 414.00 (Coronary Arteriosclerosis) and 272.4 (Hyperlipidemia)	
1st Highest Attention Weighted Path	Weight: 0.0153	Gastroesophageal reflux disease $\xrightarrow[E1]{AFFECTS}$ Aspiration-action $\xrightarrow[E2]{CAUSES}$ Bronchitis
	Evidence E1	*Gastroesophageal reflux, gastroparesis and achalasia are all associated with aspiration.*
	Evidence E2	*The absence of LLM in 29 control infants suggest that the aspiration may be one cause of recurrent bronchitis in infants.*
2nd Highest Attention Weighted Path	Weight: 0.0118	Coronary Arteriosclerosis $\xrightarrow[E3]{AFFECTS}$ Inflammation $\xrightarrow[E4]{CAUSES}$ Bronchitis
	Evidence E3	*Epicardial fat (EF) is a visceral fat deposit, located between the heart and the pericardium, which shares many of the pathophysiological properties of other visceral fat deposits, It also potentially causes local inflammation and likely has direct effects on coronary atherosclerosis.*
	Evidence E4	*It is speculated that an initial respiratory insult such as viral infection disrupts normal surface morphology and ciliary function, which leads to chronic self-perpetuating inflammation with the formation of bacterial biofilms, leading to PBB.*
3rd Highest Attention Weighted Path	Weight: 0.0118	Fatigue $\xrightarrow[E5]{AFFECTS}$ Ammonia $\xrightarrow[E6]{CAUSES}$ Bronchitis
	Evidence E5	*Serum levels of urea nitrogen (SUN), triglyceride fatty acids (TG), lactate dehydrogenase (LDH), lactic acid (LA), ammonia and hepatic glycogen (HG) were also examined for potential mechanisms underlying the anti-fatigue effect of RPL extracts.*
	Evidence E6	*Acute lung injuries caused due to inhalation of toxic irritant gases such as ammonia, chlorine, hot smoke and burning plastic fumes predominantly affect the airways, causing tracheitis, bronchitis, and other inflammatory responses.*
One of the Lowest Attention Weighted Path	Weight: 0.0000	Agonists $\xrightarrow[E7]{PART\_OF}$ Patients $\xrightarrow[E8]{AFFECTS}$ Bronchitis
	Evidence E7	*Treatment strategy for elderly diabetic patient with insulin or GLP-1 receptor agonist.*
	Evidence E8	*However, in half of the patients (15 cases) the cause was obscure although it was associated with sinusitis, bronchitis or bronchiectasis (Young\'s syndrome).*

After that, `MedPath` finds out all possible disease progression paths from the PKG and uses them to learn a personalized embedding to augment the base risk prediction models for the final prediction. Since the graph neural network encoder for PKG assigns attention weights on disease progression paths instead of independent medical codes, `MedPath` is able to provide reliable explicit explanations in the testing phrase by showing paths in PKG with high attention weights. Experimental results show that our model is able to improve the performance of existing models in terms of both F1 score and AUC. More importantly, in case studies, we confirm that `MedPath` can provide explicit explanations for its prediction through paths in PKG. In the future, we would like to work on how to allow risk prediction models to infer new emerging links between conditions.

Table 2.7: Case study results of kidney disease for showing the explicit interpretability that `MedPath` has.

EHR Data	**Visit 1**: 272.4 (Hyperlipidemia), 726.1 (Rotator cuff syndrome) and 401.1 (Benign essential hypertension); **Visit 2**: 401.1 (Benign essential hypertension), 272.4 (Hyperlipidemia) and 794.31 (EEG abnormal); **Visit 3**: 715.00 (Generalized osteoarthritis), 272.4 (Hyperlipidemia); **Visit 4**: 401.1 (Benign essential hypertension), 272.4 (Hyperlipidemia); **Visit 5**: 784.0 (Headache); **Visit 6**: 401.1 (Benign essential hypertension), 272.4 (Hyperlipidemia); **Visit 7**: 719.7 (Difficulty walking), 734 (Flat foot); **Visit 8**: 734 (Flat foot)	
1st Highest Attention Weighted Path	Weight: 0.0152	Hyperlipidemia $\xrightarrow[E1]{CAUSES}$ Hypertensive disease $\xrightarrow[E2]{AFFECTS}$ Kidney Failure
	Evidence E1	*A literature search indicates that Anglo-Saxon countries report alarming hyperplastic changes, particularly in the liver, blood clots, hyperlipidemia leading to high blood pressure, porphyria, atypical leiomyomas and cervical hyperplasia.*
	Evidence E2	*Various factors may play a role in the pathogenesis of hypertension in chronic renal failure.*
2nd Highest Attention Weighted Path	Weight: 0.0133	Rotator cuff syndrome $\xrightarrow[E3]{CAUSES}$ Atrophic $\xrightarrow[E4]{CAUSES}$ Kidney Failure
	Evidence E3	*The RCT made by transecting the supraspinatus (SSP) tendon resulted in atrophy of the SSP muscle.*
	Evidence E4	*Tubular atrophy in the pathogenesis of chronic kidney disease progression.*
3rd Highest Attention Weighted Path	Weight: 0.0119	Headache $\xrightarrow[E5]{CAUSES}$ Magnetic Resonance Imaging $\xrightarrow[E6]{DIAGNOSES}$ Kidney failure
	Evidence E5	*Visual failure poor growth or headache led to MRI diagnosis of CP.*
	Evidence E6	*These findings indicate that DTI MRI may be able to evaluate RF in CKD by DN.*
One of the Lowest Attention Weighted Path	Weight: 0.0000	Renal disease $\xrightarrow[E7]{AFFECTS}$ Homeostasis $\xrightarrow[E8]{AFFECTS}$ Kidney Failure
	Evidence E7	*Renal sympathetic nerve activity has an important role in renal disease-associated hypertension and in the modulation of fluid homeostasis.*
	Evidence E8	*Calcium phosphorus and magnesium homeostasis is altered in chronic kidney disease (CKD).*

Table 2.8: Performance Comparison with GRAM as the base model.

Dataset	Heart Failure			COPD			Kidney Disease		
Metric	Vanilla	`MedPath`-TA	`MedPath`-SA	Vanilla	`MedPath`-TA	`MedPath`-SA	Vanilla	`MedPath`-TA	`MedPath`-SA
AUC	0.748	**0.751**	0.749	0.722	0.741	**0.744**	**0.780**	0.775	0.778
F1 Score	0.628	**0.639**	0.632	0.582	**0.602**	0.600	**0.677**	0.659	0.666

Chapter 3
Augmenting Health Risk Prediction via Medical Texts

3.1 Introduction

Electronic health records (EHR) data have become increasingly available due to the wide adoption of EHR systems. As a result, the data mining community has been working on designing predictive models to predict patients' future health risks by extracting patterns from their EHR data, which is referred to as health risk prediction [4–6]. Commonly used EHR data such as health insurance claims for this task are encoded by International Classification of Diseases (ICD) codes,[1] and the challenges of learning from EHR data lie in their temporal, discrete, and sparse nature because patients usually have several visits but their symptoms are recorded with a small amount of codes.

To resolve the challenges, two main lines of strategies are generally adopted in existing methods. The first strategy focuses on utilizing temporal deep learning models to capture the complex temporal patterns inherent in the EHR data [4–7, 10–12]. The second strategy is to further augment the embedding learning and model interpretation with external knowledge. For instance, GRAM [13], KAME [14] and DG-RNN [15] use the medical knowledge graph as the external knowledge, while PRIME [16] incorporates prior medical knowledge to improve health risk prediction. Although existing works have demonstrated good performance, they still suffer from the following limitations.

Limitations of exiting works. The existing works fail to provide reliable interpretation that can be easily understood by patients. For existing models [4, 11, 12] using the first strategy, they usually use abstract attention weights to provide implicit interpretation for

[1] `https://www.cdc.gov/nchs/icd/icd9.htm`

the predictions, which are hard to understand and unreliable without explaining the logic. For the second strategy, the interpretation obtained from knowledge graph is hard for patients without professional training to understand. In addition, the relation knowledge between medical entities of used medical knowledge graphs such as KnowLife [18] and SemMed [29, 30] are represented by CUIs (Concept Unique Identifiers) instead of ICD codes. Although SNOMED-CT [31] provides mapping from ICD codes to CUIs, only about 70% ICD codes can be mapped to CUI codes, which inevitably loses diagnosis information during transformation. Besides, the used hierarchical ICD ontology cannot capture the relations between codes belonging to different categories. Thus, this issue raises a question: *Is there unused external knowledge that can improve the interpretability and performance of health risk prediction models?*

Rethinking health risk prediction. Our answer to this question is to use the unstructured medical text collected from authoritative online sources such as Mayo Clinic[2] and WebMD,[3] which provide a large number of informative descriptions related to different symptoms and can be easily understood by humans. The rationale for using medical text for interpretation is that each ICD code is not only a discrete symbol but it also represents a symptom or abnormal finding associated with a short description by human language. For example, "250.0" refers to *diabetes mellitus without mention of complication*. Therefore, medical text is an organic source for interpretable health risk prediction. However, to utilize it in health risk prediction, we need to solve the following challenges.

• *Retrieving unstructured medical text.* Due to the limited length, the short descriptions cannot explicitly demonstrate the relations with the target disease. Although as mentioned above we can find the relations from the unstructured medical text which contains richer and more understandable information, they are noisy and not directly associated with every ICD code. Therefore, the first challenge is how to retrieve useful information that connects input ICD codes and the target disease.

• *Exploiting the informative target disease documents.* Compared to general classification tasks, the target disease stood by the label in the health risk prediction task is a specific disease or condition whose unique characteristics can be described by human languages. For example, the target disease "*heart failure*" is described by a combination of certain symptoms, causes, risk factors, and so on.[4] Hence, modeling the inference

[2] `https://www.mayoclinic.org/`

[3] `https://www.webmd.com/`

[4] `https://www.mayoclinic.org/diseases-conditions/heart-failure/symptoms-causes/syc-20373142`

direction from the target disease to the input data can help the model improve the interpretation comprehensibility. For instance, with the knowledge of risk factors we can discover whether there are any ICD code worth noting. This motivation leads to a new challenge of how to explicitly equip the predictive models with the target disease documents.

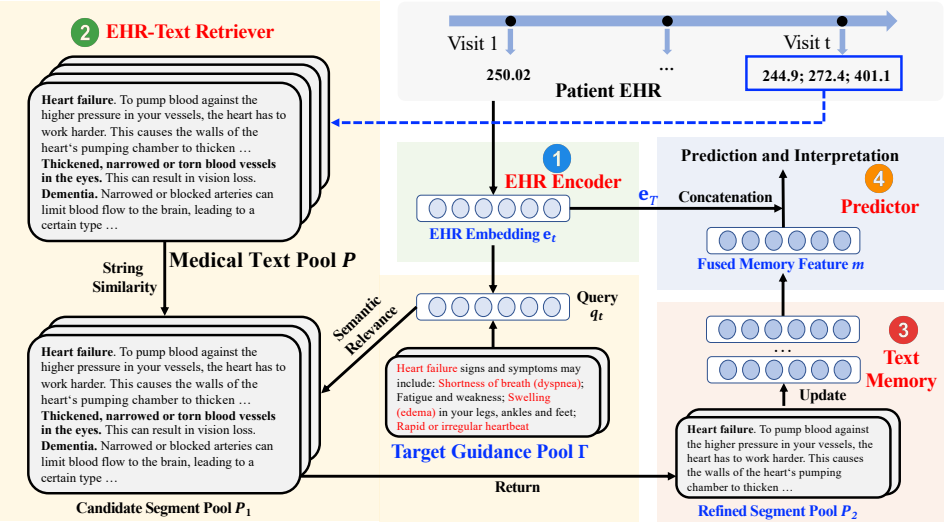


Figure 3.1: Overview of the proposed `MedRetriever`.

The proposed approach. To solve the aforementioned challenges, as shown in Figure 3.1 we propose an effective and general framework named `MedRetriever`, which utilizes a new form of external knowledge, i.e., unstructured medical text, in a target-driven manner. `MedRetriever` includes four modules: *EHR Encoder*, *EHR-Text Retriever*, *Text Memory*, and *Predictor*. Firstly, `MedRetriever` is a framework that is flexible in the choice of *EHR Encoder*. That is, most health risk prediction models can be used as the backbone for learning EHR embeddings. Next, *EHR-Text Retriever* completes the retrieval in two stages. In the first stage, it obtains a candidate segment pool based on string similarity scores between the ICD code descriptions and the medical text corpus. Given the embedding of each visit and target disease documents, in the second stage we use them to generate a query aggregated with generalized and personalized information to retrieve relevant text segments from the candidate segment pool. During this process, *Text Memory* stores and dynamically updates the top text segments relevant to both the symptoms of patients and the target disease. Finally, the comprehensive EHR embedding and the memorized texts are aggregated together to conduct the final prediction and interpretation. A salient benefit of using the proposed method is to provide reliable

interpretation for decision-making using natural language, which are highly readable and understandable by patients.

Contributions. To be specific, our contributions are as follows:

- To the best of our knowledge, we are the first to exploit the unstructured medical text as the external knowledge with the emphasize on the target disease documents to improve the EHR embedding and model interpretability for the health risk prediction task, which is made publicly available.
- We design a novel general framework named `MedRetriever` to process medical text for health risk prediction, which is flexible in the choice of EHR embedding backbone with high expandability.
- We propose a new retrieval and memory mechanism to keep the most relevant text segments during the iteration process. The retrieval process uses both generalized and personalized information to obtain relevant medical texts by the query features aggregated with EHR embeddings and target disease documents in self-attention, and the saved texts in the memory can be used for generating understandable interpretation.
- The experimental results show that compared to the state-of-the-art approaches, `MedRetriever` achieves the best performance measured by AUC, recall and F1 score in most cases on three health insurance claims datasets. Besides, case studies further show the understandable interpretation of model prediction.

3.2 Literature Review

In the healthcare data mining domain, using memory mechanism can help models increase their capacities in memorizing medical knowledge and historical patient data. In [39], the authors propose a model named DMNC, which is a memory augmented neural network for the medication combination recommendation task on EHR data. In [40], the authors use a memory component to fuse multi-model graphs as a memory bank for medication recommendation. The drawback of these existing works is that their memories memorize abstract feature vectors instead of human readable text. On the contrary, our method `MedRetriever` [41] chooses to store the retrieved medical texts as well as their features for improving the prediction performance and interpretability. With the help of text memory, `MedRetriever` can dynamically find out the relevant medical text segments and use it to provide understandable interpretation.

3.3 Data & Task

The risk prediction task is to predict the future status of patients based on their historical electronic health records (EHRs). In this paper, we propose to utilize unstructured medical text as the external knowledge. We now formally define our input data and the task as follows.

3.3.1 Electronic Health Records

The EHRs of all patients are encoded by a high dimensional ICD-9 dictionary denoted as $\mathcal{C} = \{c_1, \cdots, c_N\}$, where N represents the number of unique ICD codes in $\mathcal{C}$. Mathematically, let $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_T]$ denote the EHR data of a specific patient , where $\mathbf{v}_t$ $(1 \leq t \leq T)$ represents the diagnoses of visit t, and T is the total number of visits. Each individual visit $\mathbf{v}_t$ contains a subset of ICD codes $\mathcal{C}_t \subseteq \mathcal{C}$, where $|\mathcal{C}_t| = n_t$ is the number of ICD codes in t-th visit. $\mathbf{v}_t$ can be represented by a binary vector $\{0, 1\}^N$, where the i-th position is 1 when the corresponding diagnosis code appears in the t-th visit, and 0 otherwise.

3.3.2 Medical Text Corpus

As previously showed in the example ICD-9 code "250.0", for each code $c_i \in \mathcal{C}$ it has a short description sentence for what symptom or abnormal finding it represents. However, the description is too abstract for humans, not to mention computers, to infer the disease progression and provide interpretation between the symptoms and target diseases for patients. To alleviate this limitation, we collect and introduce authoritative medical text as another input for constructing interpretable health risk prediction framework. Specifically, we crawl the medical texts from Mayo Clinic and WebMD because they provide organized professional descriptions for more than 1,000 diseases or conditions, which include symptoms, causes, risk factors, and complications. To facilitate the text processing by deep neural networks, we further divide each crawled document D_i into segments $[D_{(i,1)}, D_{(i,2)}, \cdots, D_{(i,\tau_i)}]$ according to their corresponding sections in D_i, where segment $D_{(i,j)}(1 \leq j \leq \tau_i)$ is the j-th segment of D_i and τ_i is the total number of segments in D_i.

With the crawled medical text corpus, we select the segments related to the target disease as the set of **target guidance pool** $\Gamma = \{\gamma_1, \cdots, \gamma_k\}$, where k is the total number of target disease segments. The rest h collected documents form the **medical**

text pool $\mathcal{P} = \{D_1, D_2, \cdots, D_h\}$. We distinguish these two text sets in order to make our model **target-driven**, which can explicitly model the decision-making of human doctors and provide informative interpretation, and Γ and $\mathcal{P}$ constitutes the medical text corpus.

3.3.3 Health Risk Prediction

Now given the input EHR $\mathbf{V}$ of a patient and the medical text corpus Γ and $\mathcal{P}$, the risk prediction task is to design a function f that can accurately predict the status of patient with respect to the target disease. The ground truth y is set to 1 when the patient will suffer from the target disease, and to 0 otherwise. The prediction result given by f is

$$\hat{y} = f(\mathbf{V}, \Gamma, \mathcal{P}). \tag{3.1}$$

The aim of function f is to provide an accurate prediction such that $\hat{y}$ is as close as y. In addition, we reframe the problem as a target-driven one by asking f to employ the correlation between $\mathbf{V}$, Γ, and $\mathcal{P}$ for prediction and interpretation.

3.4 Methodology

As shown in Figure 3.1, `MedRetriever`, the proposed solution to Eq. (3.1), will process the input EHR and medical text corpus through the following modules. First, an **EHR encoder** will learn EHR embedding $\mathbf{e}_t$ for the EHR from $\mathbf{v}_1$ to $\mathbf{v}_t$, then an **EHR-Text retriever** will first retrieve a subset of segments $\mathcal{P}_1$ from $\mathcal{P}$ by using the ICD-9 code descriptions in the t-th visit, and learn a query $\mathbf{q}_t$ based on the target guidance pool Γ and the EHR embedding $\mathbf{e}_t$ via self-attention. The query $\mathbf{q}_t$ will be used to retrieve μ segments with the highest relevance scores (denoted as $\mathcal{P}_2$) from $\mathcal{P}_1$. Next, a **text memory** module is designed to store top κ relevant medical texts for interpretation, which will be updated with the newly retrieved $\mathcal{P}_2$ in each time step. After finishing the iteration process above, in the final time step **predictor** makes **prediction** and **interpretation** based on the comprehensive EHR embedding and text memory. Below are the detailed mechanism of each module.

3.4.1 EHR Encoder

`MedRetriever` is a general framework that can be built on various existing EHR encoding approaches. We notice that a majority of existing health risk prediction studies require either recurrent neural networks (RNNs) or Transformer [21] to embed the EHRs, which can output features step by step during the process of learning a comprehensive feature. Thus, in this paper, the EHR encoder backbones include RNN-based models [4–7, 13, 20] and Transformer-based models [10–12].

Let f_b denote the EHR encoder backbone, which is used to learn an embedding from visit $\mathbf{v}_1$ to visit $\mathbf{v}_t$ in each time step. That is, in time step t, the backbone f_b will take all the previous visits $[\mathbf{v}_1, \cdots, \mathbf{v}_t]$ as the input to learn the embedding $\mathbf{e}_t$,

$$\mathbf{e}_t = f_b([\mathbf{v}_1, \cdots, \mathbf{v}_t]), \tag{3.2}$$

where $\mathbf{e}_t \in \mathbb{R}^{d_1}$, and d_1 is the dimension of output embedding. The embedding $\mathbf{e}_t$ will later be used to retrieve unstructured medical text segments and update the text memory.

3.4.2 EHR-Text Retriever

The EHR-Text retriever is a key module in our model, which aims to retrieve target disease-related segments to improve both performance and model interpretability. The retrieval process includes two steps, i.e., preliminary retrieval and refined retrieval, which are described as follows.

Preliminary retrieval by string similarity. Recall that each ICD-9 code c_i in visit $\mathbf{v}_t = [c_1, c_2, \cdots, c_{n_t}]$ has a short description sentence denoted as r_i. Since the short description sentence of each ICD-9 code is the equivalent description in human language, we should make use of the description sentence of each input ICD-9 code to condense the retrieval space and search for potential relevant segments for interpretation in $\mathcal{P}$ by conducting a preliminary retrieval. In addition, because the sentence is relatively short, we can simply use the string similarity to filter out some irrelevant document segments in $\mathcal{P}$.

To specific, for each code c_i, we use the Levenshtein distance [42] to measure the similarity between r_i and each segment in the medical text pool $\mathcal{P} = \{D_1, \cdots, D_h\} = \{[D_{(1,1)}, D_{(1,2)}, \cdots, D_{(1,\tau_1)}], \cdots, [D_{(h,1)}, D_{(h,2)}, \cdots, D_{(h,\tau_h)}]\}$. If the token set ratio between r_i and a certain segment $s \in \mathcal{P}$ is greater than a threshold ϵ, then s is regarded as being related to r_i and put into the candidate segment pool $\mathcal{P}_1$. In other words, for each time

step t, we will only keep the document segments that are related to $\mathbf{v}_t$ in the candidate segment pool $\mathcal{P}_1$ based on the string similarity. Suppose there are l segments in total in $\mathcal{P}_1$, then we have $\mathcal{P}_1 = \{s_1, \cdots, s_l\}$, which is the output of the preliminary retrieval.

Refined retrieval by semantic relevance. In the next step, we need to further find out top semantically relevant segments measured in the embedding space, which is necessary because it can organically integrate retrieval with the end-to-end training of deep neural networks. In this step, we first need to generate a query $\mathbf{q}_t$ from the EHR embedding $\mathbf{e}_t$ and the target guidance pool Γ. After that, we use the query to retrieve semantically relevant segments from the candidate segment pool $\mathcal{P}_1$ in the embedding space.

• *Query generation.* A good query for retrieving the medical text pool should have both generalized information from the target disease and personalized information from the EHR of patient. Therefore, given that we already have EHR embedding $\mathbf{e}_t$, to generate a query the first thing that we are required to do is to obtain an embedding for each segment of the target guidance pool $\Gamma = \{\gamma_1, \cdots, \gamma_k\}$. To solve this problem, we employ a language model called PubMedBERT [43] pre-trained on biomedical publications in the PubMed database and a two-layer multilayer perceptrons denoted as MLP_1 for finetuning in our datasets to learn an embedding for each segment.[5] Denoting PubMedBERT as f_s, we get the embedding for segment γ_i $(1 \leq i \leq k)$ by

$$\mathbf{x}_i = \text{MLP}_1(f_s(\gamma_i)), \tag{3.3}$$

where $\mathbf{x}_i \in \mathbb{R}^{d_2}$ and d_2 is the dimension of embedding in both layers of MLP_1. Thus, we can get a set of embeddings for Γ as $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_k]$.

After getting the embeddings for the target disease-related segments, we need to generate a query aggregated with both generalized and personalized information. Toward this end, we use self-attention [21] mechanism to aggregate the EHR embedding $\mathbf{v}_t$ with the target disease semantics $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_k]$ as follows:

$$\mathbf{q}_t = \mathbf{W}_O \cdot \text{softmax}(\frac{\mathbf{U}_Q \mathbf{U}_K^{\mathrm{T}}}{\sqrt{d_3}}) \cdot \mathbf{U}_V, \tag{3.4}$$

where $\mathbf{q}_t \in \mathbb{R}^{d_1}$ denotes the generated query, $\mathbf{U}_Q = \mathbf{W}_Q \mathbf{e}_t$, $\mathbf{U}_K = \mathbf{W}_K \mathbf{X}$ and $\mathbf{U}_V = \mathbf{W}_V \mathbf{X}$ are the query, key and value matrix, respectively, and the transformation matrices $\mathbf{W}_Q \in \mathbb{R}^{d_3 \times d_1}$, $\mathbf{W}_K \in \mathbb{R}^{d_3 \times d_2}$, $\mathbf{W}_V \in \mathbb{R}^{d_3 \times d_2}$, and $\mathbf{W}_O \in \mathbb{R}^{d_1 \times d_3}$.

[5] `https://pubmed.ncbi.nlm.nih.gov/`

In Eq. (3.4), the self-attention-based mechanism we design for generating query embeddings can help merge generalized information from $\mathbf{X}$ and the personalized embedding $\mathbf{e}_t$, which contributes to expanding the application of self-attention.

• *Segment retrieval.* Now we have a new feature $\mathbf{q}_t$ aggregating the visit embedding with the target disease semantics, the next thing to do is to retrieve semantically relevant segments from the candidate segment pool $\mathcal{P}_1$. Similar to Eq. (3.3), we can get the embedding $\mathbf{g}_i$ for each segment $s_i \in \mathcal{P}_1$ by $\mathbf{g}_i = \text{MLP}_1(f_s(s_i))$. After that, we can get the top μ relevant segments from $\mathcal{P}_1$ ranked by the relevance scores between $\mathbf{q}_t$ and each $\mathbf{g}_i$, and the relevance score is calculated from a two-layer multilayer perceptrons MLP_2,

$$\text{relevance score} = \text{MLP}_2(\text{concate}(\mathbf{q}_t, \mathbf{g}_i)), \tag{3.5}$$

where concate refers to the concatenation operation, and the first layer of MLP_2 maps a $(d_1 + d_2)$-dimensional vector to a d_4-dimensional vector, and the final layer of MLP_2 maps a d_4-dimensional vector to a real value as the relevance score.

After ranking the segments in the descending order of the relevance scores calculated by Eq. (3.5), we take the top μ segments $\mathcal{P}_2 = \{s_{(1)}, \cdots, s_{(\mu)}\}$ from the pool $\mathcal{P}_1$ as the refined retrieval result, which will then be used to update the text memory.

3.4.3 Text Memory

During the retrieval process of medical text, we maintain a text memory called TEMem which can memorize up to $\mu + \kappa$ text segments to record the highly relevant segments from $\mathcal{P}_2$. Suppose after $(t-1)$ iterations, TEMem stores κ retrieved segment embeddings, where $\text{TEMem} = [\mathbf{m}_{(1,t-1)}, \cdots, \mathbf{m}_{(\kappa,t-1)}]$. Now we update TEMem in the t-th iteration with $\mathcal{P}_2$ as follows. By filling the embeddings of μ text in $\mathcal{P}_2$ into the text memory, we have $\text{TEMem} = [\mathbf{g}_{(1)}, \cdots, \mathbf{g}_{(\mu)}, \mathbf{m}_{(1,t-1)}, \cdots, \mathbf{m}_{(\kappa,t-1)}]$, which contains the candidates for updating the text memory in the t-th iteration.

Similar to the calculation in Eq. (3.5), we use another multilayer perceptrons denoted MLP_3 to get the relevance scores between the query $\mathbf{q}_t$ and all the segment vectors stored in TEMem. After getting the relevance scores for $\mu + \kappa$ embeddings in TEMem, we only keep the top κ embeddings with the highest relevance scores in the memory and get the updated text memory

$$\text{TEMem} = [\mathbf{m}_{(1,t)}, \cdots, \mathbf{m}_{(\kappa,t)}], \tag{3.6}$$

which saves the the most relevant medical text segments after t iterations given the newly

retrieved $\mathcal{P}_2$.

Note that there may be repeated ICD-9 codes appearing in different visits, and some text segments may be retrieved multiple times. In our implementation we will keep the repeated segments because it indicates that they are important for the interpretation when they are always stored in the memory.

3.4.4 Predictor

After iterating over T visits, the final outputs we have are the comprehensive visit feature $\mathbf{e}_T$ and the text memory TEMem $= [\mathbf{m}_{(1,T)}, \cdots, \mathbf{m}_{(\kappa,T)}]$. To get the feature for the final prediction, we firstly fuse the embeddings stored in TEMem into a single feature $\mathbf{m}$ by max pooling operation over each feature dimension as follows:

$$\mathbf{m} = \text{maxpooling}([\mathbf{m}_{(1,T)}, \cdots, \mathbf{m}_{(\kappa,T)}]), \tag{3.7}$$

where $\mathbf{m} \in \mathbb{R}^{d_2}$. With the comprehensive feature $\mathbf{e}_T$ and the fused memory feature $\mathbf{m}$, we make the final prediction by concatenation, linear transformation and softmax activation:

$$\hat{y} = \text{softmax}(\mathbf{w}_p^\top \cdot \text{concate}(\mathbf{e}_T, \mathbf{m}) + b), \tag{3.8}$$

where $\mathbf{w}_p \in \mathbb{R}^{d_1+d_2}$ and $b \in \mathbb{R}$ are the learnable parameters of linear transformation.

In addition, we also record the indices of the corresponding features saved in the text memory in each time step. After making the prediction, `MedRetriever` outputs the memorized medical text segments to explain what external knowledge is used for decision-making. Hence, our method is more interpretable than previous ones because its decision can be expressed by natural language.

3.5 Experiments

In this section, we will show the experimental results including performance comparison, ablation study and case studies to evaluate the performance and interpretability of `MedRetriever` against the state-of-the-art baselines.

3.5.1 Experimental Setup

3.5.1.1 Datasets

Our experiments are conducted on three EHR datasets collected from real-world health insurance claim data by a health information technology company, whose target disease is Heart Failure, Chronic Obstructive Pulmonary Disease (COPD), and Kidney Disease, respectively. Patients of these diseases normally experience a chronic and progressive condition for a long period. The statistics of these datasets are shown in Table 3.1.

Table 3.1: Statistics of the used claim datasets.

Dataset	Heart Failure	COPD	Kidney Disease
Positive Cases	3,080	7,314	2,810
Negative Cases	9,240	21,942	8,430
Average Visits per Patient	38.74	30.39	39.09
Average Codes per Visit	4.24	3.50	4.40
Unique ICD-9 Codes	8,692	10,053	8,802

In addition, the collected medical text corpus including Γ and $\mathcal{P}$ contains 1,148 diseases or conditions with 17,691 segments, which leads to 15.4 segments for each disease or condition on average. For each segment, the average number of words is 41.4.

3.5.1.2 Baselines

We consider four types of state-of-the-art health risk prediction models as baselines, which can also be used as the EHR encoder of `MedRetriever`: (1) **plain RNN** including LSTM [20] which is a commonly used baseline; (2) **temporal DNN-based models** which are built on RNN or Transformer and employ attention mechanisms to aggregate visit embeddings, including Dipole [5], Retain [4], SAnD [10], and LSAN [12]; (3) **time-aware models** which consider the importance of time information associated with visits, including RetainEx [6], Timeline [7], and HiTANet [11]; and (4) **knowledge graph-based model** including GRAM [13] which takes the ICD ontology as the external knowledge to improve the ICD code embedding learning.

3.5.1.3 Implementation

We implement `MedRetriever` by the PyTorch framework on an NVIDIA Tesla V100 GPU. The parameters are trained by Adam optimizer with the learning rate of 10^{-4} and weight decay of 10^{-3}, and the mini-batch size is set to 64. The numbers of hidden state

of baselines are all 256. In `MedRetriever`, all the feature sizes from d_1 to d_4 are also 256, the size of $\mathcal{P}_2$ is $\mu = 15$, and the memory size $\kappa = 20$. Besides, we use different ways to extract visit representations $\mathbf{e}_t$ based on the characteristics of each backbone in ablation study. For backbones using RNNs or Transformer for feature learning, we can simply extract the output of RNNs and Transformer in each time step as $\mathbf{e}_t$ for retrieval and prediction. As for Retain and RetainEx which only apply RNNs to generate attention weights assigned to the input embedding matrix, we use the weighted embedding matrix as visit representations. Moreover, we randomly partition the datasets into training set, validation set, and test test in the ratio of 0.75:0.10:0.15. We select the best model based on the performance on the validation set, and we run the algorithms three times and report both the mean and standard deviation of metrics for performance evaluation.

3.5.1.4 Evaluation Metrics

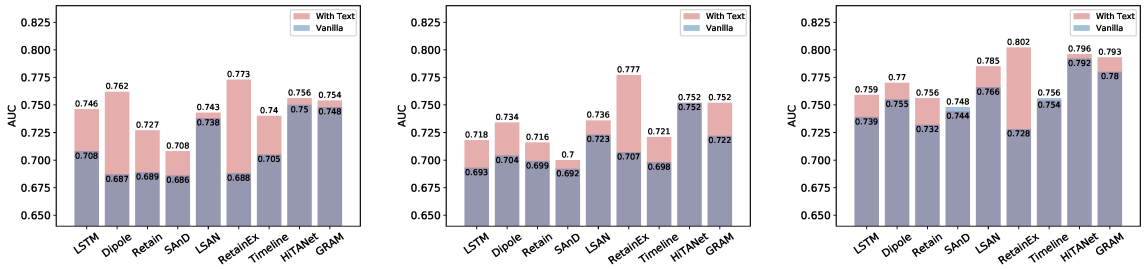
To fairly compare `MedRetriever` with baselines, we use *AUC* (area under the receiver operating characteristics), *precision*, *recall*, and *F1 score* as the evaluation metrics, which are the most commonly used ones for evaluating health risk prediction models.

Table 3.2: Performance comparison in terms of AUC, Precision, Recall and F1 score. Note that `MedRetriever` uses RetainEx as the backbone, and we report the mean and standard deviation values of the results after running three times.

Dataset	Heart Failure				COPD				Kidney Disease			
Metrics	AUC (↑)	Precision (↑)	Recall (↑)	F1 (↑)	AUC (↑)	Precision (↑)	Recall (↑)	F1 (↑)	AUC (↑)	Precision (↑)	Recall (↑)	F1 (↑)
LSTM	0.708	0.640	0.510	0.561	0.693	0.680	0.461	0.548	0.739	0.680	0.572	0.616
Dipole	0.687	0.713	0.445	0.542	0.704	0.687	0.477	0.562	0.755	**0.771**	0.571	0.656
Retain	0.689	0.655	0.474	0.549	0.699	0.696	0.463	0.555	0.732	0.706	0.544	0.614
SAnD	0.686	0.661	0.466	0.544	0.692	0.653	0.462	0.539	0.748	0.690	0.592	0.636
LSAN	0.738	0.621	0.626	0.623	0.723	0.661	0.500	0.574	0.766	0.651	0.672	0.661
RetainEx	0.688	0.730	0.438	0.546	0.707	**0.728**	0.470	0.570	0.728	0.745	0.520	0.612
Timeline	0.705	0.661	0.510	0.574	0.698	0.654	0.478	0.550	0.756	0.697	0.607	0.648
HiTANet	0.750	**0.724**	0.587	0.647	0.752	0.707	0.583	0.637	0.792	0.743	0.668	**0.702**
GRAM	0.748	0.570	0.698	0.628	0.722	0.603	0.562	0.582	0.780	0.681	0.672	0.677
`MedRetriever`	**0.773**	0.595	**0.746**	**0.660**	**0.777**	0.576	**0.725**	**0.645**	**0.802**	0.636	**0.763**	0.688
(std)	(7e-3)	(4e-2)	(3e-2)	(1e-2)	(6e-3)	(2e-2)	(3e-2)	(2e-3)	(7e-3)	(5e-2)	(4e-2)	(1e-2)

3.5.2 Performance Evaluation

In Table 3.2 we show the experimental results of all approaches on three test datasets in terms of four evaluation metrics. For `MedRetriever`, the results are obtained by using RetainEx as the EHR encoder backbone. We can observe that `MedRetriever` performs stably and outstandingly, achieving the state-of-the-art performance measured by most evaluation metrics on three datasets.



(a) Heart Failure (b) COPD (c) Kidney Disease

Figure 3.2: Comparison of AUCs with different baselines as the EHR encoder backbone.

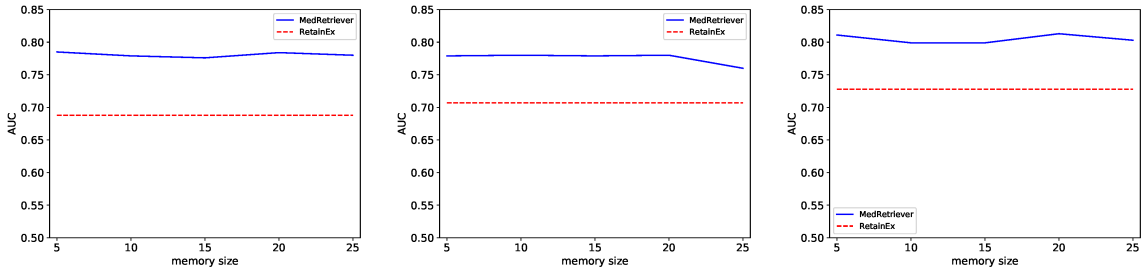
Table 3.3: Ablation study results in term of AUC when removing each medical text processing component of `MedRetriever`, which uses RetainEx as the EHR encoder backbone.

Dataset	Heart Failure	COPD	Kidney Disease
`MedRetriever`	**0.773**	**0.777**	**0.802**
without Target Guidance	0.762	0.751	0.784
without Refined Retrieval	0.769	0.775	0.797
without Text Memory	0.766	0.773	0.796

In particular, the AUC and recall of `MedRetriever` are always the highest, and the values of recall increase 4.8%, 14.2%, and 9.1% compared to the second best ones, respectively. We note that `MedRetriever` sacrifices some precision for high recall, which is beneficial for the health risk prediction task. This is because the target diseases are usually life-threatening to the patients, and in the healthcare domain we wish machine learning models not to have too many false negative cases, which is penalized in the recall metric. As for F1 score which is the harmonic mean of precision and recall, `MedRetriever` achieves the highest on the Heart Failure and COPD datasets and the second highest on Kidney Disease dataset. Therefore, based on the analysis of four evaluation metrics, `MedRetriever` surpasses the current state-of-the-art methods with respect to health risk prediction performance.

3.5.3 Ablation Study

We further conduct the following ablation studies to examine the design of `MedRetriever`. Firstly, to show that `MedRetriever` is a flexible framework that can benefit most EHR encoder backbones by retrieving and memorizing the medical text corpus, we compare the performance with nine vanilla baseline models in terms of AUC score as shown in



(a) Heart Failure (b) COPD (c) Kidney Disease

Figure 3.3: Comparison of AUCs with different memory sizes in `MedRetriever` using RetainEx as the backbone on three datasets.

Figure 3.2. For almost every baseline `MedRetriever` can bring performance gain on all three datasets, even for a complex model such as GRAM which is an RNN-based model utilizing the knowledge graph. Specifically, RNN-based models, including LSTM, Dipole, Retain, RetainEx, and Timeline, could get larger improvement with medical text than Transformer-based models such as SAnD, LSAN, and HiTANet. Besides, we can observe that RNN-based models can achieve higher improvement, which is mainly because their fashion of processing EHR data step-by-step fits well with the step-by-step querying process from Γ. Particularly, with RetainEx as the backbone, `MedRetriever` obtains a margin of 8.5%, 7.0%, and 7.4% in terms of AUC over the vanilla RetainEx model on the Heart Failure, COPD, and Kidney Disease datasets, respectively, which is the reason that we select it as the backbone for `MedRetriever`. Such significant improvements prove that medical text corpus can work well with existing baselines and increase their prediction performance.

After examining the effectiveness of medical text corpus and deciding to use RetainEx as the backbone, we then conduct an ablation study to validate the effectiveness of the designed components for medical text processing, which include using target guidance pool for query generation, refined retrieval, and text memory. By removing each component individually from the original framework, we obtain the results as shown in Table 3.3. We can observe that the model performance drops in all the datasets, which confirms that these component is essential for medical text processing: firstly, after excluding Γ as target guidance and directly using EHR embedding obtained by Eq. (3.2) as the query in Eq. (3.4), we observe that the performance decreases by 1.1%, 2.6%, and 1.8% in terms of AUC on the three test datasets, respectively. Thus, it is essential to incorporate the target guidance. Secondly, after removing the refined retrieval stage and directly using the outputs from the preliminary retrieval to update the memory, the AUC still drops

by 0.4%, 0.2%, 0.5% compared with `MedRetriever` on the three datasets, respectively, which proves the necessity of refined retrieval. Thirdly, as for the last scenario where we only use the EHR embedding e_T output in the last visit T to retrieve relevant texts without dynamically update, the model performance decreases by around 0.7% on three datasets. Thus, the dynamical update capacity of text memory is useful for processing medical text. We also further vary the value of κ of the text memory from 5 to 25 to investigate the influence of memory size. As shown in Figure 3.3, we observe that setting $\kappa = 20$ can help achieve best performance in test datasets.

Table 3.4: Case study of a positive case on the heart failure dataset.

EHR		**Visit 1:** Diabetes mellitus (250.00), Atrial fibrillation (427.31), Vaginitis and vulvovaginitis (616.10), Benign essential hypertension (401.1) **Visit 2:** Senile osteoporosis (733.01) **Visit 3:** Benign essential hypertension (401.1), Diabetes mellitus (250.00), Atrial fibrillation (427.31) **Visit 4:** Atrial fibrillation (427.31) **Visit 5:** Coronary atherosclerosis of native coronary artery (414.01), Atrial flutter (427.32), Diseases of tricuspid valve (397.0)
Visit 1	Target Guidance	1. Other diseases. Chronic diseases — such as diabetes, HIV, hyperthyroidism, hypothyroidism, or a buildup of iron (hemochromatosis) or protein (amyloidosis) — also may contribute to heart failure. *(Weight: 0.02384)* 2. Coronary artery disease. Narrowed arteries may limit your heart's supply of oxygen-rich blood, resulting in weakened heart muscle. *(Weight: 0.02381)* 3. Diabetes. Having diabetes increases your risk of high blood pressure and coronary artery disease. *(Weight: 0.02379)*
	Text Memory	1. Age. The older you are, the greater your risk of developing atrial fibrillation. *(Weight: 0.0701)* 2. Inactivity. The less active you are, the greater your risk. Physical activity helps you control your weight, uses up glucose as energy and makes your cells more sensitive to insulin. *(Weight: 0.0698)* 3. Weight. Being overweight before pregnancy increases your risk of diabetes. *(Weight: 0.0686)*
Visit 2	Target Guidance	1. Diabetes. Having diabetes increases your risk of high blood pressure and coronary artery disease. *(Weight: 0.02383)* 2. But heart failure can occur even with a normal ejection fraction. This happens if the heart muscle becomes stiff from conditions such as high blood pressure. *(Weight: 0.02380)* 3. Congenital heart defects. Some people who develop heart failure were born with structural heart defects. *(Weight: 0.02380)*
	Text Memory	1. Race. You're at greatest risk of osteoporosis if you're white or of Asian descent. *(Weight: 0.0537)* 2. Age. The older you get, the greater your risk of osteoporosis. *(Weight: 0.0521)* 3. Inactivity. The less active you are, the greater your risk. Physical activity helps you control your weight, uses up glucose as energy and makes your cells more sensitive to insulin. *(Weight: 0.0519)*
Visit 3	Target Guidance	1. High blood pressure. Your heart works harder than it has to if your blood pressure is high. *(Weight: 0.02389)* 2. Valvular heart disease. People with valvular heart disease have a higher risk of heart failure. *(Weight: 0.02384)* 3. Heart rhythm problems. Heart rhythm problems (arrhythmias) can be a potential complication of heart failure. *(Weight: 0.02381)*
	Text Memory	1. Cardiovascular disease. Diabetes dramatically increases the risk of various cardiovascular problems, including coronary artery disease with chest pain (angina), heart attack, stroke and narrowing of arteries (atherosclerosis). If you have diabetes, you're more likely to have heart disease or stroke. *(Weight: 0.0526)* 2. Age. The older you get, the greater your risk of osteoporosis. *(Weight: 0.0525)* 3. Other chronic conditions. People with certain chronic conditions such as thyroid problems, sleep apnea, metabolic syndrome, diabetes, chronic kidney disease or lung disease have an increased risk of atrial fibrillation. *(Weight: 0.0514)*
Visit 4	Target Guidance	1. Irregular heartbeats. These abnormal rhythms, especially if they are very frequent and fast, can weaken the heart muscle and cause heart failure. *(Weight: 0.02385)* 2. Diabetes. Having diabetes increases your risk of high blood pressure and coronary artery disease. *(Weight: 0.02384)* 3. Valvular heart disease. People with valvular heart disease have a higher risk of heart failure. *(Weight: 0.02383)*
	Text Memory	1. Cardiovascular disease. Diabetes dramatically increases the risk of various cardiovascular problems, including coronary artery disease with chest pain (angina), heart attack, stroke and narrowing of arteries (atherosclerosis). If you have diabetes, you're more likely to have heart disease or stroke. **(appears twice)** *(Weight: 0.0526)* 3. Heart failure. Atrial fibrillation, especially if not controlled, may weaken the heart and lead to heart failure — a condition in which your heart can't circulate enough blood to meet your body's needs. *(Weight: 0.0524)*
Visit 5	Target Guidance	1. Irregular heartbeats. These abnormal rhythms, especially if they are very frequent and fast, can weaken the heart muscle and cause heart failure. *(Weight: 0.02382)* 2. Diabetes. Having diabetes increases your risk of high blood pressure and coronary artery disease. *(Weight: 0.02380)* 3. Coronary artery disease. Narrowed arteries may limit your heart's supply of oxygen-rich blood, resulting in weakened heart muscle. *(Weight: 0.02380)*
	Text Memory	1. Heart failure. Atrial fibrillation, especially if not controlled, may weaken the heart and lead to heart failure — a condition in which your heart can't circulate enough blood to meet your body's needs. **(appears twice)** *(Weight: 0.0519)* 2. Heart disease. Anyone with heart disease — such as heart valve problems, congenital heart disease, congestive heart failure, coronary artery disease, or a history of heart attack or heart surgery — has an increased risk of atrial fibrillation. *(Weight: 0.0516)*

Table 3.5: Case study of a positive case on the COPD dataset.

EHR	**Visit 1:** Esophageal reflux (530.81), Acute conjunctivitis (372.00), Asthma (493.90) **Visit 2:** Conjunctivitis (372.30) **Visit 3:** Other mucopurulent conjunctivitis (372.03) **Visit 4:** Lumbago (724.2), Unspecified contraceptive management (V25.9) **Visit 5:** Lumbago (724.2), Asthma (493.90), Nausea with vomiting (787.01)
Target Guidance	1. Asthma, a chronic inflammatory airway disease, may be a risk factor for developing COPD. The combination of asthma and smoking increases the risk of COPD even more. *(Weight: 0.034482)* 2. Exposure to tobacco smoke. The most significant risk factor for COPD is long-term cigarette smoking. The more years you smoke and the more packs you smoke, the greater your risk. Pipe smokers, cigar smokers and marijuana smokers also may be at risk, as well as people exposed to large amounts of secondhand smoke. *(Weight: 0.034479)*
Text Memory (Visit 1-4)	1. Proper treatment makes a big difference in preventing both short-term and long-term complications caused by asthma. *(Weight: 0.05109)* 2. Exposure to various irritants and substances that trigger allergies (allergens) can trigger signs and symptoms of asthma, including: Respiratory infections such as the common cold, Physical activity, Air pollutants and irritants such as smoke, Strong emotions and stress, Gastroesophageal reflux disease (GERD) and etc. *(Weight: 0.05107)* 3. Signs that your asthma is probably worsening include: Asthma signs and symptoms that are more frequent and bothersome, Increasing difficulty breathing, The need to use a quick-relief inhaler more often and etc. *(Weight: 0.05106)* 4. Asthma complications include: Signs and symptoms that interfere with sleep, work and other activities, Sick days from work or school during asthma flare-ups, A permanent narrowing of the tubes that carry air to and from your lungs (bronchial tubes), which affects how well you can breathe. *(Weight: 0.05105)* 5. Conditions that can increase your risk of GERD include: Obesity, Pregnancy, Connective tissue disorders, such as scleroderma and etc *(Weight: 0.05104)*
Text Memory (Visit 5)	1. Exposure to various irritants and substances that trigger allergies (allergens) can trigger signs and symptoms of asthma, including: Respiratory infections such as the common cold, Physical activity, Air pollutants and irritants such as smoke, Strong emotions and stress, Gastroesophageal reflux disease (GERD) and etc. **(appears twice)** *(Weight: 0.05015)* 3. Asthma complications include: Signs and symptoms that interfere with sleep, work and other activities, Sick days from work or school during asthma flare-ups, A permanent narrowing of the tubes that carry air to and from your lungs (bronchial tubes), which affects how well you can breathe. *(Weight: 0.05012)* 4. Asthma signs and symptoms include: Shortness of breath, Chest tightness or pain, Wheezing when exhaling, which is a common sign of asthma in children, Trouble sleeping caused by shortness of breath, coughing or wheezing, Coughing or wheezing attacks that are worsened by a respiratory virus, such as a cold or the flu. *(Weight: 0.05012)* 5. A number of factors are thought to increase your chances of developing asthma. They include: Being a smoker, Exposure to secondhand smoke, Exposure to exhaust fumes or other types of pollution and etc. *(Weight: 0.05011)*

3.5.4 Case Studies on Interpretability

Now in this section we use case studies on the Heart Failure and COPD datasets to show the good interpretability of `MedRetriever`. The first example is about a positive Heart Failure case, which is shown in Table 3.4. In this example, we show three target guidance segments selected from Γ with highest self-attention weights and three medical text segments selected from $\mathcal{P}$ stored in the text memory at each visit with highest relevance scores. Since the most prominent symptoms are *diabetes mellitus* (250.00) and *atrial fibrillation* (427.31) which appear multiple times across different visits, the most relevant target guidance segments are the ones stating that they are two key risk factors of heart failure, and the saved texts in the text memory are explaining the supporting evidence they use to make the prediction based on these two symptoms. Additionally, the relevant target guidance segments and memorized medical texts can be dynamically updated due to the condition change of the patient.

As for another case study on a positive COPD case with five visits shown in Table 3.5, due to space limit we present two target guidance segments with the highest attention weights and five medical text segments with the highest relevance scores for the first four visits and the last visit, respectively. Due to the symptom of *asthma* (493.90) of the patient, `MedRetriever` attaches the highest importance to the target disease

segment stating asthma is a risk factor for developing COPD. Since ICD codes in the following three visits does not have much relevance with the target guidance segments, `MedRetriever` mainly stores segments related to asthma in the text memory. Besides, one memorized segment is related to the gastroesophageal reflux disease due to the symptom of *esophageal reflux* (530.81). In addition, in the final time step, the originally most relevant segment is saved twice in the text memory for the symptom of *asthma* (493.90) is observed again in the fifth visit.

From these two case studies, we can observe that the text memory can be dynamically updated corresponding to the patient condition. Most importantly, patients can easily understand the relation between their symptoms and the target disease by referring to the target guidance segments, and they can read more about the explanations of their symptoms by referring to the text memory. Compared to knowledge graph, medical text does not required readers to have received much training. Hence, `MedRetriever` is able to provide more understandable interpretation for prediction.

3.6 Conclusion

Health risk prediction is an important task in the medical domain. Existing approaches either model longitudinal EHR data with deep temporal models or incorporate external information such as medical knowledge graph to enhance the diagnosis code embedding. However, ignoring the human language descriptions of diagnosis codes limits the model performance and interpretation. To solve this issue, we propose a novel framework named `MedRetriever`, which introduces the unstructured medical text as external knowledge and employs it in a target-driven manner. In particular, `MedRetriever` first learns the EHR embedding for each visit by utilizing existing EHR embedding models. After that, within each visit it conducts the preliminary retrieval according to the string similarity between diagnosis code descriptions and collected medical texts, and then retrieves relevant medical text segments by a query aggregating the EHR embedding and target guidance pool. Meanwhile, a text memory dynamically update the memorized relevant segments, and it is used with the comprehensive EHR embedding for final prediction and interpretation. Experiments results demonstrate that `MedRetriever` improves health risk prediction performance compared with existing baselines, and case studies further show that utilizing medical text can achieve understandable interpretation.

Chapter 4
Automated Medical Risk Predictive Modeling on Electronic Health Records

4.1 Introduction

Medical risk prediction is a representative task in healthcare, which aims at building actionable predictive models to forecast the future health conditions or outcomes of patients based on their historical electronic health records (EHR) [3, 44]. EHR data consist of a time-ordered sequence of visits, and each visit contains several clinical codes such as International Classification of Diseases (ICD) codes. To model such sequential characteristics of EHR data, most of existing approaches usually apply recurrent neural networks (RNN) [19, 20] and Transformer [21] as the backbone and equip advanced techniques such as attention mechanisms with them to improve the prediction performance [4, 5, 8–10, 12].

Besides, EHR data have temporal characteristics since each visit is associated with a timestamp, which is the key factor in modeling disease progression. Existing work that models the time information in the risk prediction task can be categorized into two groups. One follows the information decay assumption and uses monotonically non-increasing functions to model irregular time intervals between two consecutive visits, such as T-LSTM [22]. The other such as HiTANet [11] treats visits as words in a sentence and time stamps as words' positions and employs Transformer to model the EHR data. These approaches are powerful and effective to enhance the prediction performance, but *designing such time-aware models requires substantial efforts of human experts.* Although some automated machine learning-based frameworks are proposed in the medical domain

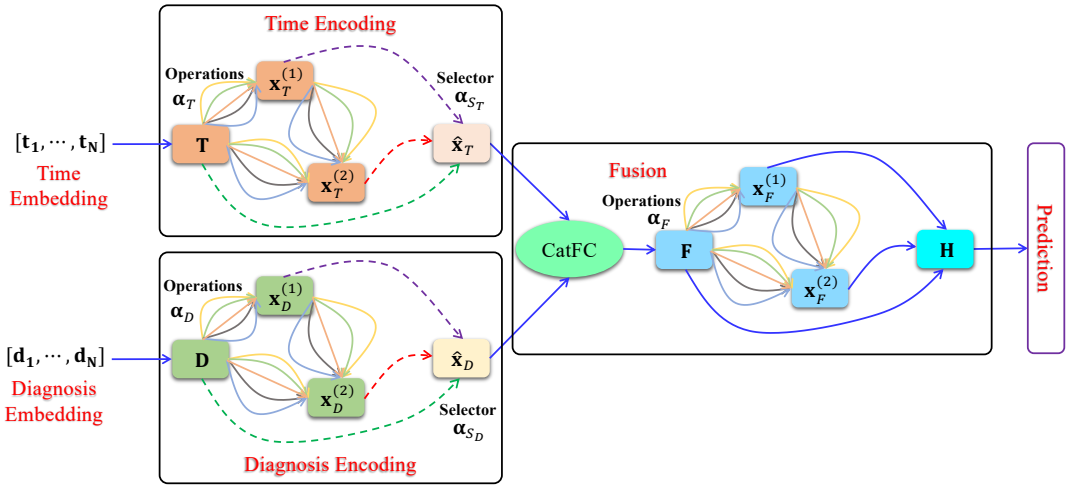


Figure 4.1: Overview of the proposed `AutoMed` in the searching stage, i.e., the supernet.

recently such as AutoPrognosis [45] and Clairvoyance [46], they mainly focus on configuring machine learning **pipelines**, instead of automatically designing network architectures. Therefore, it is an urgent need to develop new models to automatically model sequential yet temporal EHR data simultaneously with minimal human interventions.

Neural architecture search (NAS) [47–53] is a promising solution for addressing this concern that uses data-driven methods to discover the optimal architectures for various tasks, including but not limited to, computer vision [54–58] and natural language processing [59, 60]. However, it is difficult to directly apply existing frameworks to our task. The reasons are as follows: First, as we previously mentioned, EHR data are complex, which not only have sequential visits but also time stamps associated with visits. However, most existing NAS-based frameworks only consider to deal with one type of feature such as text or image. Second, even though there are several models focusing on dealing with multimodal data such as MMnas [61], MMIF [62] and BM-NAS [63], they mainly search the feature fusion strategies and *ignore the importance of aligning the inconsistency among different types of data using automatically searchable strategies*. Thus, determining how to design an effective and reasonable automated risk prediction model on EHR data is still an open challenge in the medical domain.

To tackle the aforementioned challenges, in this paper, we propose a new automated medical risk prediction model, named `AutoMed`, which can automatically search an optimal network architecture on time-ordered EHR data as shown in Figure 4.1. `AutoMed` consists of five modules: (1) The *embedding module* that maps discrete medical codes with each visit and the associated timestamp to dense embeddings $\mathbf{D}$ and $\mathbf{T}$, respectively.

(2) The *time encoding module* contains a directed acyclic graph (DAG), i.e., a cell, and a searchable feature selector. The cell can automatically search for the optimal operation between a pair of computation nodes of DAG, and the feature selector can output the representative representation $\hat{\mathbf{x}}_T$, which is taken as the input of the fusion module. (3) The *diagnosis encoding module* has the same structure as the time encoding module, and its output $\hat{\mathbf{x}}_D$ is also the input of the fusion module. (4) The *fusion module* also contains a cell to search the optimal architecture for fusing two types of features simultaneously and learning the final EHR representation $\mathbf{H}$ as the input of the risk prediction module. (5) The *prediction module* is designed to make the search stage learning more stable, which consists of an RNN layer with attention mechanisms. We use the bi-level optimization technique as DARTS [49] to jointly optimize three cells and two feature selectors and further obtain the optimal model architecture.

Our contributions can be summarized as follows:

- To the best of our knowledge, we are the first to design an NAS-based model to solve the health risk prediction problem in the medical domain, which largely reduces the human interventions of model design.
- The proposed `AutoMed` tailors a novel search space to model sequential yet temporal EHR data. Correspondingly, two separate modules are used to search the optimal architectures for discrete medical codes within each visit and the associate time information. Moreover, a fusion cell is designed to search the optimal fusion strategy. These designs make `AutoMed` learn better representation and further improve the prediction performance.
- Experimental results on three real-world claims data show that `AutoMed` achieves significant improvement over all state-of-the-art baselines, and ablation study further shows the effectiveness of all the designed modules.

4.2 Literature Review

Existing health risk prediction models are mainly to model the sequential characteristics of EHR data using RNN [19, 20] and Transformer [21] as the backbones. Then they are equipped with different types of attention mechanisms [4, 5, 10] or incorporating external knowledge such as ICD hierarchy [13, 14, 17], medical text [**?**], and medical knowledge graphs [15,16,27,64], to further improve the prediction performance. There are several approaches are proposed to model the **time information**, such as T-LSTM [22],

RetainEX [6], Timeline [7]. They mainly design the model architecture based on human prior assumptions about the effect of time information, which limits the models' learning ability. Thus, there is an urgent need of the automatic model design for health risk prediction.

Neural architecture search (NAS) [53] is a general approach for automatically discovering the optimal model architecture for deep neural networks, which is a bi-level optimization problem in essence that aims to optimize both the network parameters and the model architecture simultaneously. Some work aims to directly solve the searching problem with huge computing cost, such as using reinforcement learning [47] or evolutionary search [57]. To improve the searching efficiency of NAS methods from different perspectives, weight sharing [48], sequential model-based optimization [50], and Bayesian optimization [52] are used. More recently, differentiable architecture search (DARTS) [49] is proposed and achieves remarkable improvement in terms of searching efficiency, which introduces a continuous relaxation to the discrete model architecture and designs a unified gradient optimization framework for both the network weights and architecture. In this paper, we utilize the differentiable methods as the search algorithm and design a unified searching space for learning heterogeneous EHR features and the fusion strategy at the same time.

4.3 Methodology

4.3.1 Data & Task

The **EHR data** of each patient consists of multiple time-ordered visits $V = [(\mathbf{v}_1, t_1), (\mathbf{v}_2, t_2), \cdots, (\mathbf{v}_N, t_N)]$, where N is the total number of visits. At each visit, a set of diagnosis codes is recorded, which is represented as a binary vector $\mathbf{v}_n \in \{0, 1\}^M$, where M represents the total number of unique codes in the dataset. $\mathbf{v}_n^m = 1$ denotes that the m-th code appears in the i-th visit; otherwise, $\mathbf{v}_n^m = 0$. In addition, a timestamp in terms of date t_n is recorded at each visit. The **task** of health risk prediction is to predict whether the patient will suffer from the target disease or condition in the future according to the historical EHR data V.

4.3.2 Overview of `AutoMed`

To investigate the optimal way of integrating the heterogeneous features of EHR data, we propose `AutoMed` as shown in Figure 4.1, which contains five modules: the embedding

module, the time encoding module, the diagnosis encoding module, the fusion module, and the prediction module. The **embedding module** aims to map the input diagnosis $\mathbf{v}_n$ and time t_n features into dense vector representations $\mathbf{d}_n$ and $\mathbf{t}_n$, respectively. Then we use three modules to automatically fuse $\mathbf{d}_n$ and $\mathbf{t}_n$ following the idea of the neural architecture search (NAS) [53] to learn the optimal architectures of these three modules in a unified way. Specifically, in each module, we design a searchable cell, which shares the same search space but uses different network weights. The **time** and **diagnosis encoding modules** take $\mathbf{D} = [\mathbf{d}_1, \cdots, \mathbf{d}_N]$ and $\mathbf{T} = [\mathbf{t}_1, \cdots, \mathbf{t}_N]$ as the inputs and automatically learn a representation for the computational node in each cell, respectively. Then a searchable feature selector is developed to select optimal representations outputted by computational nodes. The selected features from the time and diagnosis encoding modules are then considered as the inputs of the **fusion module** to generate the final visit-level EHR representations, which contains a searchable cell followed by a linear combination layer (CatFC in Figure 5.1). Finally, the EHR representations are used as the inputs of the **prediction module** to make risk predictions. Next, we introduce the details of each module.

4.3.3 Embedding Diagnosis and Time Features

Diagnosis Embedding Given the binary input visit vector $\mathbf{v}_n$, we apply a linear function to transform it into a latent representation $\mathbf{d}_n \in \mathbb{R}^d$, i.e., $\mathbf{d}_n = \mathbf{W}_d\mathbf{v}_n + \mathbf{b}_d$, where $\mathbf{W}_d \in \mathbb{R}^{d\times M}$ and $\mathbf{b}_d \in \mathbb{R}^d$ is the weight matrix and bias vector, respectively. Since there are N visits in each patient's EHR data, the diagnosis features of a patient will become a sequence of representations $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_N]$.

Time Embedding Following [11], we embed the time information using the time interval Δt_n between the current time t_n and the last recorded time t_N, i.e., $\Delta t_n = t_N - t_n$, as follows:

$$\mathbf{t}_n = \mathbf{W}_t\mathbf{r}_n + \mathbf{b}_t,\ \mathbf{r}_n = \mathbf{1} - \tanh((\mathbf{W}_r\frac{\Delta t_n}{180} + \mathbf{b}_r)^2), \tag{4.1}$$

where $\mathbf{W}_r \in \mathbb{R}^a$, $\mathbf{W}_t \in \mathbb{R}^{d\times a}$, $\mathbf{b}_r \in \mathbb{R}^a$, and $\mathbf{b}_t \in \mathbb{R}^d$ are all network parameters. Similarly, the time features of the patient will be represented by a sequence of representations $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \cdots, \mathbf{t}_N]$. The *network parameters* in the embedding module are $\mathbf{W}_E = [\mathbf{W}_d, \mathbf{b}_d, \mathbf{W}_r, \mathbf{W}_t, \mathbf{b}_r, \mathbf{b}_t]$.

4.3.4 Encoding Diagnosis Representations

Cell Design Given the input diagnosis features $\mathbf{D}$, we aim to find an optimal neural architecture to encode them. In particular, following DARTS [49], we adopt the general DAG (directed acyclic graph) setting that a cell contains an ordered sequence of C computation nodes[1], where each node $\mathbf{x}_D^{(i)}$ is a latent representation, and each directed edge (i, j) is associated with some operation $o_D^{(i,j)}$ that draw from an operation set $\mathcal{O}$ to transform $\mathbf{x}_D^{(i)}$. During the search stage, each node is computed based on all of its predecessors, i.e.,

$$\mathbf{x}_D^{(j)} = \sum_{i<j} o_D^{(i,j)}(\mathbf{x}_D^{(i)}) = \sum_{i<j} \sum_{o\in\mathcal{O}} \frac{\exp(\alpha_{Do}^{(i,j)})}{\sum_{o'\in\mathcal{O}} \exp(\alpha_{Do'}^{(i,j)})} o(\mathbf{x}_D^{(i)}), \tag{4.2}$$

where $\mathbf{x}_D^{(0)} = \mathbf{D}$ and all of $\mathbf{x}$'s have the same shape as $\mathbf{D}$. The operations $\mathcal{O}$ include *1-D convolution*, *multi-head self attention*, *recurrent layer*, *feed-forward layer*, *identity*, and *zero*. The details of these operations are introduced in Section 4.4.1.3. $\alpha_{Do}^{(i,j)}$ denotes the weight of the operation o on edge (i, j) in the diagnosis encoding module.

Searchable Feature Selector. Existing methods generate the output of the diagnosis encoding module by averaging or concatenating $[\mathbf{x}_D^{(0)}, \mathbf{x}_D^{(1)}, \cdots, \mathbf{x}_D^{(C)}]$ learned by Eq. (4.2) [48, 49]. Such mandatory operations require to use all the node outputs, and the averaged or concatenated output may not be the most representative one. To avoid this issue and increase the capability of `AutoMed`, we design a searchable feature selector. Let $\alpha_{S_D}^{(k)}$ denotes the architecture weight of the k-th computation node in the cell. In the search stage, we define the mixed selection on C nodes as follows:

$$\hat{\mathbf{x}}_D = \sum_{k=0}^{C} \frac{\exp(\alpha_{S_D}^{(k)})}{\sum_{k'=0}^{C} \exp(\alpha_{S_D}^{(k')})} \mathbf{x}_D^{(k)}, \tag{4.3}$$

where $\hat{\mathbf{x}}_D$ is the output of the diagnosis encoding module. In this module, we need to optimize the *model architecture parameters*, including the operation weights $\boldsymbol{\alpha}_D$ on all edges and the selection weights $\boldsymbol{\alpha}_{S_D}$ on all computation nodes. We need to optimize the *operation parameter set* $\mathbf{W}_{O_D}$.

4.3.5 Encoding Time Representations

We apply the same cell and feature selector design introduced in the previous subsection to encode time representations. Taking $\mathbf{T}$ as the input of the time encoding module, we

[1]To reduce the computational complexity, we set $C = 2$ in this paper, i.e., three computation nodes with IDs 0, 1, and 2 in the DAG.

first obtain a list of computation node features $\{\mathbf{x}_T^{(k)}\}, k \in \{0, \cdots, C\}$ using Eq. (4.2) with operation weight parameters $\boldsymbol{\alpha}_T$. Then we generate the module output $\hat{\mathbf{x}}_T$ via Eq. (4.3) with node selection parameters $\boldsymbol{\alpha}_{S_T}$. The *operation parameters* to be optimized in this module are denoted as $\mathbf{W}_{O_T}$.

4.3.6 Fusing Diagnosis and Time Representations

After obtaining the selected features $\hat{\mathbf{x}}_D = [\hat{\mathbf{x}}_1^D, \cdots, \hat{\mathbf{x}}_N^D]$ and $\hat{\mathbf{x}}_T = [\hat{\mathbf{x}}_1^T, \cdots, \hat{\mathbf{x}}_N^T]$, we first concatenate them together and then apply a linear transformation to map the concatenation into a single feature, i.e.,

$$\mathbf{f}_n = \mathbf{W}_c \text{concat}(\hat{\mathbf{x}}_n^D, \hat{\mathbf{x}}_n^T) + \mathbf{b}_c, \tag{4.4}$$

where $\mathbf{W}_c \in \mathbb{R}^{d \times 2d}$, $\mathbf{b}_c \in \mathbb{R}^d$ are network parameters of the linear transformation layer. Then the obtained features $\mathbf{F} = [\mathbf{f}_1, \cdots, \mathbf{f}_N]$ are taken as the input of the fusion cell, which has the same design as the cells in the diagnosis and time encoding modules.

Similarly, we can obtain a list of node features as well, i.e., $\{\mathbf{x}_F^{(k)}\}, k \in \{0, \cdots, C\}$ using Eq. (4.2), when taking $\mathbf{F}$ as the input and using $\boldsymbol{\alpha}_F$ as the operation weights. However, different from the previously two encoding modules, we do not apply a feature selector here since we need to get the comprehensive representation for the whole EHR data. Thus, we combine all the node features into a single representation:

$$\mathbf{h}_n = \sum_{k=0}^{C} w_k \mathbf{x}_{Fn}^{(k)}, \tag{4.5}$$

where $w_k \in \mathbb{R}$ is the network weight parameter of the k-th computation node and $\mathbf{w}_f = [w_0, \cdots, w_C]^\top$. The output of the fusion module is $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_N]$. In this module, we need to optimize the *model architecture parameters* $\boldsymbol{\alpha}_F$ and the *network parameters* $\mathbf{W}_F = [\mathbf{W}_c, \mathbf{b}_c, \mathbf{w}_f, \mathbf{W}_{O_F}]$, where $\mathbf{W}_{O_F}$ is the operation parameter set used in the fusion cell.

4.3.7 Predicting Health Risks

To make the search stages more stable, we add a fixed RNN layer (GRU [65]) to transform the features and aggregate them over the sequence dimension through the attention mechanism and then pass the aggregated EHR representation through the final classifier

as follows:

$$
\begin{aligned}
&[\mathbf{h}'_1, \cdots, \mathbf{h}'_N] = \text{RNN}([\mathbf{h}_1, \cdots, \mathbf{h}_N]), \\
&[\beta_1, \cdots, \beta_N] = \text{softmax}(\mathbf{w}_h^\top \mathbf{h}'_1 + b_h, \cdots, \mathbf{w}_h^\top \mathbf{h}'_N + b_h), \\
&\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}_y \mathbf{u} + \mathbf{b}_y) = \text{softmax}(\mathbf{W}_y \sum_{n=1}^{N} \beta_n \mathbf{h}'_n + \mathbf{b}_y),
\end{aligned} \tag{4.6}
$$

where $\mathbf{w}_h \in \mathbb{R}^d$, $b_h \in \mathbb{R}$, $\mathbf{W}_y \in \mathbb{R}^{2\times d}$, and $\mathbf{b}_y \in \mathbb{R}^2$ are all network parameters, β's are the aggregation weights for all N time steps, and $\hat{\mathbf{y}} \in \mathbb{R}^2$ is the final output distribution. The network parameters of the prediction module are $\mathbf{W}_P = [\mathbf{w}_h, b_h, \mathbf{W}_y, \mathbf{b}_y, \mathbf{W}_{rnn}]$, where $\mathbf{W}_{rnn}$ is the parameter set of the RNN layer.

4.3.8 Optimization

Let $\boldsymbol{\alpha}$ denote the collection of architecture weights, which includes $\boldsymbol{\alpha}_T$ for the time cell, $\boldsymbol{\alpha}_D$ for the diagnosis cell, $\boldsymbol{\alpha}_F$ for the fusion cell, $\boldsymbol{\alpha}_{S_T}$ for the time feature selector, and $\boldsymbol{\alpha}_{S_D}$ for the diagnosis feature selector. Let $\mathbf{W}$ denote the network weights, which contains $\mathbf{W}_E$ for the embedding module, $\mathbf{W}_F$ for the fusion module, $\mathbf{W}_P$ for the prediction module, and $\mathbf{W}_O = [\mathbf{W}_{O_D}, \mathbf{W}_{O_T}, \mathbf{W}_{O_F}]$ for the operation parameters used in the three cells. We use the bi-level optimization technique as DARTS [49] to optimize the model architecture $\boldsymbol{\alpha}$ and the network weights $\mathbf{W}$ simultaneously:

$$
\begin{aligned}
&\min_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\mathbf{W}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \\
&\text{s.t. } \mathbf{W}^*(\boldsymbol{\alpha}) = \text{argmin}_{\mathbf{W}} \mathcal{L}_{train}(\mathbf{W}, \boldsymbol{\alpha})
\end{aligned} \tag{4.7}
$$

where $\mathcal{L}_{val}$ and $\mathcal{L}_{train}$ mean the validation loss and training loss, respectively.

4.3.9 Deriving Discrete Architectures

Using the learned architecture parameters $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_D, \boldsymbol{\alpha}_T, \boldsymbol{\alpha}_F, \boldsymbol{\alpha}_{S_D}, \boldsymbol{\alpha}_{S_T}]$, we are able to derive the discrete model architectures based on the optimal $\boldsymbol{\alpha}$ for each module. For each searched cell, based on the obtained $\alpha^{(i,j)}$ for each edge in the DAG, we can choose the optimal operation, which is $o'^{(i,j)} = \text{argmax}_{o\in\mathcal{O}} \{\alpha_o^{(i,j)}\}$. Then we compute the node feature via $\mathbf{x}^{(j)} = \sum_{i<j} o'^{(i,j)}(\mathbf{x}^{(i)})$. Also, for the feature selection in the diagnosis and time cells, we choose the node features by $\mathbf{x}'_D = \text{argmax}_{k\in\{0,\cdots,C\}} \{\alpha_{S_D}^{(k)}\}$ and $\mathbf{x}'_T = \text{argmax}_{k\in\{0,\cdots,C\}} \{\alpha_{S_T}^{(k)}\}$. Eventually, we can derive the final model architecture and train the model from scratch for evaluation.

Table 4.1: Statistics of the four EHR datasets.

Dataset	COPD	Amnesia	Kidney	HF
Positive Cases	7,314	2,982	2,810	3,080
Negative Cases	21,942	8,946	8,430	9,240
Avg. Visits/Patient	30.39	39.00	39.09	38.74
Avg. Codes/Visit	3.50	4.70	4.40	4.24
Unique Codes	10,053	9,032	8,802	8,692

4.3.10 Complexity Analysis

In this section, we conduct the complexity analysis for the proposed `AutoMed`. Based on the used search space, we can analyze the number of possible architectures that `AutoMed` is able to search. For each designed cell, there are total $\prod_{k=1}^{C} |\mathcal{O}|^k$ possible DAGs without considering graph isomorphism. Then, for all three cells, we will have $\Big(\prod_{k=1}^{C} |\mathcal{O}|^k\Big)^3$ possible architectures. In addition, our proposed feature selector offers C^2 combinations. Thus, the final complexity of the search space will become $C^2\Big(\prod_{k=1}^{C} |\mathcal{O}|^k\Big)^3$, which is sufficiently large to search for optimal model architectures for risk prediction task.

4.4 Experiments

4.4.1 Experimental Setup

4.4.1.1 Datasets

In our experiments, we conduct retrospective analysis on four common chronic and progressive health conditions, which are Chronic Obstructive Pulmonary Disease (COPD), Amnesia, Kidney Disease, and Heart Failure (HF). With the guidance of clinicians, we extract the corresponding EHR data, which includes positive cases and negative/control cases, from a real-world claims database. When extracting positive cases, we identify *the first disease diagnosis date* and then only keep the EHR data within six months before that date. For each positive case, we extract at most three control cases based on gender, race, age group, and underlying diseases. We keep the whole EHR data for negative/control cases. In our experiments, we only the last 50 visits as the input data. The statistics of the datasets are shown in Table 4.1.

We randomly partition the datasets into the training set, validation set, and testing set with the ratio of $0.75 : 0.10 : 0.15$. The best model is selected based on the performance on

the validation set. We report the average results of five runs for performance evaluation.

4.4.1.2 Baselines

We select traditional and state-of-the-art risk prediction models as our baselines, which are divided into two categories: (1) Without using time information: LSTM [20], Dipole [5], Retain [4], SAnD [10], AdaCare [8], LSAN [12]. (2) Using time information: RetainEx [6], Timeline [7], T-LSTM [22], HiTANet [11].

4.4.1.3 Operations

In this paper, we use the following six operations when searching the network architectures: 1-D Convolution (*conv*), Multi-head Self Attention (*attention*), Recurrent Layer (*rnn*), Feed-Forward Layer (*ffn*), Identity and Zero.

- **1-D Convolution** (*conv*). We use the convolution layer as a candidate operation that performs 1-D convolution with respect to the sequence dimension of the input tensor. In addition, we utilize the SAME padding method to maintain the output size. This operation can capture the interaction patterns of local features among neighboring time steps.

- **Multi-head Self Attention** (*attention*). Multi-head self-attention layer is a major component in Transformer [21] that is particularly suitable for sequence modeling. Thus, we incorporate it into the candidate operation set. Different from the convolution operation, self-attention could capture the long-term dependencies of the sequence effectively.

- **Recurrent Layer** (*rnn*). Recurrent neural network (RNN) is also a powerful architecture for processing sequence data. LSTM [20] and GRU [19] are known to be more advantageous than the vanilla RNN for capturing long-term dependencies of sequences; while GRU is usually several times faster than LSTM without loss of precision. Therefore, we leverage the GRU layer as the recurrent layer operation.

- **Feed-Forward Layer** (*ffn*). We also define feed-forward operation that applies independently to each time step of the input. Specifically, we apply two linear transformations with a ReLU activation in between, which can increase the model representation power if necessary.

- **Identity & Zero**. Except for previous operations, we also include Identity and Zero operations into the candidate set, which allows the model for discovering skip connections and discarding certain features if necessary.

4.4.1.4 Implementation Details

During the searching stage, we set different optimization configurations for the architecture weights $\boldsymbol{\alpha}$ and network weights $\mathbf{W}$. For both of them, we apply Adam optimizer [66]. For $\mathbf{W}$, we use the learning rate of 10^{-4} and weight decay of 10^{-4}, while for $\boldsymbol{\alpha}$, we use the learning rate of 10^{-5} and weight decay of 10^{-4}. We tune the learning rate and weight decay from a candidate set of $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$. Through grid search method, we obtain the most suitable values and use them in the experiments. After searching, we train the model from scratch with the derived architecture, and we also apply the Adam optimizer with learning rate of 10^{-4} and weight decay of 10^{-4}, which is tuned in the same way as aforementioned. Besides, the hidden dimension size d of all the node features within our framework is set to 256, and the dimension of the intermediate time encoding a is set to 64. The setting of these dimensions maintains the same during both searching and training stages. We implement the **baselines** on the same platform with the proposed model and apply the same optimization settings as training the searched architecture. We use the standard cross-entropy loss for all baselines. The numbers of hidden dimensions of baselines are all 256 no matter for RNN or Transformer based models. We use *PR-AUC* (area under the precision-recall curve), *F1 score*, and *Cohen's Kappa* as the evaluation metrics considering the imbalanced data property in our datasets shown in Table 4.1.

4.4.2 Performance Evaluation

Table 4.2 shows the overall performance of the proposed `AutoMed` and baselines on four datasets. We report the **average values** of five runs and the corresponding standard deviations (***std.***). We also conduct significance testing (**t-test**) to justify whether the proposed `AutoMed` is significantly better than the best baseline model.

From Table 4.2, we can observe that the baselines incorporating time information usually perform better than those without considering the importance of time information. Especially, time-aware LSTM (T-LSTM) [22] that uses an information decay function to model the time information in the LSTM cell achieves the best PR-AUC score on the COPD dataset among all the baselines. HiTANet [11] takes the time information as

Table 4.2: Performance comparison in terms of PR-AUC, F1 score, and Cohen's Kappa (*mean*±*std.*). The results produced by the best baseline and the best model in each column are marked by underlined and **boldfaced**, respectively. $*$ denotes that the p-value is smaller than 0.01.

Dataset		COPD			Amnesia		
Metrics		PR-AUC (%)	F1 (%)	Kappa (%)	PR-AUC (%)	F1 (%)	Kappa (%)
Without Time	LSTM [20]	55.34 ± 3.05	55.96 ± 0.97	41.78 ± 1.13	55.36 ± 3.35	61.14 ± 0.50	48.38 ± 1.39
	Dipole [5]	58.70 ± 1.19	56.18 ± 1.29	42.18 ± 1.44	58.04 ± 1.77	60.16 ± 2.84	46.46 ± 3.39
	Retain [4]	53.56 ± 0.69	50.96 ± 0.65	37.46 ± 0.80	56.04 ± 3.20	55.06 ± 1.52	43.48 ± 1.88
	SAnD [10]	51.70 ± 2.27	52.12 ± 2.36	37.66 ± 2.36	52.50 ± 4.98	56.38 ± 2.81	41.68 ± 2.70
	Adacare [8]	60.50 ± 1.61	55.08 ± 0.36	42.34 ± 0.85	59.68 ± 2.10	60.68 ± 1.21	47.84 ± 2.87
	LSAN [12]	63.84 ± 1.75	54.98 ± 0.98	43.52 ± 0.88	68.16 ± 1.52	64.12 ± 1.64	52.88 ± 1.87
With Time	RetainEx [6]	60.52 ± 0.61	54.04 ± 2.69	43.44 ± 2.55	63.44 ± 1.92	58.92 ± 4.00	49.06 ± 4.27
	Timeline [7]	54.86 ± 1.85	49.02 ± 0.85	36.40 ± 1.10	56.46 ± 2.52	58.24 ± 2.04	45.52 ± 2.48
	T-LSTM [22]	68.62 ± 0.80	62.92 ± 0.61	51.55 ± 1.06	63.19 ± 2.14	62.91 ± 0.83	51.08 ± 1.62
	HiTANet [11]	68.46 ± 0.44	61.86 ± 0.80	49.80 ± 0.57	70.80 ± 0.96	64.06 ± 1.87	53.28 ± 2.18
	`AutoMed`	**71.57***± 2.48	**65.08***± 2.13	**54.34***± 1.86	**73.13***± 2.58	**68.91***± 1.73	**58.42***± 2.60
Dataset		Kidney			Heart Failure		
Metrics		PR-AUC (%)	F1 (%)	Kappa (%)	PR-AUC (%)	F1 (%)	Kappa (%)
Without Time	LSTM [20]	61.96 ± 2.49	63.69 ± 1.18	50.36 ± 1.78	54.98 ± 1.97	59.58 ± 0.99	43.92 ± 0.74
	Dipole [5]	64.88 ± 3.35	64.65 ± 1.74	51.60 ± 2.22	56.80 ± 2.12	58.84 ± 1.32	43.02 ± 1.16
	Retain [4]	61.72 ± 2.76	57.15 ± 2.40	44.61 ± 2.83	53.90 ± 1.45	49.96 ± 2.69	34.88 ± 2.73
	SAnD [10]	57.69 ± 3.21	60.36 ± 1.06	45.75 ± 1.10	53.74 ± 3.41	55.38 ± 1.05	39.42 ± 1.88
	Adacare [8]	71.29 ± 2.46	65.01 ± 1.49	52.89 ± 2.29	60.74 ± 4.41	57.46 ± 3.28	42.98 ± 3.95
	LSAN [12]	72.31 ± 0.63	64.44 ± 1.43	52.48 ± 1.87	**68.24** ± 1.47	60.10 ± 1.13	46.38 ± 1.02
With Time	RetainEx [6]	69.15 ± 1.48	61.61 ± 1.48	50.61 ± 1.65	62.46 ± 1.09	54.06 ± 2.61	41.08 ± 2.82
	Timeline [7]	63.89 ± 3.12	59.87 ± 1.18	46.71 ± 1.12	60.18 ± 3.12	57.08 ± 2.44	42.10 ± 3.27
	T-LSTM [22]	68.90 ± 3.29	66.16 ± 0.61	54.26 ± 0.73	64.66 ± 4.02	62.40 ± 2.33	48.89 ± 2.86
	HiTANet [11]	75.65 ± 0.44	68.01 ± 0.74	56.72 ± 0.81	67.56 ± 2.01	61.92 ± 1.49	47.92 ± 1.46
	`AutoMed`	**76.63***± 1.83	**70.41***± 1.26	**59.13***± 2.23	67.07*± 1.99	**66.64***± 1.36	**53.10***± 1.81

word positions in Transformer and achieves the best performance on all three datasets. These two kinds of approaches are representative in the health risk prediction task when modeling time information.

Although existing approaches can improve the prediction performance by modeling time information via human prior knowledge, they all entangle the time features with diagnosis features during the model architecture design. Since two different features have inconsistent patterns and scales, it is extremely difficult for human-designed architecture to fuse them together appropriately. Thus, our proposed `AutoMed` uses disentangled cells to process each type of features independently and designs a fusion cell to automatically search the feature fusion strategy, which can solve the feature inconsistency problem better. In such a way, the proposed `AutoMed` significantly outperforms all the baselines in terms of PR-AUC, F1, and Cohen's Kappa.

4.4.3 Ablation Study

The benefit of the proposed `AutoMed` is to automatically discover optimal network architectures via the three designed cells. Next, an ablation study is conducted to investigate the performance change when we add the cells one by one. Besides, for both the diagnosis and time encoding modules, we use a searchable feature selector to automatically learn the representative module outputs. To validate the efficiency of the proposed feature selector, we also conduct an ablation study. Specifically, we design the following four settings:

- **Fusion Only**: In this setting, we do not use the diagnosis and time modules and only use the fusion module. We achieve this by replacing $\hat{\mathbf{x}}_T$ and $\hat{\mathbf{x}}_D$ in Eq. (4.4) with $\mathbf{T}$ and $\mathbf{D}$ obtained in the embedding module.
- **Fusion+Time**: We use two searchable cells in this setting, i.e., the fusion and time cells. Towards this end, we replace $\hat{\mathbf{x}}_D$ with $\mathbf{D}$ in Eq. (4.4). In the time encoding module, we use the searchable feature selector.
- **Fusion+Diagnosis**: Similar to the above ablation setting, we replace $\hat{\mathbf{x}}_T$ with $\mathbf{T}$ in Eq. (4.4). In the diagnosis encoding module, we also use the searchable feature selector.
- **W.O. Selectors**: This setting means that `AutoMed` removes the feature selector for the diagnosis and time encoding modules (i.e., without using Eq. (4.3)) and uses the average computation node representations as the outputs of these modules, i.e., $\hat{\mathbf{x}}_D = \frac{1}{C}\sum_{k=0}^{C}\mathbf{x}_D^{(k)}$ and $\hat{\mathbf{x}}_T = \frac{1}{C}\sum_{k=0}^{C}\mathbf{x}_T^{(k)}$.

Table 4.3: Ablation study results in terms of F1 score (%).

Dataset	COPD	Amnesia	Kidney	HF
`AutoMed`	65.08	**68.91**	**70.41**	**66.64**
Fusion Only	62.32	64.75	69.42	62.76
Fusion+Time	62.81	68.79	69.02	63.64
Fusion+Diagnosis	61.90	63.95	69.31	64.75
W.O. Selectors	**65.90**	66.00	69.35	64.57

We present the ablation study results in Table 4.3 in terms of F1 score (%). Note that the results of the other two metrics have similar patterns as those of F1 scores. We

can observe that removing any of the cells will lead to performance drop to some degree, which can validate that it is necessary to design three cells to jointly learn the optimal model architecture for risk prediction. Additionally, the contribution of each cell varies on different datasets.

Compared to Fusion Only, `AutoMed` designs separate cells for each type of feature, which enables the search algorithm to find the best model architecture for each one of them. Thus, `AutoMed` can largely improve the model learning ability on heterogeneous EHR data. Another noteworthy thing is that it would lead to performance drop compared to single-cell search when adding the diagnosis cell. This indicates that simply searching for one type of feature might lead to the inconsistency of time and diagnosis features, which affects the learning of the fusion cell. Therefore, it is optimal to design both time and diagnosis cells and combine them with the fusion cell to learn the overall model architecture simultaneously.

When we use the average representations of nodes in each cell as the output of the encoding modules (i.e., W.O. Selectors), we can find that on the COPD dataset, the F1 score sightly increases. For other three datasets, the performance of W.O. Selectors is worse than that of `AutoMed`. These results demonstrate that using the designed searchable feature selector does not harm the model performance, and in turn, it can boost the performance in most cases.

4.4.4 Demonstration of the Searched Architectures

Figure 4.2 shows the searched architectures on the four datasets by the proposed `AutoMed` from a certain run. Due to the randomness, different runs may obtain different searched architectures, but in our experiments, we observe that they usually have similar architectures. From these searched architectures, we can derive several insights for the risk prediction model design as follows:

- **Time Cell**. Convolution operations are often selected for the time cell, which indicates that neighboring visits' time features are typically crucial in the design of model architectures since simply analyzing the time information for a single step does not capture the disease progression through time.

- **Diagnosis Cell**. For diagnosis features, `AutoMed` might choose different feature processing methods such as the feed-forward layer, attention, or convolution operations based on the unique patterns of each dataset. This indicates that for diagnosis features, there is no one optimal operation for all the datasets, which leads to the

issue that we need to specify architectures for different datasets. However, this issue can be avoided by the proposed `AutoMed`.

- **Fusion Cell**. After combining two types of features together, the fusion cell typically chooses RNN models for processing the fused features, which indicates that the RNN model is an optimal choice of comprehensively analyzing the heterogeneous features for risk prediction task.

4.4.5 Varying # of Cell Nodes

In this subsection, we conduct experiments to validate whether the number of step nodes within each cell affects the effectiveness of different model capacities. Towards this end, we run experiments under different configurations of C's and report the validation error curves of the supernet during the search stage. Since we have a unified setting for all three designed cells, the change of C affects all cells.

Figure 4.3 shows the results. We can observe when C is equal to 1, the validation loss typically can not reach the lowest value during the searching stage since it has too limited model capacity to fit the EHR data well. When C increases to 4 or 5, the searching loss curve will become less stable which may lead to poor performance since too large model capacity might overfit the data. Therefore, we recommend the selection of C to be 2 or 3 on these datasets.

4.4.6 Additional Results on Dementia Dataset

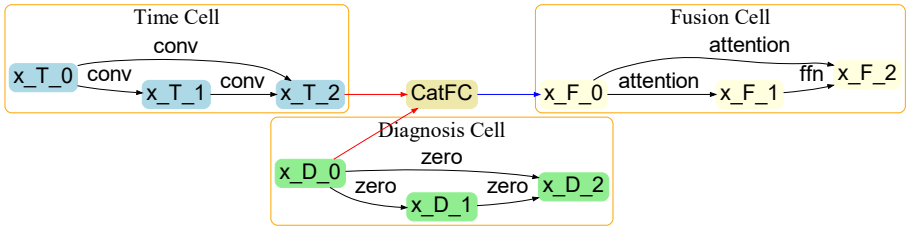
In this section, we present the results of the fifth EHR dataset, which is Dementia. We use the same setting for this dataset as other four datasets shown in the paper and conduct all of the evaluation methods on this dataset, including performance evaluation in Table 4.5, ablation study in Table 4.6, analyzing the searched architecture in Figure 4.4, and varying number of cell nodes in Figure 4.5.

4.4.6.1 Dataset

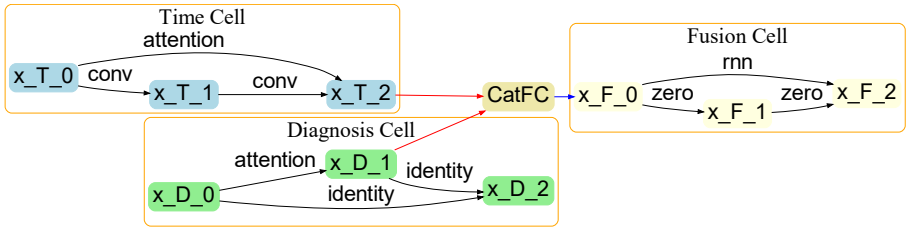
The statistics of Dementia dataset are shown in Table 4.4.

4.4.6.2 Performance Evaluation

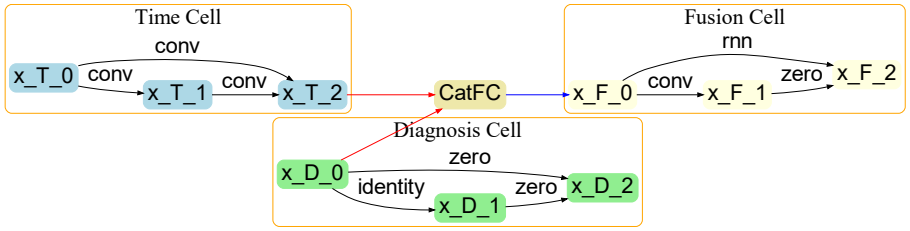
As shown in Table 4.5, the performance of `AutoMed` on Dementia is still higher than all of the baselines for most metrics, which shows that the proposed `AutoMed` achieves stable



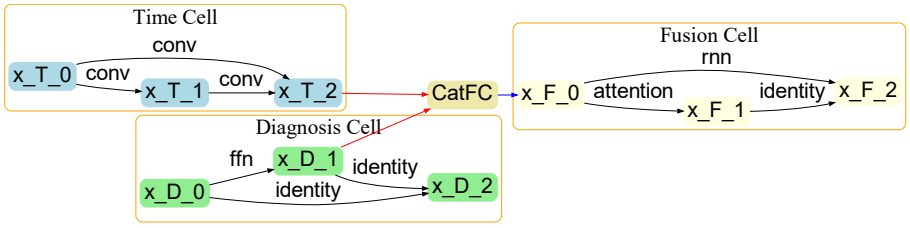
(a) COPD



(b) Amnesia



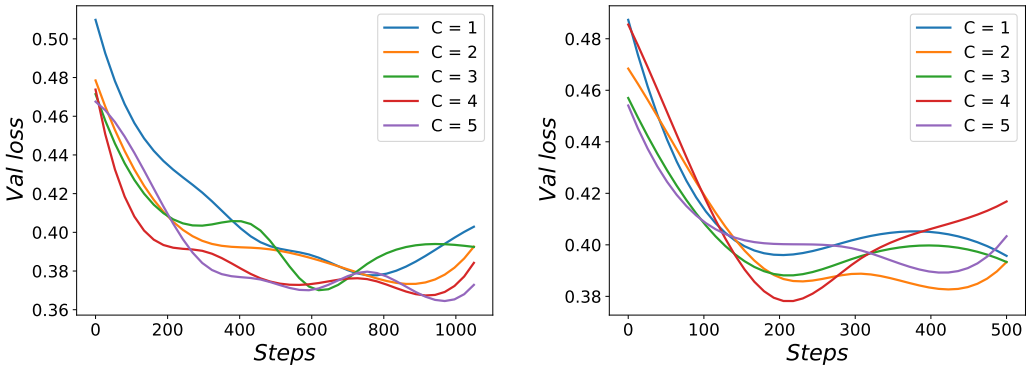
(c) Kidney



(d) Heart Failure

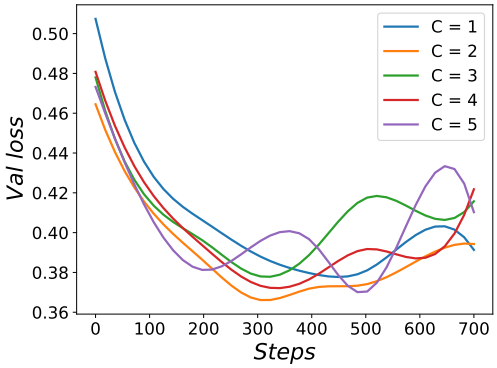
Figure 4.2: The searched architectures. The black arrows (→) denote the searched operation on the edges of DAG. The red arrows (→) mean the selected computation node by the searchable feature selectors. The blue arrows (→) represent the input of the fusion cells, which are not searched by `AutoMed`. "x_T_0" means $\mathbf{T}$, and "x_D_0" means $\mathbf{D}$. Similarly, "x_D_1" is $\mathbf{x}_D^{(1)}$ learned by Eq. (4.2).

improvements over the state-of-art risk prediction models.

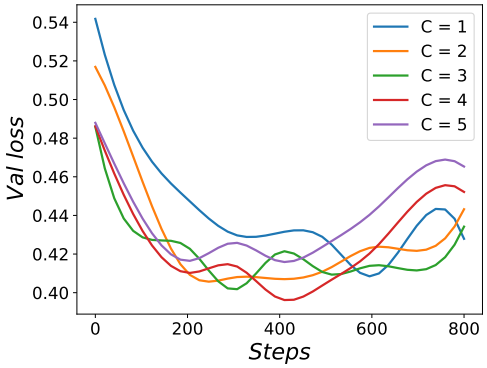


(a) COPD

(b) Amnesia



(c) Kidney Disease



(d) Heart Failure

Figure 4.3: The validation loss curves on four datasets when trying different number of step nodes.

Table 4.4: Statistics of Dementia dataset.

Dataset	Dementia
Positive Cases	2,385
Negative Cases	7,155
Avg. Visits/Patient	41.05
Avg. Codes/Visit	4.71
Unique Codes	7,813

4.4.6.3 Ablation Study

As shown in Table 4.6, the ablation study results on the Dementia dataset also show that removing any of the modules from the original `AutoMed` will lead to performance decrease to some degree. This again verifies that all of the components designed for `AutoMed` are necessary.

Table 4.5: Performance comparison in terms of PR-AUC, F1 score, and Cohen's Kappa on the Dementia dataset. The results produced by the best baseline and the best performer in each column are marked with underlined and **boldfaced**, respectively.

Dataset	Dementia		
Metrics	PR-AUC	F1	Kappa
LSTM	56.76 ± 3.25	56.46 ± 1.47	42.98 ± 1.58
Dipole	56.80 ± 3.92	56.07 ± 2.25	42.34 ± 3.31
Retain	60.64 ± 1.74	52.79 ± 1.73	41.36 ± 1.94
SAnD	51.38 ± 3.18	50.47 ± 3.03	33.42 ± 4.34
Adacare	61.81 ± 1.50	57.81 ± 2.12	44.72 ± 3.20
LSAN	**64.86** ± 1.01	57.35 ± 2.52	44.90 ± 2.23
RetainEx	61.94 ± 1.18	54.29 ± 0.44	42.58 ± 0.54
Timeline	59.01 ± 0.85	55.24 ± 1.65	40.99 ± 1.61
TLSTM	56.82 ± 5.62	56.00 ± 1.82	41.67 ± 2.72
HiTANet	59.64 ± 1.85	55.75 ± 0.94	42.04 ± 0.87
`AutoMed`	61.35*± 1.91	**58.98***± 2.83	**45.53***± 2.39

Table 4.6: Ablation study results in terms of F1 on the Dementia dataset.

Dataset	Dementia
`AutoMed`	**58.98**
Fusion Only	58.73
Fusion + Time	58.46
Fusion + Diagnosis	57.14
W.O. Selectors	57.67

4.4.6.4 Searched Architecture

We demonstrate the searched architecture on the Dementia dataset in Figure 4.4. Similarly, `AutoMed` searches for convolution operations to process time features. Differently from the searched architectures shown in Figure 4.2, the convolution operation is chosen to process diagnosis features, and attention combined with feed forward layer is chosen to handle the fused features, which is determined by the unique characteristics of the Dementia dataset.

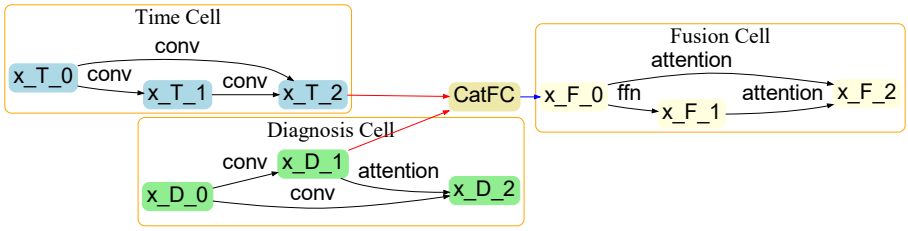


Figure 4.4: The searched architectures on Dementia dataset. The black arrows (→) denote the searched operation on the edges of DAG. The red arrows (→) mean the selected computation node by the searchable feature selectors. The blue arrows (→) represent the input of the fusion cells, which are not searched by `AutoMed`.

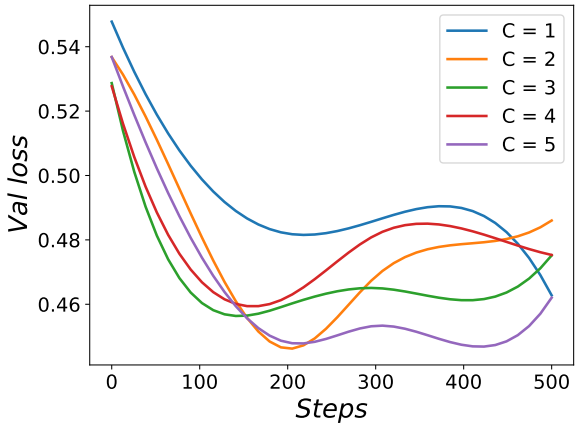


Figure 4.5: The validation loss curves on the Dementia dataset when trying different numbers of step nodes.

4.4.6.5 Varying # of Cell Nodes

The results of varying number of cell nodes for the Dementia dataset are shown in Figure 4.5. We can observe that the larger number of nodes is typically better for the Dementia dataset. However, the proposed `AutoMed` using $C = 2$ still achieves better performance on this dataset as shown in Table 4.5.

4.5 Conclusion

In this paper, we propose a novel automated risk predictive modeling approach, named `AutoMed`, which is able to automatically search the optimal model architecture for dealing with the sequential and temporal EHR data with minimal human interventions. The designed model consists of five modules, and they tightly work together to optimize

not only model architecture parameters and generate the optimal network architecture. Experiments on three real-world medical datasets show that the proposed `AutoMed` achieves state-of-the-art performance compared with baselines. Moreover, the ablation study demonstrates the effectiveness of the designed modules, and the case study of presenting searched architectures offers some important insights, which are helpful for the future model design. Our future work will explore how to incorporate medical knowledge graphs into automated risk predictive modeling design.

Chapter 5
Automated Fusion of Multimodal Electronic Health Records for Better Medical Predictions

5.1 Introduction

Electronic Health Record (EHR) systems have been extensively adopted in numerous hospitals and healthcare institutions, resulting in the generation of vast amounts of patient EHR data on a daily basis. This data holds significant potential for various predictive tasks, including but not limited to diagnosis prediction [5, 67], medical recommendation [40], health risk prediction [4, 16], and hospital readmission [68]. However, the heterogeneous and multimodal nature of EHR data poses significant challenges in the design of effective deep predictive models.

Most existing studies [69–72] primarily focus on the design of *hand-crafted* model architectures for integrating multimodal EHR data. However, this approach necessitates considerable domain expertise, which may introduce the potential for human bias. To address these concerns, recent research [73, 74] has proposed the utilization of neural architecture search (NAS) techniques [53] to automatically search for suitable architectures for modeling multimodal EHR data, thereby eliminate the need for human intervention. Although these methods demonstrate improved performance compared to hand-crafted approaches, they still encounter certain issues:

C1 – Diversifying the Search Space Current approaches [73, 74] typically employ a uniform search space across all modalities when conducting neural architecture search (NAS). However, EHR data encompasses diverse modalities, including tabular

demographics, discrete medical codes, continuous monitoring data from ICU stays, and unstructured clinical notes. A uniform search space fails to adequately capture the heterogeneity of these modalities. Furthermore, existing methods primarily utilize simple operations like concatenation and addition within the fusion search space, which may not effectively capture the complex interactions among different EHR modalities. Therefore, it is crucial to explore more suitable search spaces for multimodal feature encoding and fusion that can better accommodate the diverse nature of EHR modalities.

C2 – Customizing the Search Optimization Existing approaches utilize either evolutionary NAS [57] or differentiable architecture search (DARTS) [49] to find the optimal architecture within the defined search space. Although DARTS represents a significant efficiency improvement compared to evolutionary NAS, it often encounters issues of robustness [75] and may struggle to identify suitable architectures. Given the complexity of the search space in our specific task, directly applying DARTS may result in poor performance for the searched architectures. Consequently, a customized search algorithm is necessary to discover meaningful architectures for effectively fusing multimodal EHR data.

C3 – Deriving the Optimal Architecture DARTS-based methods, such as AutoMed [74], determine the final architecture by selecting operations based on the magnitudes of architecture weights obtained from the trained supernet. However, it has been observed that the operation with the highest architectural weight on the supernet does not necessarily correspond to its actual contribution to the generalization performance [76]. This arbitrary discretization approach can result in a significant drop in performance, a concern that becomes more pronounced when modeling multimodal EHR data due to the complexity of the search space. Thus, the development of a new discretization technique tailored to our specific task is necessary to derive an optimal architecture.

Our Approach To overcome the aforementioned challenges, we propose a novel neural architecture search (NAS) framework called `AutoFM` for **Auto**matically **F**using **M**ulti-modal EHR data. `AutoFM` addresses **C1** by introducing a new two-stage search space. The first stage, called *modality-specific search*, focuses on designing specialized encoding modules for each input modality, incorporating feature encoding and feature interaction operations to explore potential early fusion strategies. The second stage, known as *multi-modal fusion search*, focuses on creating late fusion architectures. This involves a feature selector component to determine the selected modalities and a searchable fusion component to identify the optimal fusion operation.

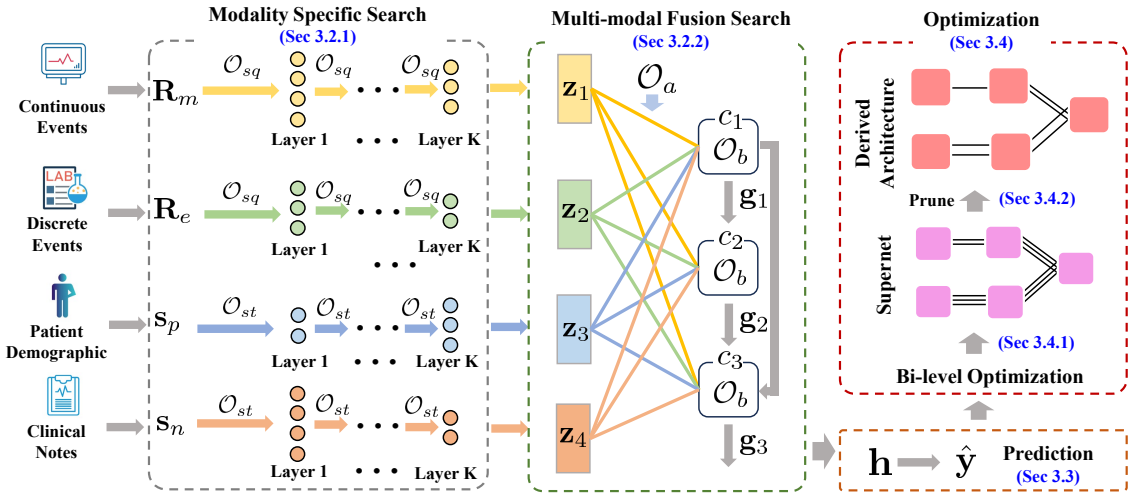


Figure 5.1: Overview of the proposed `AutoFM`.

To tackle **C2**, we introduce a customized loss term within the original bi-level optimization formulation of DARTS. This customized loss promotes diversity within the feature selector, guiding the search algorithm toward discovering more meaningful architectures. To address **C3**, we devise a novel pruning-based algorithm to select the optimal architectures after training the supernet. This algorithm effectively preserves the performance of the derived architecture during the discretization process. We extensively evaluate our proposed method on real-world multi-modal EHR data, showing its superiority over existing state-of-the-art models. Through these experiments, we demonstrate the effectiveness and advantages of our approach.

5.2 Literature Review

5.2.1 Modeling Multi-modal EHR data

Recently, clinicians and researchers have begun to leverage multi-modal EHR data to improve the performance of predictive modeling for healthcare.

For example, Raim [69] and DCMN [70] use both continuous patient monitoring data, such as electrocardiograms, and discrete clinical events to better forecast the length of ICU stays and mortality. MNN [71] improves diagnosis prediction by combining medical codes and clinical notes through a multi-modal attentional neural network. [72] combines clinical notes with patient monitoring data to conduct risk prediction of acute respiratory failure (ARF) and diagnosis prediction. And [77] further improves the mortality and

phenotype prediction performance by modeling the irregularity of clinical events and clinical notes during ICU stays. However, all of the existing works focus on designing hand-crafted neural networks to model the multi-modal EHR data. Therefore, they are limited to predefined input modalities and prediction tasks. Compared to the hand-crafted models, MUFASA [73] applies the evolutionary architecture search method to search for the multi-modal network for diagnosis prediction. However, it comes with high computational cost and have constraints on input features and fusion strategies.

Our work [78] distinguishes itself by introducing a comprehensive search space encompassing modality-specific architectures and diverse multi-modal fusion strategies. Besides, we propose a new optimization loss on top of the more efficient DARTS [49] algorithm, enabling the discovery of more reasonable and advantageous architectures for multi-modal EHR data.

5.2.2 Neural Architecture Search

Neural architecture search (NAS) [53] automates the process of finding optimal deep neural network architectures. Different approaches such as reinforcement learning [47] and evolutionary search [57] have been used, but they require significant computational costs. To improve efficiency, more advanced techniques have been employed, such as weight sharing [48], sequential model-based optimization [50], and Bayesian optimization [52]. A recent approach called differentiable architecture search (DARTS) [49] has emerged, which achieves high search efficiency by using a continuous relaxation of the discrete architecture. However, DARTS has faced challenges regarding robustness and generalization. For instance, [75] observes that DARTS often fails to select meaningful operations and degenerates to networks filled with parameter-free operations, which leads to poor performance. Also, [76] shows that selecting the optimal architectures based on the magnitude of architecture weights in DARTS is not reasonable since it does not necessarily correlate to the final performance, which leads to a performance drop when discretizing the supernet. Our work [78] addresses the limitations of DARTS when applied to multi-modal EHR data. It introduces an additional penalty during supernet training to encourage the selection of more meaningful architectures. Additionally, a new pruning-based architecture selection method is proposed to maintain the supernet's performance during discretization.

5.3 Methodology

As depicted in Figure 5.1, our proposed `AutoFM` framework takes multiple heterogeneous EHR data as input. To effectively handle this diverse data, we introduce a novel multimodal search space that enables the automatic search for optimal architectures for modality representation and fusion strategies. This search space allows our framework to discover the most suitable approaches for representing each modality and effectively fuse them together for improved predictive modeling.

5.3.1 Multimodal EHR Data Embedding

Our model takes four modalities as the input, including two types of **sequential** data – (1) continuous events $\mathbf{M} = [\mathbf{m}_1, \cdots, \mathbf{m}_T] \in \mathbb{R}^{d_1 \times T}$ and (2) discrete events $\mathbf{E} = [\mathbf{e}_1, \cdots, \mathbf{e}_T] \in \mathbb{R}^{d_2 \times T}$ and two types of **static** data – (3) patient demographics $\mathbf{p} \in \mathbb{R}^{d_3}$ and (4) the corresponding clinical note $\mathbf{n} \in \mathbb{R}^{d_4}$, where T is the number of time slots and $\mathbf{n}$ is the output of *[CLS]* from the pre-trained language model - ClinicalBERT [68]. The details of data processing can be found in Section 5.4.1.

We first map these features into the same latent space by a fully connected layer:

$$\mathbf{R}_m = \mathbf{W}_m^\top [\mathbf{m}_1, \cdots, \mathbf{m}_T] + \mathbf{b}_m, \tag{5.1}$$

$$\mathbf{R}_e = \mathbf{W}_e^\top [\mathbf{e}_1, \cdots, \mathbf{e}_T] + \mathbf{b}_e, \tag{5.2}$$

$$\mathbf{s}_p = \mathbf{W}_p^\top \mathbf{p} + \mathbf{b}_p, \tag{5.3}$$

$$\mathbf{s}_n = \mathbf{W}_n^\top \mathbf{n} + \mathbf{b}_n, \tag{5.4}$$

where $\mathbf{W}_m \in \mathbb{R}^{d_1 \times d_e}$, $\mathbf{W}_e \in \mathbb{R}^{d_2 \times d_e}$, $\mathbf{W}_p \in \mathbb{R}^{d_3 \times d_e}$, $\mathbf{W}_n \in \mathbb{R}^{d_4 \times d_e}$, $\mathbf{b}_e \in \mathbb{R}^{d_e}$, $\mathbf{b}_m \in \mathbb{R}^{d_e}$, $\mathbf{b}_p \in \mathbb{R}^{d_e}$, and $\mathbf{b}_n \in \mathbb{R}^{d_e}$ are learnable parameters.

5.3.2 Multi-Modal Search Space Design

Our approach incorporates two stages of searchable modules: (1) **modality-specific search** and (2) **multimodal fusion search**. In the first stage, we focus on designing modality-specific search spaces tailored to each individual modality. These search spaces take into account not only the encoding of the current feature within a modality but also the potential early interactions between different modalities. This allows for the exploration of various encoding strategies and fusion techniques at the modality level.

Table 5.1: Modality-specific operations.

Feature	Operation	Operation Description
Static ($\mathcal{O}_{st}$)	Identity	This is a default operation that does not transform the input feature.
	Linear Layer	Apply linear transformation with ReLU activation, where $\mathbf{W}_1 \in \mathbb{R}^{d_e \times d_e}$ and $\mathbf{b}_1 \in \mathbb{R}^{d_e}$: $o(x) = \text{ReLU}(\mathbf{W}_1 x + \mathbf{b}_1)$
	Static-Static Feature Interaction	Apply concatenation and feed-forward layer to model the interaction between the current feature and the other static feature, where $\mathbf{W}_2 \in \mathbb{R}^{2d_e \times d_e}$, $\mathbf{b}_2 \in \mathbb{R}^{d_e}$ and x' is the input feature of the other static modality $\mathbf{s}_p$ or $\mathbf{s}_n$: $o(x) = \mathbf{W}_2[x; x'] + \mathbf{b}_2$
	Static-Sequential Feature Interaction	Apply the attention mechanism to model the interaction with the current feature as the query and the other features as the keys. Since we have two sequence features, we then have two interaction operations: $o(x) = \text{Softmax}\left(\frac{(\mathbf{W}_q x)^\top \cdot (\mathbf{W}_k x')}{\sqrt{d_e}}\right) \cdot (\mathbf{W}_v x'), x' \in \{\mathbf{R}_m, \mathbf{R}_e\}$, where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d_e \times d_e}$.
Sequential ($\mathcal{O}_{sq}$)	Identity	This is a default operation that does not transform the input feature.
	Recurrent Layer	Apply Recurrent Neural Network(RNN) to process the input feature, which is able to capture the temporal patterns. In our model, we use Gated Recurrent Unit (GRU) [19] as the basic operation.
	Self-Attention Layer	Apply self-attention mechanism [21] to model the long-term dependencies of the sequence feature.
	1-D Convolution	Apply 1-D convolution operation to process the feature and capture the local correlations. We use the padding and strides that can maintain the same shape as the input tensor.
	Feed-forward Layer	Apply a linear layer to all positions of the sequence feature.
	Sequential-Sequential Feature Interaction	Apply cross-modal attention mechanism to model the interaction between two sequence features. The current feature will serve as the queries, and the other feature will serve as the keys. The formulation is the same as Static-Sequential Feature Interaction except that the queries become sequential features this time.

Moving to the second stage, our approach encompasses a search space specifically designed for late fusion architectures. This search space consists of two crucial modules: the *feature selector* and the *searchable fusion module*. The feature selector component determines which modalities should be selected and incorporated into the fusion process. The searchable fusion module then identifies the optimal fusion operation to be applied. By jointly optimizing the feature selection and fusion operation within this search space, we enable the framework to automatically discover effective strategies for multi-modal feature fusion.

In the following sections, we will provide more detailed explanations of each stage, highlighting their respective contributions and functionalities.

5.3.2.1 Modality Specific Search

To facilitate the search process using differentiable methods, we assume that the architecture for each modality type consists of a network comprising K sequentially connected

Table 5.2: Fusion operations.

Operation	Operation Description
Sum	This operation sums all the input features together to fuse them: $o(u_c) = \mathbf{u}_1^{(c)} + \mathbf{u}_2^{(c)} + \cdots + \mathbf{u}_{4+c-1}^{(c)}$
MLP	This operation applies a multi-layer perceptron to the sum of all input features, where $\mathbf{W}_3 \in \mathbb{R}^{d_e \times d_e}$ and $\mathbf{b}_3 \in \mathbb{R}^{d_e}$: $o(u_c) = \text{ReLU}(\mathbf{W}_3\mathbf{u}'^{(c)} + \mathbf{b}_3)$, and $\mathbf{u}'^{(c)} = \mathbf{u}_1^{(c)} + \mathbf{u}_2^{(c)} + \cdots + \mathbf{u}_{4+c-1}^{(c)}$
Attentive Sum	This operation first uses linear projection to generate weights for all features and then applies a weighted sum to aggregate all features, where $\mathbf{W}_\phi \in \mathbb{R}^{d_e}$ and $b_\phi \in \mathbb{R}$: $o(u_c) = \sum_{i=1}^{4+c-1} \phi_i \mathbf{u}_i^{(c)}, [\phi_1, \cdots, \phi_{4+c-1}] = \text{Softmax}(l_1, \cdots, l_{4+c-1})$, and $l_i = \mathbf{W}_\phi \mathbf{u}_i^{(c)} + b_\phi, i = 1, \cdots, 4+c-1$

layers, as illustrated in Figure 5.1. Consequently, our objective is to search for the optimal operation for each layer k. To make the search space compatible with differentiable search methods, we define a mixed operation at each layer. This mixed operation is a weighted sum of all operations within a predefined set $\mathcal{O}$. The formulation can be expressed as follows:

$$\begin{aligned} x_k &= \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(k)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(k)})} o(x_{k-1}), \\ k &= 1, \cdots, K, \end{aligned} \tag{5.5}$$

where x_k represents the feature of layer k. x_0 can be any of the input features $[\mathbf{R}_m, \mathbf{R}_e, \mathbf{s}_p, \mathbf{s}_n]$. α's are architecture parameters that measure the weights for candidate operations, and we have four groups of α's in total for different modalities.

Next, we list the candidate operation sets for each modality in Table 5.1. As we have two types of input modalities, we design two distinct modality-specific search spaces to cater to their specific characteristics. For both types of features, in addition to the feature encoding operations, we also incorporate **interaction operations**. These interaction operations handle the interactions between the current modality and other modalities, both within the same category and across different categories. By including these interaction operations, our method is capable of exploring how to effectively process all the input features and discover fine-grained early fusion operations among different modalities. This allows for comprehensive modeling and integration of the various modalities present in the EHR data.

5.3.2.2 Multimodal Fusion Search

Once we obtain the output features for all modalities from the first stage, we apply a fixed max pooling operation to the encodings of sequence features over the sequence length dimension. This step ensures that we have encodings of the same shape for all modalities, denoted as $[\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4] \in \mathbb{R}^{d_e}$. Next, we will discuss how to effectively fuse these features using searchable modules.

Our fusion module follows a directed acyclic graph (DAG) design. Given the input features $[\mathbf{z}_1, \cdots, \mathbf{z}_4]$, we add one computation node at each step, and the newly added nodes can connect to all previous nodes and features, as shown in Figure 5.1.

Assuming we have a total of C nodes, at node c, we have the following features as inputs: $[\mathbf{z}_1, \cdots, \mathbf{z}_4, \mathbf{g}_1, \cdots, \mathbf{g}_{c-1}]$, where the $\mathbf{g}$'s represent the output features for the corresponding nodes. Our goal is to search for two operations: (1) selecting the previous features that should be taken as inputs to the current node, and (2) determining the fusion operation for the selected inputs. To accomplish these steps, we design two searchable modules: the feature selector and the searchable fusion module. These modules enable us to dynamically and adaptively determine the most relevant features and fusion operations at each node of the fusion module.

Feature Selector To handle the feature selection at each node c, we define an operation set $\mathcal{O}_a$ that consists of only two operations: **Identity** and **Zero**. These operations determine whether to select the corresponding feature or not. Similar to Eq. (5.5), we apply the mixed operation over $\mathcal{O}_a$ to each input feature for node c. As a result, at node c, we obtain a list of $(4+c-1)$ features, denoted as $u_c = [\mathbf{u}_1^{(c)}, \cdots, \mathbf{u}_{4+c-1}^{(c)}]$, where the values of these features can be either 0 or the original values. This process allows us to dynamically select relevant features based on the search algorithm's decisions at each node.

Searchable Fusion To determine the fusion operation for the selected features at each node c, we define a candidate set $\mathcal{O}_b$ that contains different fusion strategies, as listed in Table 5.2. These fusion strategies represent different ways of combining the selected features. Similar to the feature selection step, we apply mixed operations over $\mathcal{O}_b$ at each node to determine the fusion operation. This process allows us to explore and search for the optimal fusion strategy that effectively combines the selected features.

5.3.3 Prediction

To obtain a comprehensive representation of the entire EHR data, we linearly combine all the node features from the multi-modal fusion module $[\mathbf{g}_1, \cdots, \mathbf{g}_C]$, i.e., $\mathbf{h} = \sum_{c=1}^{C} \mathbf{w}_c \mathbf{g}_c$, where $\mathbf{w}_c \in \mathbb{R}$ is the learned weight.

For the binary classification problem, we use sigmoid as the activation function to calculate the prediction probability as follows:

$$\hat{y} = \text{sigmoid}(\mathbf{W}_y \mathbf{h} + \mathbf{b}_y). \tag{5.6}$$

For the multi-label classification problem, we use softmax to generate the probability score for all classes as follow:

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}_y \mathbf{h} + \mathbf{b}_y), \tag{5.7}$$

where $\hat{\mathbf{y}} \in \mathbb{R}^P$ and P is the number of classes.

5.3.4 Optimization

5.3.4.1 Supernet Training

We use the bi-level optimization technique as DARTS [49] to optimize the model architecture weights and all other learnable parameters simultaneously:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\mathbf{W}^*(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}), \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) + \lambda \cdot \text{Penalty}(\boldsymbol{\beta}) \\ &\text{s.t. } \mathbf{W}^*(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \text{argmin}_{\mathbf{W}} \mathcal{L}_{train}(\mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) \end{aligned} \tag{5.8}$$

where $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ are the architecture weights involved in the modality-specific search, feature selector, and searchable fusion, respectively. $\mathbf{W}$ denotes all other learnable parameters in the network, and $\mathcal{L}_{val}$ and $\mathcal{L}_{train}$ mean the validation loss and training loss.

Additionally, we add one more penalty term Penalty($\boldsymbol{\beta}$) to the architecture weights of the feature selector $\boldsymbol{\beta}$ in order to make the feature selector select diverse modalities at different step nodes. We achieve that by maximizing the cross entropy of any two

different steps of $\boldsymbol{\beta}$:

$$\text{Penalty}(\boldsymbol{\beta}) = -\sum_{c_1=1}^{C}\sum_{c_2=1}^{C} \text{CrossEntropy}(\boldsymbol{\beta}[c_1][:4], \boldsymbol{\beta}[c_2][:4]) \tag{5.9}$$

Note that we only take the first four values of $\boldsymbol{\beta}$ at each step, which indicates the selection of input modalities $[\mathbf{z}_1, \cdots, \mathbf{z}_4]$. In this way, the penalty term will lead the feature selector to select different combinations of input modalities for fusion at different steps. However, the selection of the following computation nodes $[\mathbf{g}_1, \cdots, \mathbf{g}_C]$ will not be affected, which enables the model to search for more complex fusion strategies without any constraints.

Algorithm 2: Pruning Supernet

Input: A pretrained supernet $\mathcal{S}$, Set of mixed operations $\mathcal{E}$ from $\mathcal{S}$
Output: Set of selected operations $\{o_e^*\}_{e\in\mathcal{E}}$, Pruned supernet $\mathcal{S}'$

```
while True do
    randomly select a mixed operation e ∈ E;
    for all operation o on e do
        evaluate the validation performance of S when o is removed (S_\o);
    end
    remove the worst operation for e: o'_e ← arg max_o Evaluate(S_\o);
    re-normalize architecture weights on e for remaining operations;
    finetune the remaining supernet S' for a few steps;
    if ∀e ∈ E, |{o|o ∈ e}| = 1 then
        break;
    end
end
```

5.3.4.2 Deriving the Optimal Architecture

After training the supernet, we introduce a novel method for deriving the final architecture by gradually pruning operations. We adopt the concept of perturbation-based architecture selection [76], which evaluates the importance of each operation by removing it from the supernet and analyzing the resulting performance changes. In contrast to the original approach that directly selects the optimal operation for each edge, our method removes only one operation per iteration during discretization. This incremental strategy enables the derivation of architectures that can retain multiple operations per edge, ultimately facilitating the formation of more advanced architectures.

Table 5.3: Statistics of the four datasets.

Task	ARF	Shock	Mortality	Diagnoses
# data samples	9,393	12,289	7,440	7,440
Dimension of $\mathbf{p}$	96	97	96	96
Dimension of $\mathbf{E}$	680	739	1,001	1,001
Dimension of $\mathbf{M}$	4,452	5,056	6,726	6,726
Recorded hours T	12	12	48	48

We introduce the process of pruning in Algorithm 2. Specifically, we denote the trained supernet as $\mathcal{S}$ and the set of all mixed operations as $\mathcal{E}$. The goal is to prune the supernet until only one operation o_e^* is left for each mixed operation $e \in \mathcal{E}$. In order to achieve that without too much performance drop from the supernet, we propose to gradually prune unimportant operations from the supernet, where the importance of each operation is determined by its contribution to the validation performance.

For each iteration, we first sample a mixed operation from the supernet and then prune the worst operation from it if removing the operation brings the best validation performance. Then, we will fine-tune the remaining architecture until it converges again. The pruning will stop when all mixed operations from the supernet only have one optimal operation remaining. In this way, we can observe how the performance drops during the whole process, and select the relatively better architecture before it drops too much. Therefore, we can get the final architecture with nearly the same performance as the trained supernet without training from scratch.

5.4 Experiments

5.4.1 Experimental Setups

Data Processing. Following FIDDLE [79], we extract data from the MIMIC-III dataset [1] and specifically focus on the 17,710 patients (23,620 ICU visits) recorded from 2008 to 2012. We extract the structured features within 48/12 hours using FIDDLE. These features are randomly divided into the training, validation, and testing sets in a 7:1.5:1.5 ratio. The unstructured texts within 48/12 hours are obtained by consolidating the latest notes of each category into one document and aligning them with the other structured features. The data statistics can be found in Table 5.3.

Prediction Tasks. We aim to address four prediction tasks: (1) Acute Respiratory

Failure at 12 hours (*ARF 12h*), (2) Shock at 12 hours (*Shock 12h*), (3) Mortality at 48 hours (*Mortality 48h*), and (4) Diagnoses at 48 hours (*Diagnoses 48h*). The first three tasks involve **binary** classification, aiming to predict whether patients will experience these severe conditions during their ICU stays based on the initial 12/48 hours of data. The last task focuses on predicting diagnosis codes upon hospital discharge based on the initial 48 hours of data, constituting a **multi-label** classification problem. After extracting the top three digits of ICD-9 codes[1], we have 1,025 disease labels for predictions.

Baselines. We include both handcrafted and automated models as our baselines. **Handcrafted models**: we include vanilla LSTM [20] and CNN [80] models as baselines. Following the settings in [72], we combine LSTM, CNN, Transformer [21], and Star-Transformer [81] with ClinicalBERT using Multimodal Adaptation Gate (MAG) [82], resulting in **eight** baseline models as shown in Table 6.3. **Automated models**: we also apply MUFASA [73] to our prediction tasks as a NAS baseline for multi-modal EHR data.

Implementation. During the supernet training stage, we employ the Adam optimizer [66] with a learning rate of $1e-4$ for the network parameter $\mathbf{W}$ and $1e-5$ for the architecture parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$. The loss function is cross entropy. The batch size for all tasks is set to 64, and the hidden dimension d_e is set to 256. To balance the bi-level optimization, the penalty term weight λ is set to 0.1. In the first stage, the number of layers K is set to 2, while in the second stage, the number of step nodes C is set to 3. The entire supernet training takes less than one hour on an NVIDIA A100 GPU, demonstrating the efficiency of the search algorithm. During the pruning of the trained supernet, we utilize the same settings, except for the fine-tuning phase, where we use a learning rate of $2e-6$.

Evaluation Metrics. Following [72], for three binary classification tasks, we use **AUROC** (Area Under the Receiver Operating Characteristic curve) and **AUPR** (Area Under the Precision-Recall curve) to evaluate the performance. For the diagnosis prediction task, we use **Top-K Recall** (R@K) instead and separately set K to be 10, 20, or 30. For all methods, we run the experiments for **five times** and report the mean values for a fair comparison.

[1] `https://www.cdc.gov/nchs/icd/icd9.htm`

5.4.2 Performance Evaluation

The results of the four tasks are presented in Table 5.4, where it can be observed that our method achieves the best performance across all metrics compared to the baseline approaches. These results demonstrate the superiority of our proposed method over existing state-of-the-art baselines.

For the three binary classification tasks aiming to predict emergent conditions within a short time window, real-time time-series data, such as vital signs, play a crucial role in achieving high performance. This importance of real-time data is further confirmed by our ablation study in Section 5.4.3. However, existing approaches fail to adequately distinguish the significance of such modalities and treat all modalities equally in the input, leading to subpar performance.

In the case of the multi-label diagnosis prediction task, where the model is required to predict from a vast set of 1,025 disease groups, the task complexity is higher compared to the other tasks. Consequently, models employing advanced fusion techniques outperform those using simple fusion methods. Notably, MUFASA performs significantly worse than other baselines in this task. This can be attributed to the large label space, which makes it challenging for MUFASA to converge to a favorable local optimum.

Table 5.4: Performance comparison on four tasks. The second-best results are marked by underline.

Tasks	ARF 12h		Shock 12h		Mortality 48h		Diagnoses 48h		
Metrics	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	R@10	R@20	R@30
LSTM	0.7377	0.3268	0.7364	<u>0.2940</u>	0.8827	0.5427	0.1516	0.1932	0.2224
CNN	0.7348	0.3214	0.7356	0.2775	<u>0.8876</u>	<u>0.5479</u>	0.1670	0.2501	0.2990
LstmBert	0.7310	0.3086	<u>0.7403</u>	0.2852	0.8755	0.5362	<u>0.2299</u>	0.3320	0.3994
BertLstm	<u>0.7441</u>	0.3200	0.7391	0.2912	0.8811	0.5409	0.2170	0.3298	0.4086
CnnBert	0.7201	0.2925	0.7348	0.2891	0.8853	0.5447	0.2206	0.3110	0.3693
BertCnn	0.7389	<u>0.3292</u>	0.7268	0.2776	0.8777	0.5450	0.2250	0.3346	0.4072
EncoderBert	0.7350	0.3117	0.7377	0.2857	0.8800	0.5296	0.1985	0.2459	0.2825
BertEncoder	0.6893	0.2633	0.7164	0.2818	0.8663	0.5183	0.2198	<u>0.3366</u>	<u>0.4108</u>
StarBert	0.7238	0.3194	0.7110	0.2869	0.8729	0.4878	0.2265	0.3305	0.3996
BertStar	0.6824	0.2641	0.7044	0.2825	0.8644	0.4724	0.2215	0.3173	0.4026
MUFASA	0.7362	0.3088	0.7295	0.2735	0.8812	0.5436	0.0903	0.1156	0.1296
`AutoFM`	**0.7565**	**0.3593**	**0.7463**	**0.3039**	**0.8900**	**0.5514**	**0.3338**	**0.4780**	**0.5658**

5.4.3 Ablation Study on Input Modalities

In this ablation study, our objective is to assess the importance of each modality in the final prediction by systematically removing them from the model input alternatively.

Table 5.5: Ablation study on input modalities

Task	ARF	Shock	Mortality	Diagnoses
Metrics	AUPR	AUPR	AUPR	R@10
`AutoFM`	**0.3593**	**0.3039**	**0.5514**	**0.3338**
- Demographic	0.3376	0.2822	0.5553	0.3310
- Continuous Events	0.2966	0.2310	0.4010	0.3074
- Discrete Events	0.3551	0.3027	0.5336	0.3294
- Clinical Notes	0.3362	0.2773	0.5501	0.3265

Table 5.6: Results on different optimization methods.

Tasks	ARF	Shock	Mortality	Diagnoses
Metrics	AUPR	AUPR	AUPR	R@10
Supernet	**0.3565**	**0.3053**	**0.5549**	0.3251
Supernet⊖	0.3445	0.3027	0.5473	**0.3257**
`AutoFM`	**0.3593**	**0.3039**	**0.5514**	**0.3338**
AutoFM⊖	0.3410	0.3031	0.5471	0.3316

The experimental results are presented in Table 5.5. It can be observed that removing any modality leads to a drop in performance, highlighting the significance of considering multimodal EHR data as input for the model. Notably, the time series modality predominantly encompasses vital signs information, which is a strong indicator of emergency conditions occurring during ICU stays and aids in accurate diagnoses. Consequently, removing continuous event features from the input leads to the largest performance drop compared to the original framework.

5.4.4 Effect of Feature Selection Penalty

We introduce a new loss term in the optimization process to guide the selection of optimal modalities in the fusion stage. To assess the effectiveness of this designed penalty term (Eq.(5.9)), we conducted a study, and the results are presented in Table 5.6.

"Supernet" refers to the learned supernet using our proposed `AutoFM` with Eq. (5.8), while "Supernet⊖" indicates the supernet without the penalty term defined in Eq. (5.9). We observe that the inclusion of the penalty term improves the performance of the supernet across all three binary classification tasks. For the diagnoses prediction task, comparable performance is still achieved even without the penalty term. These results confirm the term's ability to ensure the diversity of feature selection in the multi-modal fusion process.

Table 5.7: Results on different discretization methods.

Tasks	ARF	Shock	Mortality	Diagnoses
Metrics	AUPR	AUPR	AUPR	R@10
DARTS	0.3124	0.2891	**0.5536**	0.3332
DARTS-PT	0.3433	0.2986	0.5444	0.3268
`AutoFM`	**0.3593**	**0.3039**	0.5514	**0.3338**

We also compare the derived model from Supernet$\ominus$, denoted as "AutoFM$\ominus$", which still employs the proposed pruning-based architecture discretization outlined in Algorithm 2. Notably, without the designed penalty, the derived model AutoFM$\ominus$ exhibits inferior performance compared to our proposed `AutoFM`. This comparison further demonstrates the necessity and effectiveness of the penalty term in the optimization process.

5.4.5 Effect of Pruning-based Architecture Selection

In Section 5.3.4.2, we present a pruning-based discretization method to derive the optimal architecture. To evaluate the effectiveness of this approach, we derive two models from the same supernet trained using Eq. (5.8), employing the approaches used in DARTS [49] and DARTS-PT [76]. The results are shown in Table 5.7. The performance of the derived models using our pruning-based approach is significantly better compared to the models derived using DARTS and DARTS-PT. This indicates that our method effectively captures the optimal architecture from the supernet and achieves superior performance. These results provide strong evidence supporting the effectiveness of our proposed discretization method.

5.4.6 Architecture Study

Figure 5.2 provides an example of the searched architecture for the ARF task. It demonstrates the effectiveness of our approach in capturing the interactions between different modalities during the modality-specific search stage. Furthermore, in the second stage of multi-modal fusion, our method is able to explore different combinations of input modalities $[\mathbf{z}_1, \cdots, \mathbf{z}_4]$ at different steps. This is achieved through the penalty imposed by the additional loss term, which encourages diversity in the feature selection process. As a result, our fusion module can effectively determine the optimal fusion strategy, leading to improved performance in capturing the underlying patterns and relationships

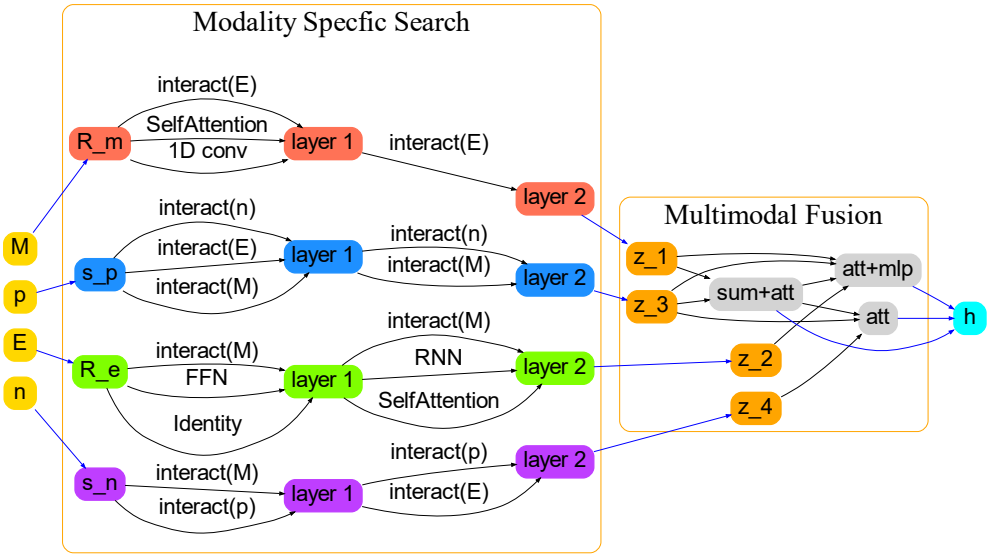


Figure 5.2: Searched architecture. The blue arrows represent fixed operations, while the other black arrows are all searched operations. The interact$(\cdot)$ means the interaction operation with the corresponding feature. For the steps nodes $[\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3]$, we omit the notations in the figure and fill the node with the selected fusion operations like (sum+att).

within the multi-modal EHR data.

5.4.7 Working Procedure of Pruning-based Architecture Selection

In this study, we investigate the process of pruning-based architecture selection and visualize the performance changes throughout the pruning procedure. We present the validation and test performance in terms of AUPR on the ARF 12h task, as shown in Figure 5.3.

Upon observation, we note a slight improvement in performance on both validation and test sets during the initial 20 steps. This improvement is potentially due to the removal of supernet operations that have a negative impact on performance, thereby increasing the AUPR to some extent. Subsequently, there are two noticeable performance drops at around steps 20 and 40. These drops indicate that when a significant number of operations are removed from the supernet, resulting in significant changes in the model architecture, the fine-tuning process is unable to effectively recover the performance.

Based on this observation, we can make the decision to halt the pruning process before reaching step 20. By doing so, we obtain the final architecture that demonstrates

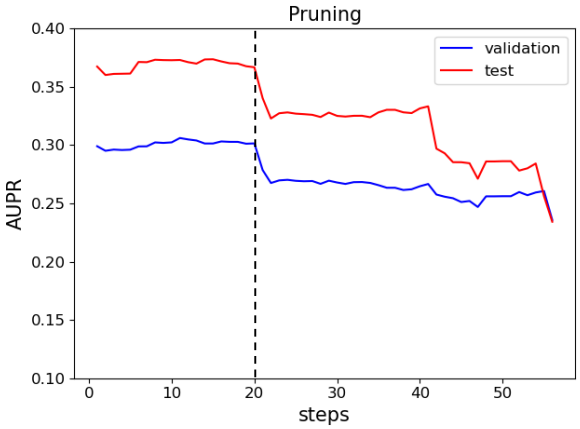


Figure 5.3: Pruning curve on the ARF 12h task.

either the same or even better performance than the original supernet.

5.5 Limitation and Future Work

The current proposed model relies on a pre-trained language model to generate embeddings for clinical notes, making it impractical to directly search for the best token-level architecture for encoding clinical notes and modeling interactions with other modalities. Additionally, the current model only considers a limited number of modalities from electronic health record (EHR) data. However, there are various other types of medical data, including diagnosis codes, procedure codes, drug codes, and medical images, that have the potential to contribute valuable information. In our future work, we aim to address these limitations by incorporating additional modalities and designing more powerful models. This expansion will allow us to capture a broader range of information from diverse medical data sources, enabling more comprehensive and accurate predictions. By exploring the integration of different modalities and developing advanced models, we strive to enhance the capabilities and performance of our framework in handling a wider array of medical data types for improved predictive modeling in healthcare.

Moreover, the current framework requires searching a specific architecture for every new task we have, limiting the efficiency when dealing with multiple datasets or tasks. So, the future direction is to develop transferrable AutoML solutions for EHR data that can leverages the information from previous tasks or datasets to facilitate the searching of new tasks.

5.6 Conclusion

In this paper, we introduced `AutoFM`, a novel Neural Architecture Search (NAS) framework designed for automatically fusing multi-modal EHR data. Our experimental results demonstrated the superior performance of our method compared to existing state-of-the-art baselines across various tasks.

Chapter 6
Automated Multi-Task Learning for Joint Disease Prediction on Electronic Health Records

6.1 Introduction

In the era of big data and digital healthcare, the voluminous Electronic Health Records (EHR) have emerged as a treasure trove of valuable information that can revolutionize patient care, medical research, and clinical decision-making. As a result, the data mining community has been working on designing machine learning models to predict patients' future health conditions by extracting patterns from their EHR data, such as mortality prediction [8], diagnosis prediction [5, 67] and hospital readmission [68].

Although most of the existing machine learning based prediction models are designed to be single-task, i.e. predicting the risk of a single target disease, some of the existing works [83–87] proposed to design multi-task learning (MTL) models to jointly predict for multiple targets. The motivation behind this lies in the fact that two or more diseases might be related to each other in terms of sharing common comorbidities, symptoms, risk factors, etc. Consequently, training on related diseases simultaneously can offer additional insights and potentially enhance prediction performance. While multi-task learning offers potential advantages, the existing MTL frameworks for EHR data still suffer from the following limitations.

Limitations of the existing MTL frameworks for EHR data. To design an effective MTL framework, two fundamental challenges need to be addressed:

(1) How can we determine which tasks should be trained together? The task grouping

problem [88] involves finding groups of tasks to train jointly. Multi-task learning only provides advantages when the tasks are synergistic, i.e., training on the tasks together makes the model learn general knowledge that helps in performing the tasks better in the test set and prevents overfitting. Thus, given a large set of related tasks in a domain, we may need to group the tasks (allowing tasks to belong to multiple groups) together to create groups of tasks on each of which we will train a model. However, existing works usually rely on human expert discretion to select multiple tasks upfront and create a shared model for those tasks [83–87]. Hence, none of them has addressed the general problem of task grouping for EHR data. Moreover, due to the complexity of disease correlations, grouping synergistic tasks together is extremely challenging for human experts. It not only demands substantial effort (trying out every possible task combination) but also introduces the risk of task interference (putting disparate diseases together), potentially leading to performance degradation. Therefore, how to design the appropriate task grouping for MTL on EHR data presents a critical challenge.

(2) How can we design model architectures for MTL? Existing works [83–87] typically rely on hand-crafted architectures for multi-task learning, which consist of a shared EHR encoder followed by several task-specific classifiers. However, due to the large number of possible operations as well as network topologies, manually tuning an optimal architecture for MTL is impossible. Furthermore, the optimal architectures for different task groups might also be distinct. Thus, things can even get worse when the number of tasks grows and different task combinations are involved for joint tuning. Therefore, we need a more efficient and effective approach to design the optimal MTL architectures for EHR data.

Automating the MTL framework design for EHR data. To address the aforementioned challenges, we look to Automated Machine Learning (AutoML) [89]. Since AutoML relies on data-driven approaches to automate the design of machine learning algorithms, it has the potential to improve the design of an MTL framework for EHR data and reduce human interventions. Several attempts have been explored in other domains, e.g., computer vision, to improve the design of task grouping [88, 90, 91] and MTL architectures [92–96]. However, the exploration of AutoML in healthcare domain remains relatively limited [97]. To the best of our knowledge, there are no existing work that automates the finding of groups of tasks for MTL towards designing an optimal framework for classification tasks using EHR data, which is a notable gap in the field.

Joint optimization over task grouping and architecture search. Morever, currently there exists no end-to-end optimization framework for automating MTL, even in other domains. Current approaches independently address the problems of task grouping

and architecture design. First, a line of work [88,90,91] solves the task grouping problems by learning the task correlations. They operate under the underlying assumption that MTL architectures are the same across different task groups, which might not be practical nor optimal. Second, researchers also apply Neural Architecture Search (NAS) [53] to automatically design MTL architectures for a predefined set of tasks [92–96, 98]. No existing work has integrated these two approaches to address both problems simultaneously. However, combining them naively could lead to sub-optimal results, as sequential optimization might result in inaccurate estimations for both aspects. Therefore, we need a more generalized AutoML framework for the joint optimization of both task grouping and architecture search.

Overview of the proposed approach. Therefore, in this paper, we show that an integrated approach for multi-task grouping and neural architecture search provides significant improvements. First, we extend existing single-task models like Retain [4], Adacare [94] to MTL in an EHR setting. Second, we apply DARTS [49], an NAS method used in MTL settings in different domains to the EHR domain. We use one shared model for predicting multiple tasks. These adaptations improve over the single-task setting. Second, we explore the impact of automated task grouping in the EHR setting by grouping tasks and finding an optimal NAS model for each task group. This further improves the performance. Finally, we propose an integrated framework an **Auto**mated multi-task learning framework, `AutoDP`, for joint **D**isease **P**rediction on electronic health records, which aims at jointly searching for the optimal task grouping and the corresponding neural architectures that maximize the multi-task performance gain. We show that this third method provides the maximal performance gain.

Specifically, in `AutoDP`, we employ a surrogate model-based optimization approach [99] for efficient search. First, we define the joint search space of task combinations and architectures that includes all possible configurations for MTL. We want to find optimal solutions from this search space. To achieve that, the first question is how we can evaluate the performance of each configuration. Performing the ground truth evaluation for every configuration is infeasible, since it requires an entire multi-task learning procedure for each pair of architecture and task combination. Therefore, instead of exhaustively evaluating all the configurations, we build a surrogate model to predict the multi-task gains for any given configurations from the search space. In this way, we only need to evaluate the ground truth gains for a subset of samples from the search space, and use them to train the surrogate model for estimating the rest ones. The intuition is that there exists an underlying mapping from each configuration to the expected multi-task gains;

thus it can be learned by a neural network. The remaining question is how we can effectively train the surrogate model using as few samples as possible. To this end, we further propose a progressive sampling strategy to guide the surrogate model training for improving sample efficiency. That is we train the surrogate model through multiple iterations. At every iteration, we select some points from the search space and update the surrogate model accordingly. The selection is conditioned on the current surrogate model and involves both exploitation and exploration. That is, we iteratively select the points that bring higher performance gains and also come from unexplored areas, which makes the training samples represent the whole search space. Eventually, after we obtain the trained surrogate model, we further use it to derive the final optimal task grouping and architectures. Because of the huge search space, it is not practical to use brute-force search. Hence, we develop a greedy search method to find the near-optimal solution.

In summary, our contributions are as follows:

- We are the first to propose an automated approach for multi-task learning on electronic health records `AutoDP`, which largely improves the design of task grouping and model architectures by reducing human interventions. Specifically, this work is the first to automate the design for the optimal task grouping and model architectures for MTL on EHR data.

- We are the first to propose a surrogate model based optimization framework that jointly searches for the optimal task grouping and corresponding model architectures with high efficiency in any domain.

- We propose a progressive sampling strategy to construct the training set for the surrogate model, which improves sample efficiency by reducing the required number of ground truth evaluations during searching. Importantly, we balance exploitation and exploration so that the sampled configurations can represent the whole search space and are highly accurate.

- We propose a greedy search algorithm to derive the final MTL configuration using the trained surrogate model and find a near-optimal solution from the huge search space efficiently.

- Experimental results on real world EHR data - MIMIC IV [2] demonstrate that `AutoDP` improves classification performance significantly over existing hand-crafted and automated methods under feasible computational costs.

6.2 Literature Review

6.2.1 Multi-Task Learning with EHR

To enhance prediction performance while forecasting patients' health conditions based on their historical data [100], existing studies employ multi-task learning to simultaneously predict multiple related target diseases or conditions, resulting in improved performance compared to single-task training. For example, Wang, et al. [85] investigated the advantages of joint disease prediction using traditional machine learning models. More recently, researchers have applied recurrent neural network (RNN) based models to conduct multi-task learning on EHR data [83, 84, 101], which is able to predict tasks like mortality, length of stay, ICD-9[1] diagnoses and etc. Additionally, Zhao, et al. [86] also utilized a transformer based method for multi-task clinical risk prediction on multi-modal EHR data. However, all these studies manually select the set of tasks for joint training without task grouping and utilize a hand-crafted MTL model architecture, which largely limits their performance.

6.2.2 Multi-Task Grouping

Due to the limitation of manually selected task groups, some of the work focus on obtaining the optimal task grouping through searching. Specifically, Standley, et al. [88] is the first work that systematically analyze the task correlations. For improving the efficiency, they use pair-wise MTL gains to estimate the high-order MTL gains, and obtain the pair-wise gains by training one model for each task pair. Based on the estimated gains, they derive the optimal task grouping using brute-force search. Fifty, et al. [90] further improves the efficiency by training one model to derive all the pair-wise gains. They derive the task affinity based on the gradient information during training. Furthermore, Song, et al. [91] propose a more general method that employs a meta model to learn the task correlations and estimates the high-order MTL gains more effectively. These works normally assume that the model architecture is the same across different task groups. But in practice, we can maximize performance gains by applying different model architectures with respect to each task group. Thus, we need a more general framework that considers the model architectures during task grouping.

[1] `https://www.cdc.gov/nchs/icd/icd9.htm`

6.2.3 Multi-Task NAS

Neural Architecture Search (NAS) [53] stands as a prominent research area in AutoML, focusing on the exploration of optimal deep network architectures through a data-driven approach. Although the main stream of NAS focuses on the setting of single task learning, some researchers also try to employ NAS in multi-task learning applications, predominantly for searching computer vision MTL architectures. Notably, studies done by Ahn, et al. [92] and Bragman, et al. [93] employ reinforcement learning and variational inference, respectively, to determine whether each filter in convolutions should be shared across tasks. Furthermore, other recent works [94–96, 98] leverage differentiable search algorithms [49], to determine the optimal sharing patterns across multiple network layers for diverse tasks. Despite the demonstrated advancements, a common limitation is their reliance on human experts to pre-define a set of tasks for joint training. This constraint poses challenges in practical scenarios where task grouping is not readily available, thereby limiting their broader applicability. What is more, their frameworks often search for better MTL architectures on top of one or several backbone architectures such as ResNet [102]. However, such backbone architectures might not be available for EHR applications in medical domain. Therefore, a new multi-task NAS framework is needed for EHR data.

In the NAS domain, surrogate model-based optimization (SMBO) has been introduced to efficiently search for the optimal architecture within a given search space [50, 103, 104]. This approach addresses the practical challenges of NAS, where optimization problems lack analytic objective functions, and evaluating a candidate solution can take hours or even days. SMBO overcomes these limitations by employing a predictive model (the surrogate model) to approximate the objective function, enabling informed decision-making during the search process. Beyond solving the NAS problem, our work extends the problem setting to joint optimization of task grouping and search space while introducing a novel SMBO framework tailored for this joint optimization.

6.3 Methodology

6.3.1 Preliminaries

Problem definition. Assume we have the input EHR data for multiple patients where each patient is represented as $\mathbf{X} \in \mathbb{R}^{L \times d_e}$, where L is the time sequence length and d_e is the hidden dimension of the input features. We have N prediction tasks using

the EHR data, denoted as $\mathcal{T} = \{T_1, T_2, \cdots, T_N\}$. We seek to maximize the overall MTL performance gain for all these prediction tasks compared to single task training. First, we define MTL gain. Conduct a single task training on each task independently using a specific backbone model (such as RNN), and obtain the single-task performance for all tasks in terms of a predefined metric (such as average precision), denoted as $\{s_1, s_2, \cdots, s_N\}$. Then, the MTL gain is defined as:

$$g_i = \frac{(m_i - s_i)}{s_i}, i = 1, \cdots, N, \tag{6.1}$$

where m_i is the multi-task performance for T_i. Therefore, our objective is to maximize the overall gain for all tasks: $G = \frac{1}{N}\sum_{i=1}^{N} g_i$.

To achieve that, our proposed method solves two searching problems at the same time using AutoML. First, we search for a list of task combinations that defines which tasks should be trained together. Second, we determine the optimal model architecture for each task combination. We aim at finding the optimal configuration for both, such that the highest overall gain G is attained.

Task grouping search space. For N tasks, there are $2^N - 1$ task combinations, $\mathcal{C} = \{C_1, C_2, \cdots, C_{2^N-1}\}$, where every C is a subset of $\mathcal{T}$. Given a budget B, we aim at searching for maximally B task combinations from $\mathcal{C}$ to determine which tasks should be trained together. The task combinations should cover all N tasks so that we are able to obtain $\{m_1, m_2, \cdots, m_N\}$. If one task T_n appears in multiple task combinations, we simply choose the highest performance for it as m_n.

Architecture search space. For every task combination, we also need to search for an MTL architecture to model the EHR data. We adopt the hard sharing mechanism as in most existing works [83, 101], which consists of a shared encoder for extracting the latent representation of the input EHR and multiple task specific classifiers to generate the output for every task.

Specifically, we enable the search for the optimal shared encoder. For the search space of the encoder, we adopt the setting of directed acyclic graph (DAG) [49]. The architecture is represented as a DAG that consists of P ordered computation nodes, and each node is a latent feature that has connections to all previous nodes. For each connection (also called edge), we can choose one operation from a predefined set of candidate operations $\mathcal{O}$ for feature transformation. Let $\mathbf{E}^0 = \mathbf{X}$, the formulation of node p is defined as follows:

$$\mathbf{E}^p = \sum_{i=0}^{p-1} o_{(i,p)}(\mathbf{E}^i), o_{(i,p)} \in \mathcal{O}, \tag{6.2}$$

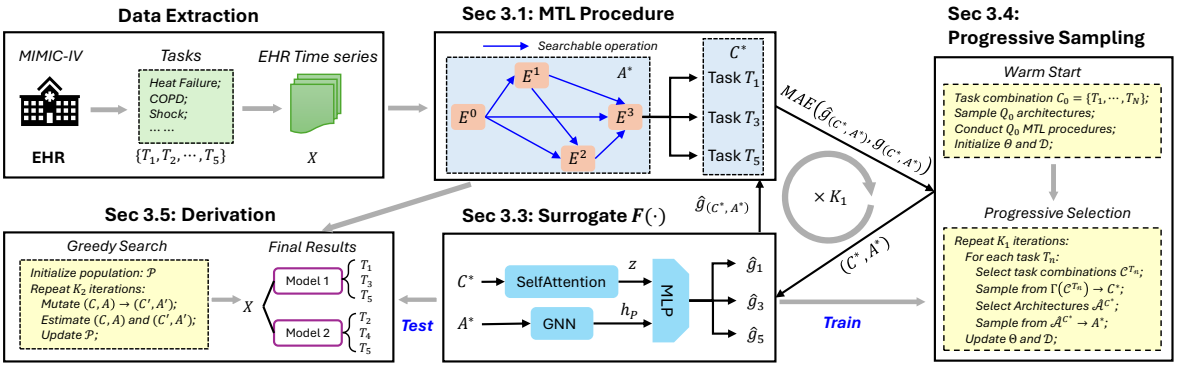


Figure 6.1: Overview of the proposed `AutoDP`

where node features $\mathbf{E}^i \in \mathbb{R}^{L \times d_e}$'s all have the same dimension as $\mathbf{X}$, and $o_{(i,p)}$ is the operation that transform $\mathbf{E}^i$ to $\mathbf{E}^p$. Essentially, sampling one architecture from the search space is equivalent to sampling one operation for every edge in the DAG. In this way, we can get the set of all possible architectures denoted as $\mathcal{A}$.

Finally, to predict, we take the last node representation $\mathbf{E}^P$ as the encoded feature for the input EHR, and use task-specific classifiers to output final predictions, which are all fixed fully connected network layers.

MTL procedure. To evaluate a specific sample from the joint search space $\mathcal{C} \times \mathcal{A}$, we need to conduct an MTL experiment to obtain the multi-task performances. Specifically, given an architecture $A \in \mathcal{A}$ and a task combination $C \in \mathcal{C}$, we train the model A to predict for all tasks in C and get the multi-task performances for those tasks. Then, we can compute their gains by Eq.(6.1). In this way, we are able to evaluate how much gains that this sample (C, A) could achieve.

6.3.2 Overview

We propose a surrogate model based AutoML framework to search for the optimal task grouping and corresponding architectures simultaneously. To achieve that, we need to first evaluate the MTL gains for all the samples in the joint search space $\mathcal{C} \times \mathcal{A}$, and then select the best B samples (pairs of task combinations and architectures) that maximize G. However, it is not practical to obtain the ground-truth gains for every sample, since the whole search space is normally very huge and every MTL procedure is also considerably expensive. Therefore, we build a neural network (called surrogate model) to learn the mapping from a pair of task combination and architecture to the multi-task gains:

$$\mathbf{g}_{(C,A)} = F(C, A), C \in \mathcal{C}, A \in \mathcal{A}, \tag{6.3}$$

where $\mathbf{g}_{(C,A)} \in \mathbb{R}^{|C|}$ is the per-task gains for task combination $C \in \mathcal{C}$ if using A as the model, and $F(\cdot)$ is the surrogate model. In this way, we only need to evaluate the ground truth gains for a small subset of samples from the search space, and use them to train the surrogate model for estimating all other unseen samples. The assumption is that the multi-task gains are essentially determined by the configuration of the task combination and the architecture, so there exists an underlying mapping that could be learned by a neural network. We set universal hyperparameters and optimization settings for all MTL procedures, hence the influence of other factors can be ignored.

Specifically, we introduce the model architecture of the surrogate model in Section 6.3.3. Then, we outline the training procedure of the surrogate model in Section 6.3.4, where we propose an active learning strategy to collect training samples. Eventually, we use greedy search to derive the final configuration of task grouping and architectures by utilizing the trained surrogate model, as discussed in Section 6.3.5. The framework overview is shown in Figure 6.1.

6.3.3 Surrogate Model

For learning the mapping from an input configuration to the multi-task gains, the surrogate model is required to encode both architectures and task combinations. Also, the model needs to output multi-task gains. Therefore, we design a new surrogate model that consists of two encoders that respectively transform the input architecture and task combination into latent representations. Then, two representations are fused together to predict the multi-task gains.

Architecture Encoding. For encoding a given architecture A, we apply a graph encoder [105] that is specifically designed for modeling DAGs, which is suitable for encoding the architectures in our search space. It can sequentially update the hidden states for the P computation nodes in preceding order by aggregating information from all predecessors. For node p, we have:

$$\mathbf{h}_p = \text{Aggregate}(\mathbf{W}_0 \cdot \mathbf{h}_0, \mathbf{W}_1 \cdot \mathbf{h}_1, \cdots, \mathbf{W}_{p-1} \cdot \mathbf{h}_{p-1}), \tag{6.4}$$

where $\mathbf{h}_0 \in \mathbb{R}^{d_s}$ is the input node representation which contains trainable parameters, and $\mathbf{W} \in \mathbb{R}^{d_s \times d_s}$'s are learnable transition matrices constructed for each operation in $\mathcal{O}$. For every operation in the architecture, we also apply the corresponding $\mathbf{W}$ in our graph encoder. For aggregating all incoming representations, we apply average pooling to obtain the node representation $\mathbf{h}_p$. Finally, we use the node representation for the last

node $\mathbf{h}_P$ as the overall encoding for the input architecture.

Task Combination Encoding. For encoding a given task combination C, we use the self attention mechanism [21] to model the high order interactions among the selected tasks in C. Specifically, we randomly initialize the embedding for all N tasks, and for task combination C, we have:

$$\mathbf{z} = \text{Pool}(\text{SelfAttention}(\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_{|C|})), \tag{6.5}$$

where $\mathbf{u} \in \mathbb{R}^{d_s}$'s are corresponding embeddings for the selected tasks and $\mathbf{z} \in \mathbb{R}^{d_s}$ is the final representation for task combination C. Additionally, we also use average pooling on top of the self attention layers to obatin $\mathbf{z}$.

Prediction. Eventually, we apply a two layer MLP to fuse both architecture encoding $\mathbf{h}_P$ and task combination encoding $\mathbf{z}$, and output the predicted gains for all selected tasks $\hat{\mathbf{g}}_{(C,A)} \in \mathbb{R}^{|C|}$. We use the mean absolute error to supervise the surrogate model as follows:

$$\mathcal{L}(\hat{\mathbf{g}}_{(C,A)}, \mathbf{g}_{(C,A)}) = ||\hat{\mathbf{g}}_{(C,A)} - \mathbf{g}_{(C,A)}||_1, \tag{6.6}$$

where $\mathbf{g}_{(C,A)} \in \mathbb{R}^{|C|}$ is the ground truth gains generated by conducting an MTL procedure for (C, A).

6.3.4 Progressive Sampling

In order to efficiently train the surrogate model defined in previous section, we develop a progressive sampling method to collect training samples. Start with an empty training set and a random initialized surrogate model, we progressively sample more points from the search space $\mathcal{C} \times \mathcal{A}$, and then use them to train the surrogate model. Specifically, we include two stages for the surrogate model training:

Warm start. Firstly, we warmup the surrogate model by selecting a small number of samples from the search space. Specifically, we use the task combination that contains all N tasks $C_0 = \{T_1, \cdots, T_n\}$ and randomly sample Q_0 architectures from $\mathcal{A}$. Then we conduct Q_0 MTL procedures to evaluate their gains by training on C_0. In this way, we collect Q_0 training samples as the initial training set denoted as $\mathcal{D}$. We further train the surrogate model on D, and denote the model parameters as $\mathbf{\Theta}$.

Progressive selection. Then, we progressively select more points and train the surrogate model as introduced in Algorithm 3. Totally, we conduct K_1 rounds of sampling. For each round, we iterate through all N tasks. With respect to one task T_n, we build

Algorithm 3: Progressive Selection

Input: Training set $\mathcal{D}$, surrogate model parameter $\mathbf{\Theta}$, Q_1, Q_2, K_1; $Q_1 > Q_2$.
Output: Updated $\mathcal{D}$ and $\mathbf{\Theta}$

1 **for** $k = 1, 2, \cdots, K_1$ **do**
2 **for** $n = 1, 2, \cdots, N$ **do**
3 Collect all task combinations that contains T_n: $\mathcal{C}^{T_n} = \{C_j | \forall C_j \in \mathcal{C}, T_n \in C_j\}$;
4 **for** $\forall C_j \in \mathcal{C}^{T_n}$ **do**
5 Randomly sample Q_1 architectures from $\mathcal{A}$, denote the set as $\mathcal{A}^{C_j}$;
6 Forward the surrogate model to collect gains for T_n with every architecture in $\mathcal{A}^{C_j}$: $\mathcal{G} = \{\mathbf{g}[T_n] | \forall A \in \mathcal{A}^{C_j}, \mathbf{g} = F(C_j, A)\}$;
7 Select the top Q_2 architectures from $\mathcal{A}^{C_j}$ with highest gains in $\mathcal{G}$, denoted as $\hat{\mathcal{A}}^{C_j}$;
8 Calculate the mean over top Q_2 gains from $\mathcal{G}$, denoted as μ^{C_j};
9 Calculate the variance over top Q_2 gains from $\mathcal{G}$, denoted as σ^{C_j};
10 **end**
11 Compute the acquisition values over $\mathcal{C}^{T_n}$ as: $\mathbf{\Gamma}(\mathcal{C}^{T_n}) = \{\mu^{C_j} + \lambda \cdot \sigma^{C_j}, \forall C_j \in \mathcal{C}^{T_n}\}$;
12 Sample a task combination C^* from $\mathcal{C}^{T_n}$ that has highest value in $\mathbf{\Gamma}(\mathcal{C}^{T_n})$, and randomly sample an architecture A^* from $\hat{\mathcal{A}}^{C^*}$;
13 Conduct an MTL procedure on (C^*, A^*), and collect the ground truth labels $\mathbf{g}_{(C^*, A^*)}$;
14 Add $(C^*, A^*, \mathbf{g}_{(C^*, A^*)})$ to $\mathcal{D}$;
15 **end**
16 Update $\mathbf{\Theta}$ by training the surrogate model on $\mathcal{D}$;
17 **end**

the acquisition function $\mathbf{\Gamma}$ over the set of task combinations that contains T_n based on the predicted gains for T_n. Then, we select one task combination C^* that have highest value. We apply Upper Confidence Bound [106] as the acquisition function that considers both exploration and exploitation by explicitly estimating the mean and variance of predicted gains (line 11 marked by blue). Besides that, we would also like to see the effect of exploration vs exploitation. so we try out different settings of $\mathbf{\Gamma}$. Specifically, we propose three variants of `AutoDP`, namely `AutoDP`$^{\mu+\sigma}$, `AutoDP`$^{\mu}$ and `AutoDP`$^{\sigma}$, which corresponds to the original setting, including only μ or only σ in $\mathbf{\Gamma}$. In this way, we can compare the results with pure exploration and pure exploitation during sampling, and find out the optimal strategy for `AutoDP`. Moreover, we also select one architecture A^* with high predicted gain for T_n when combined with C^*. The selection of C^* and A^* is interdependent, and the details are introduced in Algorithm 3. In this way, we collect

one sample (C^*, A^*) to update the training set $\mathcal{D}$ with respect to each T_n. At the end of each round, we also update the surrogate model parameters $\mathbf{\Theta}$ with the updated $\mathcal{D}$. After K_1 rounds, we are able to obtain a well trained surrogate model for estimating the whole search space.

Algorithm 4: Greedy Search

Input: Trained surrogate model $F(\cdot)$, B, K_2.
Output: Searched population $\mathcal{P}$.

```
Randomly sample B pairs from C × A to initialize P ;
for v = 1, 2, ··· , K_2 do
    Randomly select one pair (C, A) from P;
    Mutate (C, A) to (C', A') by changing one task in C or one operation in A,
     and obtain a new population P';
    Estimate P' and P using F(·);
    Choose the better one: P ← Select(P', P) ;
end
```

6.3.5 Derivation

We derive the final results using the trained surrogate model. Due to the huge search space, it is still not practical to use brute force search to get the global optimum. Therefore, we propose to apply a greedy method to search for near-optimal solutions. We introduce the detailed procedure in Algorithm 4. The high level idea is that we first randomly initialize the configuration, and then gradually improve its multi-task gain by random mutation and greedy selection.

Specifically, given the budget B, we aim at searching for B samples from the search space $\mathcal{C} \times \mathcal{A}$ such that the overall gain G is maximized. We first randomly initialize the population $\mathcal{P}$ that contains B pairs of task combinations and architectures. Then, at every iteration, we randomly mutate one pair (C, A) from the population and see whether the overall multi-task gain will increase. If so, we update $\mathcal{P}$ accordingly. After K_2 iterations, we can obtain a near-optimal solution. In practice, we also apply multiple initial populations to avoid getting stuck on local optima. Although we only get an approximate solution, our method can already achieve significant improvements over baselines as shown in Section 6.4.2.

6.4 Experiments

6.4.1 Set Up

Dataset & Tasks. We adopt MIMIC - IV dataset [2] for our experiments, which is a publicly available database sourced from the electronic health record of the Beth Israel Deaconess Medical Center. Specifically, we extract the clinical time series data for the 56,908 ICU stays from the database as our input EHR data, with an average sequence length of 72.9. With respect to each ICU stay, we also extract **25 prediction tasks** (listed in Table 6.1), including chronic, mixed, and acute care conditions. Each condition is associated with a binary label indicating whether the patient has the corresponding condition during the ICU stay.

To prepare our dataset, we adopt the data pre-processing pipeline outlined in Harutyunyan, et al. [101]. Given that the original implementation[2] is designed for MIMIC-III [1], we make specific modifications to tailor it for MIMIC-IV. The 25 labels are defined using the Clinical Classifications Software (CCS) for ICD-9 code[3]. Consequently, we first map the ICD-10 codes[4] in the MIMIC-IV database to ICD-9 codes before generating the labels. After processing, we have the feature dimension d_e as 76. We partition the dataset as train, validation and test sets with a ratio of $0.7 : 0.15 : 0.15$.

Implementation We implement the framework using the PyTorch framework and run it on an NVIDIA A100 GPU. Given the dataset we have, we first train a vanilla LSTM for every task independently, and report the backbone performance in Table 6.1, which can be further used to compute multi-task gains. For the proposed method, we run three settings of experiments: **Task @ 5**, **Task @ 10** and **Task @ 25**, which refers to using the first 5 tasks, 10 tasks and 25 tasks respectively. For different settings, we use specific hyperparameters as shown in Table 6.2. Besides that, we define the candidate operation set $\mathcal{O}$ as {Identity, Zero, FFN, RNN , Attention}, which includes widely used operations for processing EHR time series. Among them, Identity means maintaining the output identical to the input. Zero means setting all the values of the input feature to 0. Attention and FFN represents one self-attention layer and one feed-forward layer respectively, which are the same as in Transformer [21]. RNN is one recurrent layer, and we adopt LSTM [20] in our framework. For all the MTL procedures and baseline training, we apply the batch size of 64 and learning rate of $3e-4$. For training the

[2] `https://github.com/YerevaNN/mimic3-benchmarks`

[3] `https://www.cdc.gov/nchs/icd/icd9.htm`

[4] `https://www.cms.gov/medicare/coding-billing/icd-10-codes/2018-icd-10-cm-gem`

Table 6.1: Performance of the single task backbone.

Task	ROC	AVP
Acute and unspecified renal failure	0.7827	0.5647
Acute cerebrovascular disease	0.9079	0.4578
Acute myocardial infarction	0.7226	0.1761
Cardiac dysrhythmias	0.6948	0.5168
Chronic kidney disease	0.7296	0.4383
Chronic obstructive pulmonary disease and bronchiectasis	0.6791	0.2689
Complications of surgical procedures or medical care	0.7229	0.4045
Conduction disorders	0.6712	0.1880
Congestive heart failure; nonhypertensive	0.7601	0.5129
Coronary atherosclerosis and other heart disease	0.7351	0.5589
Diabetes mellitus with complications	0.8844	0.5559
Diabetes mellitus without complication	0.7484	0.3355
Disorders of lipid metabolism	0.6730	0.5816
Essential hypertension	0.6298	0.5258
Fluid and electrolyte disorders	0.7396	0.6129
Gastrointestinal hemorrhage	0.7076	0.1281
Hypertension with complications and secondary hypertension	0.7141	0.4243
Other liver diseases	0.6849	0.2303
Other lower respiratory disease	0.6371	0.1417
Other upper respiratory disease	0.7602	0.2228
Pleurisy; pneumothorax; pulmonary collapse	0.7051	0.1417
Pneumonia	0.8171	0.3786
Respiratory failure; insufficiency; arrest (adult)	0.8651	0.5497
Septicemia (except in labor)	0.8291	0.4866
Shock	0.8792	0.5574

surrogate model, we use the batch size of 5 and learning rate of $5e-5$. During searching, we compute all multi-task gains on the validation set for guiding the surrogate model training. After we obtain the optimal configuration, we train the searched models and report their multi-task gains on the test set.

Baselines. To compare the proposed method with existing work, we choose several state-of-art-baselines, including both *hand-crafted* and *automated* methods. Specifically, as described below, we include several human-designed EHR encoders to compare with the searched architecture we defined in Eq. (6.2). Also, we include one NAS method and one multi-task grouping method as the automated baselines. More importantly, we combine the multi-task grouping method with the NAS method and hand-crafted encoders to show the superiority of our joint optimization method.

Table 6.2: Hyperparameter setting.

Parameters		**Task @ 5**	**Task @ 10**	**Task @ 25**
# of tasks	N	5	10	25
Dimension of $F(\cdot)$	d_s	64	64	64
# of nodes	P	2	2	3
Progressive sampling	Q_0	10	10	20
	Q_1	50	100	100
	Q_2	10	20	20
	λ	0.5	0.5	0.5
	K_1	20	30	25
Greedy search	K_2	1000	1000	1000
	B	3	5	10
Runtime	GPU Hours	~ 20	~ 75	~ 200

- EHR encoders: We choose four models that are widely utilized for analyzing EHR time series, including LSTM [20], Transformer [21], Retain [4] and Adacare [8].
- NAS: We choose DARTS [49] as the NAS baseline, which is a differentiable search method for efficient architecture search. We apply it to our search space $\mathcal{A}$ to find better EHR encoders. Several state-of-the-art works in other domains have also used it to find MTL architectures [94–96, 98].
- Multi-task grouping: MTG-Net [91] is the current state-of-the-art multi-task grouping algorithm, which uses a meta learning approach to learn the high-order relationships among different tasks. We refer to this method as MTG in latter sections.

Evaluation Metric. We use two widely used metrics for binary classification to evaluate our method and baselines: **ROC** (Area Under the Receiver Operating Characteristic curve) and **AVP** (Averaged Precision). During surrogate model training, we use **AVP** as the metric to compute multi-task gains as in Eq. (6.1), since it is a more suitable choice for considering the class imbalance.

6.4.2 Performance Evaluation

We show our results in Table 6.3. Each experiment is run five times and the average of the runs are reported. We run three settings: **Task @ 5**, **Task @ 10** and **Task @ 25**, which refers to using the first 5 tasks, 10 tasks and 25 tasks respectively. Since grouping all 25 tasks takes a long time to run, we include two small settings that only have the first 5 or 10 tasks in Table 6.1 for grouping. Our results demonstrate our hypotheses:

(a) applying Retain, Adacare, and DARTS improves over the single-task setting, (b) applying different NAS models for each group further improves the performance, and finally, (c) `AutoDP` provides the best results in terms of averaged per-task gain for **ROC** and **AVP**, a significant improvement over existing MTL frameworks for EHR data.

Table 6.3: Performance comparison in terms of averaged per-task gain over single task backbone (All results are in the form of percentage values %).

Settings	**Included Tasks**	**Tasks @ 5**		**Tasks @ 10**		**Tasks @ 25**	
	Metric	**ROC**	**AVP**	**ROC**	**AVP**	**ROC**	**AVP**
One model for all tasks	LSTM	+0.09	+0.18	+1.06	+3.22	+1.83	+7.46
	Transformer	+0.97	+4.82	+1.41	+4.14	+1.75	+7.45
	Retain	+0.46	+1.80	+0.66	+0.75	+1.41	+5.88
	Adacare	+1.03	+5.21	+1.32	+4.05	+1.68	+6.94
	DARTS	+1.28	+5.01	+2.01	+6.87	+1.87	+7.71
Task Grouping + One model for each group	MTG+LSTM	+0.51	+2.10	+0.65	+1.87	+1.74	+7.40
	MTG+Transformer	+0.91	+3.64	+1.20	+3.95	+1.79	+9.15
	MTG+Retain	+0.55	+3.11	+1.51	+5.20	+1.54	+8.87
	MTG+Adacare	+1.25	+5.78	+1.44	+4.63	+1.75	+7.84
	MTG+DARTS	+1.47	+6.41	+2.02	+6.65	+2.41	+11.76
Variants of `AutoDP`	`AutoDP`$^{\mu}$	+1.49	+7.12	+2.08	+7.53	+2.68	+12.70
	`AutoDP`$^{\sigma}$	**+1.95**	+7.68	+2.49	+8.45	+2.62	+13.37
	`AutoDP`$^{\mu+\sigma}$	+1.69	**+7.74**	**+2.55**	**+8.81**	**+2.80**	**+13.43**
	(std)	± 0.08	± 0.25	± 0.13	± 0.29	± 0.12	± 0.33
	(p-value)	0.045	0.029	0.036	0.045	0.027	0.032

First, without considering task grouping, we train one shared model to predict for all tasks in three settings and compute the multi-task gains for them. Results show that this setting only provides minimal improvement over single task training. Note that the automated method (DARTS) performs better than other hand-crafted methods. We also see that sequential optimization over task grouping and architecture search (MTG+DARTS) performs better than MTG + other hand-crafted encoders.

Moreover, we see that the three variants of `AutoDP` performed better than the other methods. Among them, `AutoDP`$^{\mu+\sigma}$ performs the best, which means the balance of exploration and exploitation is the most effective strategy for training the surrogate model. For the last method, we also report the standard deviations and p-values of statistical tests (compared to MTG+DARTS), which justifies that the improvement is significant. The runtime is approximately as the same for MTG+DARTS and `AutoDP` and thus this is a fair comparison. Please refer to Table 6.2 for the GPU hours.

Beside the overall performance gain, we also look at the distribution of performance

gains for each individual task as shown in Figure 6.2. We can observe that the proposed method does not have the issue of negative transfer, since all tasks have a positive gain. Also, for some of the tasks, it can achieve over 20% improvement, which further shows the effectiveness of `AutoDP`.

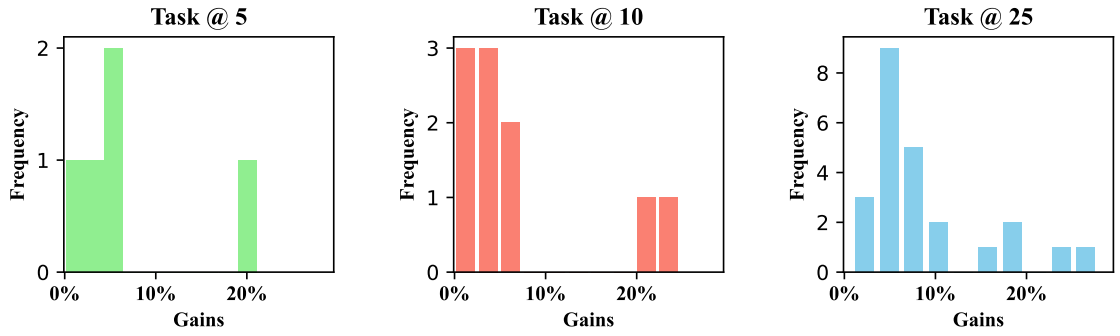


Figure 6.2: Histogram of task gains for `AutoDP` in terms of Averaged Precision.

6.4.3 Hyperparameter & Complexity Analysis

We analyze the effect of two vital hyperparameters of our method: K_1 and B, since they are the crucial parameters that largely define the complexity of our method during searching and inference respectively. We choose the setting of **Task @ 25** for a comprehensive analysis of all tasks. We try out different values and report the corresponding performance gain (**AVP**) in Figure 6.3.

First, K_1 determines the number of training samples collected during searching. Given that each sample invokes an MTL procedure, it constitutes the major portion of the

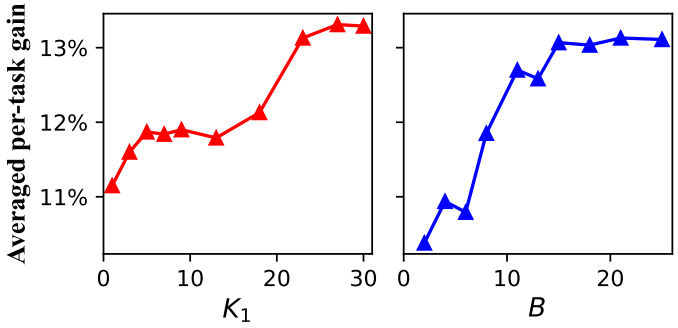


Figure 6.3: Analysis for the number of progressive sampling rounds K_1 and the budget of task groups B under the setting of Task @ 25.

Table 6.4: Ablation results in terms of **AVP**.

Settings	Task @ 5	Task @ 10	Task @ 25
AutoDP	**+7.74**	**+8.81**	**+13.43**
Random sampling	+6.75	+7.04	+11.30
Random search	+6.89	+7.15	+12.04
Disease grouping	+6.29	+6.99	+8.61

search cost. Therefore, our goal is to find an optimal value for K_1, striking a balance between cost-effectiveness and achieving commendable performance. We notice that the performance change plateaus after K_1 reaches 25. That is, the surrogate model effectively learns the distribution of the search space after consuming 25×25 training samples during active selection (25 samples per round). Consequently, we can empirically decide to halt the iteration at this point.

Second, B determines the number of task groups for the final configuration, which indicates the number of MTL models required for achieving the expected performance gain after searching. We also observe similar phenomenon that the performance becomes stable after B reaches 12. We could also choose the optimal value for B accordingly.

6.4.4 Ablation Study

We further analyze the effect of several components within AutoDP, including progressive sampling (Section 6.3.4), greedy search (Section 6.3.5), and task grouping as a whole. We replace these components with naive or human intuition-inspired baselines and report the performances in Table 6.4. Removing any of the components from the original framework leads to noticeable performance decreases, demonstrating the effectiveness of the designed components.

Specifically, we replace progressive sampling and greedy search with purely random methods, referred to as Random Sampling and Random Search. In all three settings, performance generally decreases, highlighting the contributions of these components of AutoDP.

Additionally, we use disease-based grouping (Section6.4.5) to first assign tasks into different groups based on their medical relevance and then employ DARTS to search for the model architecture for each group. This allows us to analyze the effectiveness of automated task grouping. By comparing disease-based grouping with the searched configurations (Section 6.4.6), we observe that AutoDP does not strictly follow medical

Table 6.5: Disease Based Grouping.

Groups	Diseases
Cardiovascular Diseases	Acute cerebrovascular disease Acute myocardial infarction Cardiac dysrhythmias Congestive heart failure; nonhypertensive Coronary atherosclerosis and other heart disease Essential hypertension
Respiratory Diseases	Chronic obstructive pulmonary disease and bronchiectasis Other lower respiratory disease Other upper respiratory disease Pleurisy; pneumothorax; pulmonary collapse Pneumonia (except that caused by tuberculosis or sexually transmitted disease) Respiratory failure; insufficiency; arrest (adult)
Kidney Diseases	Acute and unspecified renal failure Chronic kidney disease
Metabolic Diseases	Diabetes mellitus with complications Diabetes mellitus without complication Disorders of lipid metabolism Fluid and electrolyte disorders
Gastrointestinal Diseases	Gastrointestinal hemorrhage
Infections	Septicemia (except in labor)
Surgical/Medical Complications	Complications of surgical procedures or medical care
Neurological/Cardiac Conditions	Conduction disorders Shock
Liver Diseases	Other liver diseases

classifications for task grouping but achieves significant performance improvements over disease-based grouping. This indicates the necessity of using an automated search algorithm to find the optimal task grouping, which surpasses human intuition.

6.4.5 Disease Based Grouping

To show the effectiveness of automated task grouping, we conduct experiments using a predefined task grouping based on disease categories. We asked GPT-4 [107] to classify the 25 prediction tasks into different groups based on their medical meaning. The result is shown in Table 6.5. Using this grouping, we further apply DARTS to each group and report the multi-task gains as shown in Table 6.4. Compared to `AutoDP`, there is a notable performance drop for the disease based grouping. This means human intuition dose not provide the optimal task grouping, which underscores the necessity of employing search algorithm to automatically discover better task grouping for MTL.

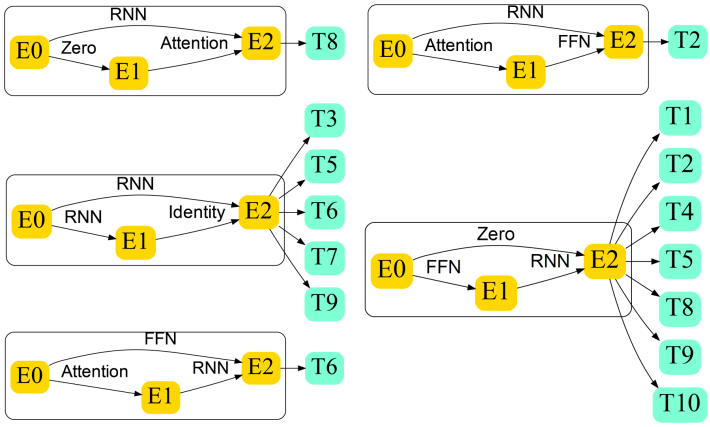


Figure 6.4: Illustration of the searched configuration under the setting of Task @ 10.

6.4.6 Visualization of the Searched Configurations

Here, we show two example of the final configuration for setting **Task @ 10** in Figure 6.4 and for **Task @ 25** in Figure 6.5. The proposed `AutoDP` identifies 5 and 10 different task groups respectively and also searches for the corresponding architectures. We can observe that some of the tasks tend to be trained independently, while others are grouped together for joint training. This supports our claim that fine-grained task grouping is necessary to bring the optimal performance gain. Also, the optimal architecture is also different for each task group, which further justifies the necessity of joint optimization over task grouping and architecture search.

6.5 Limitation and Future Work

There are some valuable future directions based on the current version of `AutoDP`.

First, from the application perspective, if we aim at deploying `AutoDP` to real-world healthcare systems, it would be advantageous to apply it to more complex problem settings. For example, the incorporation of diverse clinical data sources beyond EHR such as claims, drugs, medical images and texts will significantly enhance the practical utility of `AutoDP`.

Additionally, considering the dynamic nature of healthcare environments with continuously updated input data and evolving tasks, adapting the surrogate model to accommodate new data and tasks would be imperative.

Moreover, addressing privacy concerns within healthcare systems is a promising direction. Therefore, extending `AutoDP` with data processing pipelines for automatic

feature engineering could offer enhanced privacy safeguards and further improve its applicability in sensitive healthcare contexts.

Finally, we assume all the tasks have the same input EHR data in our problem setting, which might not always be the case in practical scenarios. Chances are that, for some diseases, there are large and well-annotated data, while for the others, there are limited data available. How we should extend the current framework to handle more heterogeneous diseases/tasks remains a challenge.

6.6 Conclusion

In this paper, we propose `AutoDP`, an automated multi-task learning framework for joint disease prediction on EHR data. Compared to existing work, our method largely improves the design of task grouping and model architectures by reducing human interventions. Experimental results on real-world EHR data demonstrate that the proposed framework achieves significant improvement over existing state-of-the-art methods, while maintaining a feasible search cost.

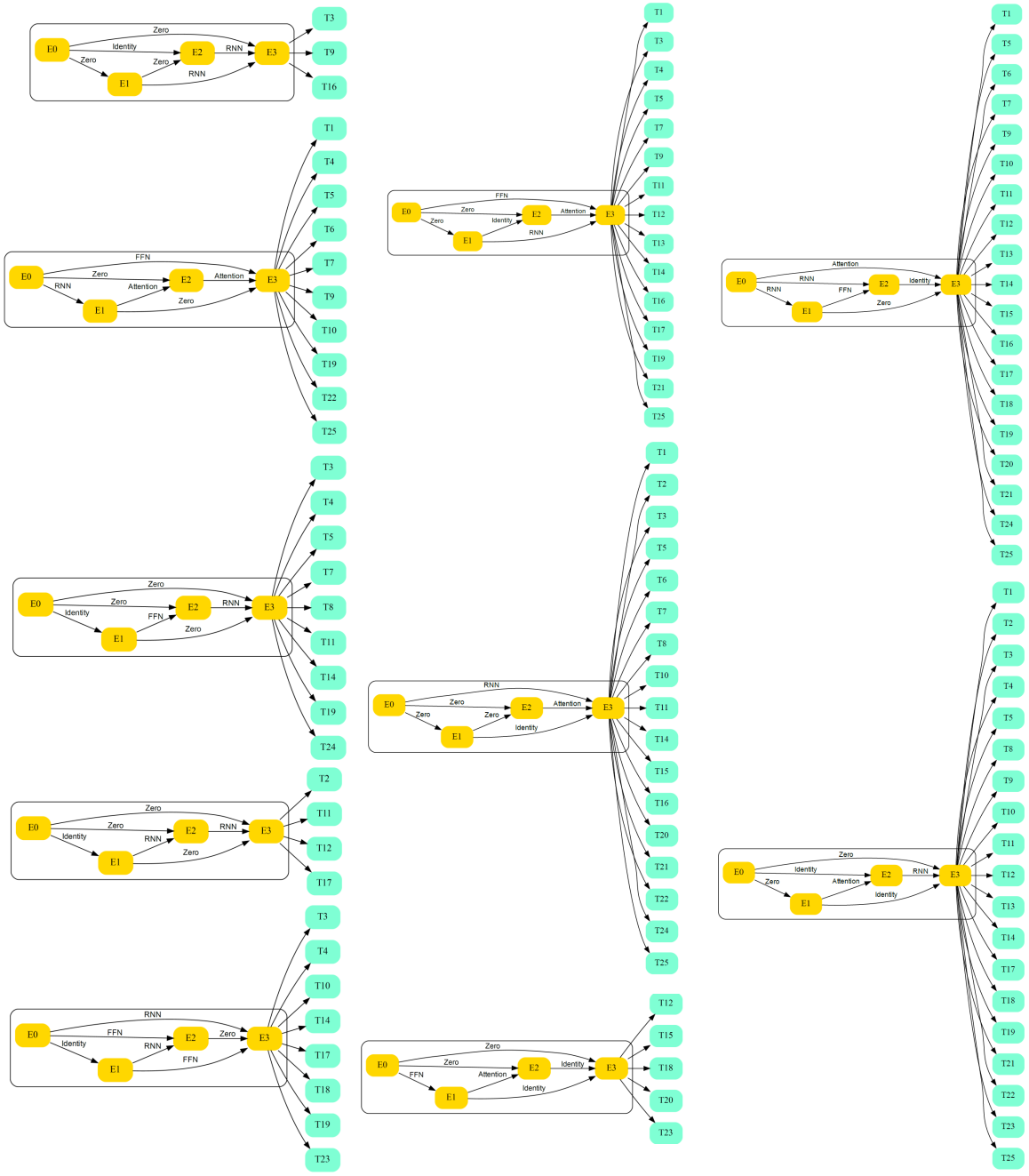


Figure 6.5: Illustration of the searched configuration under the setting of Task @ 25.

Chapter 7
Conclusion and Future Directions

7.1 Conclusion

In this dissertation, we explored the challenges and opportunities associated with predictive modeling on EHR data and proposed innovative methodologies to address them. By transitioning from handcrafted approaches to automated frameworks, we sought to reduce reliance on human expertise, improve model performance, and enhance generalizability across diverse predictive tasks.

The handcrafted methods, `MedPath` and `MedRetriever`, demonstrated the potential of integrating external medical knowledge to address domain-specific limitations. These methods not only improved predictive accuracy but also enhanced interpretability, offering human-readable explanations through knowledge graphs and natural language processing techniques. These contributions underscored the importance of leveraging domain knowledge to create more robust and interpretable models.

- `MedPath`: In this work, we proposed an model-agnostic framework that leverages medical knowledge graphs (KG) to improve existing prediction models, enhancing both their performance and interpretability.
- `MedRetriever`: In this work, we proposed an retrieval-based text augmentation framework that can improve any existing prediction models on EHR data using medical texts.

Furthermore, we introduced automated frameworks that leveraged neural architecture search to address the complexities of multi-modal EHR data and multi-task learning. `AutoMed` and `AutoFM` showcased the power of automation in designing model architectures for multi-modal data fusion, significantly reducing the need for manual intervention while

achieving superior predictive performance. Similarly, `AutoDP` addressed the challenges of multi-task learning by automating both architecture design and task grouping, paving the way for more scalable and generalizable predictive solutions.

- `AutoMed`: In this work, we developed an automated machine learning framework that focus on the design of multi-modal architectures for predictive modeling on EHR data. We focus on fusing time features and diagnosis features in this work.
- `AutoFM`: Following `AutoMed`, we further proposes a more general framework that can automated the design of multi-modal architectures, which can incorporate more modalities, including patient demographic, clinical notes, and ICU monitoring data.
- `AutoDP`: In this work, we proposed an automated multi-task learning framework for joint disease prediction on EHR data. Our framework can jointly perform task grouping and architecture search that maximizes the overall multi-task gains.

Collectively, these contributions represent a significant step forward in advancing machine learning methodologies for EHR data analysis. By addressing the limitations of traditional approaches and proposing automated solutions, this work lays the foundation for future research that bridges the gap between machine learning and healthcare. As the field continues to evolve, the methodologies presented here can serve as a blueprint for developing scalable, interpretable, and high-performing models that drive progress in personalized medicine, disease prediction, and healthcare delivery.

7.2 Future Directions

Despite the success of the current AutoML methodologies for predictive modeling on EHR data, they still suffer from the following limitations. First, there is the **trade-off between effectiveness and efficiency**. AutoML systems often incur high computational costs to achieve effective solutions, significantly limiting their practicality in real-world scenarios. Second, the **interpretability** of AutoML systems remains a challenge. It is often difficult to provide clear reasoning behind why the solutions generated perform well—or fail to do so. Finally, current AutoML methods face limitations in **adapting to new tasks**. In practice, multiple target diseases may need to be predicted, but these targets are not always available from the beginning. Current methods like `AutoDP` are typically designed to handle a predefined set of target diseases, which restricts their adaptability to emerging tasks.

To address the challenges above, Large Language Models (LLMs) can be a promising solution [108]. With their strong reasoning capabilities, LLMs can act as powerful optimization tools through their textual interfaces and have demonstrated their potential in a variety of reasoning tasks [109]. Additionally, LLMs have been successfully applied in the AutoML domain, including tasks such as neural architecture search [110] and hyperparameter optimization [111]. Hence, we have the potential to enhance the current AutoML system to solve the aforementioned challenges. By harnessing the reasoning capabilities and rich prior knowledge embedded in LLMs through pretraining, it enhances both the effectiveness and efficiency of the AutoML system. Furthermore, the textual interfaces of LLMs enable the frame- work to deliver human-understandable explanations for its solutions, thereby improving interpretability. Also, the LLM based optimization framework can be designed to be flexible at handling evolving tasks through textual interfaces.

Bibliography

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

[2] Johnson, A., L. Bulgarelli, T. Pollard, S. Horng, L. A. Celi, and R. Mark (2020) "Mimic-iv," *PhysioNet. Available online at: https://physionet.org/content/mimiciv/1.0/(accessed August 23, 2021)*.

[3] Ma, F., M. Ye, J. Luo, C. Xiao, and J. Sun (2021) "Advances in Mining Heterogeneous Healthcare Data," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 4050–4051.

[4] Choi, E., M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. Stewart (2016) "Retain: An interpretable predictive model for healthcare using reverse time attention mechanism," in *Advances in Neural Information Processing Systems*, pp. 3504–3512.

[5] Ma, F., R. Chitta, J. Zhou, Q. You, T. Sun, and J. Gao (2017) "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.

[6] Kwon, B. C., M.-J. Choi, J. T. Kim, E. Choi, Y. B. Kim, S. Kwon, J. Sun, and J. Choo (2018) "Retainvis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records," *IEEE transactions on visualization and computer graphics*, **25**(1), pp. 299–309.

[7] Bai, T., S. Zhang, B. L. Egleston, and S. Vucetic (2018) "Interpretable representation learning for healthcare via capturing disease progression through time," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 43–51.

[8] Ma, L., J. Gao, Y. Wang, C. Zhang, J. Wang, W. Ruan, W. Tang, X. Gao, and X. Ma (2020) "AdaCare: Explainable Clinical Health Status Representation Learning via Scale-Adaptive Feature Extraction and Recalibration," in *AAAI*.

[9] Ma, F., Y. Wang, J. Gao, H. Xiao, and J. Zhou (2020) "Rare Disease Prediction by Generating Quality-Assured Electronic Health Records," in *Proceedings of the 2020 SIAM International Conference on Data Mining*, SIAM, pp. 514–522.

[10] Song, H., D. Rajan, J. Thiagarajan, and A. Spanias (2018) "Attend and diagnose: Clinical time series analysis using attention models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32.

[11] Luo, J., M. Ye, C. Xiao, and F. Ma (2020) "HiTANet: Hierarchical Time-Aware Attention Networks for Risk Prediction on Electronic Health Records," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 647–656.

[12] Ye, M., J. Luo, C. Xiao, and F. Ma (2020) "LSAN: Modeling Long-term Dependencies and Short-term Correlations with Hierarchical Attention for Risk Prediction," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*.

[13] Choi, E., M. T. Bahadori, L. Song, W. F. Stewart, and J. Sun (2017) "GRAM: graph-based attention model for healthcare representation learning," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 787–795.

[14] Ma, F., Q. You, H. Xiao, R. Chitta, J. Zhou, and J. Gao (2018) "Kame: Knowledge-based attention model for diagnosis prediction in healthcare," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ACM, pp. 743–752.

[15] Yin, C., R. Zhao, B. Qian, X. Lv, and P. Zhang (2019) "Domain Knowledge guided deep learning with electronic health records," in *2019 IEEE International Conference on Data Mining (ICDM)*, IEEE, pp. 738–747.

[16] Ma, F., J. Gao, Q. Suo, Q. You, J. Zhou, and A. Zhang (2018) "Risk prediction on electronic health records with prior medical knowledge," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1910–1919.

[17] Ma, F., Y. Wang, H. Xiao, Y. Yuan, R. Chitta, J. Zhou, and J. Gao (2018) "A general framework for diagnosis prediction via incorporating medical code descriptions," in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, pp. 1070–1075.

[18] Ernst, P., A. Siu, and G. Weikum (2015) "Knowlife: a versatile approach for constructing a large knowledge graph for biomedical sciences," *BMC bioinformatics*, **16**(1), p. 157.

[19] Chung, J., C. Gulcehre, K. Cho, and Y. Bengio (2014) "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*.

[20] Hochreiter, S. and J. Schmidhuber (1997) "Long short-term memory," *Neural computation*, **9**(8), pp. 1735–1780.

[21] Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017) "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008.

[22] Baytas, I. M., C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou (2017) "Patient subtyping via time-aware lstm networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 65–74.

[23] Chen, C., J. Liang, F. Ma, L. Glass, J. Sun, and C. Xiao (2021) "Unite: Uncertainty-based health risk prediction leveraging multi-sourced data," in *Proceedings of the Web Conference 2021*, pp. 217–226.

[24] Kipf, T. N. and M. Welling (2016) "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*.

[25] Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio (2017) "Graph attention networks," *arXiv preprint arXiv:1710.10903*.

[26] Schlichtkrull, M., T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling (2018) "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*, Springer, pp. 593–607.

[27] Ye, M., S. Cui, Y. Wang, J. Luo, C. Xiao, and F. Ma (2021) "MedPath: Augmenting Health Risk Prediction via Medical Knowledge Paths," *Proceedings of the Web Conference 2021*.

[28] Feng, Y., X. Chen, B. Y. Lin, P. Wang, J. Yan, and X. Ren (2020) "Scalable Multi-Hop Relational Reasoning for Knowledge-Aware Question Answering," *arXiv preprint arXiv:2005.00646*.

[29] Kilicoglu, H., M. Fiszman, A. Rodriguez, D. Shin, A. Ripple, and T. C. Rindflesch (2008) "Semantic MEDLINE: a web application for managing the results of PubMed Searches," in *Proceedings of the third international symposium for semantic mining in biomedicine*, vol. 2008, pp. 69–76.

[30] Rindflesch, T. C., H. Kilicoglu, M. Fiszman, G. Rosemblat, and D. Shin (2011) "Semantic MEDLINE: An advanced information management application for biomedicine," *Information Services & Use*, **31**(1-2), pp. 15–21.

[31] Donnelly, K. (2006) "SNOMED-CT: The advanced terminology and coding system for eHealth," *Studies in health technology and informatics*, **121**, p. 279.

[32] Sedgewick, R. (2001) *Algorithms in c, part 5: graph algorithms*, Pearson Education.

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

[34] Pham, T., T. Tran, D. Phung, and S. Venkatesh (2016) "Deepcare: A deep dynamic memory model for predictive medicine," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, pp. 30–41.

[35] Bordes, A., N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko (2013) "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, pp. 2787–2795.

[36] Lafferty, J. D., A. McCallum, and F. C. N. Pereira (2001) "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 282–289.
URL `http://dl.acm.org/citation.cfm?id=645530.655813`

[37] Santos, C. d., M. Tan, B. Xiang, and B. Zhou (2016) "Attentive pooling networks," *arXiv preprint arXiv:1602.03609*.

[38] Song, H., D. Rajan, J. J. Thiagarajan, and A. Spanias (2017) "Attend and diagnose: Clinical time series analysis using attention models," *arXiv preprint arXiv:1711.03905*.

[39] Le, H., T. Tran, and S. Venkatesh (2018) "Dual memory neural computer for asynchronous two-view sequential learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1637–1645.

[40] Shang, J., C. Xiao, T. Ma, H. Li, and J. Sun (2019) "Gamenet: Graph augmented memory networks for recommending medication combination," in *proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1126–1133.

[41] Ye, M., S. Cui, Y. Wang, J. Luo, C. Xiao, and F. Ma (2021) "Medretriever: Target-driven interpretable health risk prediction via retrieving unstructured medical text," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 2414–2423.

[42] Levenshtein, V. I. (1966) "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, pp. 707–710.

[43] Gu, Y., R. Tinn, H. Cheng, M. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao, and H. Poon (2020) "Domain-specific language model pretraining for biomedical natural language processing," *arXiv preprint arXiv:2007.15779*.

[44] Miotto, R., F. Wang, S. Wang, X. Jiang, and J. T. Dudley (2018) "Deep learning for healthcare: review, opportunities and challenges," *Briefings in bioinformatics*, **19**(6), pp. 1236–1246.

[45] Alaa, A. and M. Schaar (2018) "Autoprognosis: Automated clinical prognostic modeling via bayesian optimization with structured kernel learning," in *ICML*, PMLR, pp. 139–148.

[46] Jarrett, D., J. Yoon, I. Bica, Z. Qian, A. Ercole, and M. van der Schaar (2020) "Clairvoyance: A pipeline toolkit for medical time series," in *International Conference on Learning Representations*.

[47] Zoph, B. and Q. V. Le (2016) "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*.

[48] Pham, H., M. Guan, B. Zoph, Q. Le, and J. Dean (2018) "Efficient neural architecture search via parameters sharing," in *International conference on machine learning*, PMLR, pp. 4095–4104.

[49] Liu, H., K. Simonyan, and Y. Yang (2019) "DARTS: Differentiable Architecture Search," in *International Conference on Learning Representations*.

[50] Liu, C., B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy (2018) "Progressive neural architecture search," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 19–34.

[51] Hutter, F., H. H. Hoos, and K. Leyton-Brown (2011) "Sequential model-based optimization for general algorithm configuration," in *International conference on learning and intelligent optimization*, Springer, pp. 507–523.

[52] Kandasamy, K., W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing (2018) "Neural architecture search with bayesian optimisation and optimal transport," *Advances in neural information processing systems*, **31**.

[53] Elsken, T., J. H. Metzen, and F. Hutter (2019) "Neural architecture search: A survey," *The Journal of Machine Learning Research*, **20**(1), pp. 1997–2017.

[54] Suganuma, M., M. Ozay, and T. Okatani (2018) "Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search," in *ICML*, PMLR, pp. 4771–4780.

[55] Chen, L.-C., M. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens (2018) "Searching for efficient multi-scale architectures for dense image prediction," *NeurIPS*, **31**.

[56] Liu, C., L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei (2019) "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *CVPR*, pp. 82–92.

[57] Real, E., A. Aggarwal, Y. Huang, and Q. V. Le (2019) "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, pp. 4780–4789.

[58] Zoph, B., V. Vasudevan, J. Shlens, and Q. V. Le (2018) "Learning transferable architectures for scalable image recognition," in *CVPR*, pp. 8697–8710.

[59] Klyuchnikov, N., I. Trofimov, E. Artemova, M. Salnikov, M. Fedorov, and E. Burnaev (2020) "NAS-Bench-NLP: neural architecture search benchmark for natural language processing," *arXiv preprint arXiv:2006.07116*.

[60] Wang, Y., Y. Yang, Y. Chen, J. Bai, C. Zhang, G. Su, X. Kou, Y. Tong, M. Yang, and L. Zhou (2020) "Textnas: A neural architecture search space tailored for text representation," in *AAAI*, vol. 34, pp. 9242–9249.

[61] Yu, Z., Y. Cui, J. Yu, M. Wang, D. Tao, and Q. Tian (2020) "Deep multimodal neural architecture search," in *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 3743–3752.

[62] Peng, Y., L. Bi, M. Fulham, D. Feng, and J. Kim (2020) "Multi-modality information fusion for radiomics-based neural architecture search," in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part VII 23*, Springer, pp. 763–771.

[63] Yin, Y., S. Huang, X. Zhang, and D. Dou (2022) "BM-NAS: Bilevel Multimodal Neural Architecture Search," in *AAAI*.

[64] Choi, E., C. Xiao, W. F. Stewart, and J. Sun (2018) "MiME: multilevel medical embedding of electronic health records for predictive healthcare," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 4552–4562.

[65] Cho, K., B. Van Merriënboer, D. Bahdanau, and Y. Bengio (2014) "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*.

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

[67] Choi, E., M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun (2016) "Doctor ai: Predicting clinical events via recurrent neural networks," in *Machine learning for healthcare conference*, PMLR, pp. 301–318.

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

[69] Xu, Y., S. Biswal, S. R. Deshpande, K. O. Maher, and J. Sun (2018) "Raim: Recurrent attentive and intensive model of multimodal patient monitoring data," in *Proceedings of the 24th ACM SIGKDD international conference on Knowledge Discovery & Data Mining*, pp. 2565–2573.

[70] Feng, Y., Z. Xu, L. Gan, N. Chen, B. Yu, T. Chen, and F. Wang (2019) "Dcmn: Double core memory network for patient outcome prediction with multimodal data," in *2019 IEEE International Conference on Data Mining (ICDM)*, IEEE, pp. 200–209.

[71] Qiao, Z., X. Wu, S. Ge, and W. Fan (2019) "Mnn: multimodal attentional neural networks for diagnosis prediction," *Extraction*, **1**, p. A1.

[72] Yang, B. and L. Wu (2021) "How to Leverage Multimodal EHR Data for Better Medical Predictions?" *arXiv preprint arXiv:2110.15763*.

[73] Xu, Z., D. R. So, and A. M. Dai (2021) "Mufasa: Multimodal fusion architecture search for electronic health records," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 10532–10540.

[74] Cui, S., J. Wang, X. Gui, T. Wang, and F. Ma (2022) "AUTOMED: Automated Medical Risk Predictive Modeling on Electronic Health Records," in *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, pp. 948–953.

[75] Zela, A., T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter (2019) "Understanding and robustifying differentiable architecture search," *arXiv preprint arXiv:1909.09656*.

[76] Wang, R., M. Cheng, X. Chen, X. Tang, and C.-J. Hsieh (2021) "Rethinking architecture selection in differentiable nas," *arXiv preprint arXiv:2108.04392*.

[77] Zhang, X., S. Li, Z. Chen, X. Yan, and L. Petzold (2022) "Improving Medical Predictions by Irregular Multimodal Electronic Health Records Modeling," *arXiv preprint arXiv:2210.12156*.

[78] Cui, S., J. Wang, Y. Zhong, H. Liu, T. Wang, and F. Ma (2024) "Automated fusion of multimodal electronic health records for better medical predictions," in *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*, SIAM, pp. 361–369.

[79] Tang, S., P. Davarmanesh, Y. Song, D. Koutra, M. W. Sjoding, and J. Wiens (2020) "Democratizing EHR analyses with FIDDLE: a flexible data-driven preprocessing pipeline for structured clinical data," *Journal of the American Medical Informatics Association*, **27**(12), pp. 1921–1934.

[80] Gu, J., Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al. (2018) "Recent advances in convolutional neural networks," *Pattern recognition*, **77**, pp. 354–377.

[81] Guo, Q., X. Qiu, P. Liu, Y. Shao, X. Xue, and Z. Zhang (2019) "Star-Transformer," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, Minneapolis, Minnesota, pp. 1315–1325.
URL https://aclanthology.org/N19-1133

[82] Rahman, W., M. K. Hasan, S. Lee, A. Zadeh, C. Mao, L.-P. Morency, and E. Hoque (2020) "Integrating multimodal information in large pretrained transformers," in *Proceedings of the conference. Association for Computational Linguistics. Meeting*, vol. 2020, NIH Public Access, p. 2359.

[83] Suo, Q., F. Ma, G. Canino, J. Gao, A. Zhang, P. Veltri, and G. Agostino (2017) "A multi-task framework for monitoring health conditions via attention-based recurrent neural networks," in *AMIA annual symposium proceedings*, vol. 2017, American Medical Informatics Association, p. 1665.

[84] Razavian, N., J. Marcus, and D. Sontag (2016) "Multi-task prediction of disease onsets from longitudinal laboratory tests," in *Machine learning for healthcare conference*, PMLR, pp. 73–100.

[85] Wang, X., F. Wang, J. Hu, and R. Sorrentino (2014) "Exploring joint disease risk prediction," in *AMIA Annual Symposium Proceedings*, vol. 2014, American Medical Informatics Association, p. 1180.

[86] Zhao, X., X. Wang, F. Yu, J. Shang, and S. Peng (2022) "UniMed: Multi-modal Multitask Learning for Medical Predictions," in *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, pp. 1399–1404.

[87] Xu, E., S. Zhao, J. Mei, E. Xia, Y. Yu, and S. Huang (2019) "Multiple MACE risk prediction using multi-task recurrent neural network with attention," in *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, IEEE, pp. 1–2.

[88] Standley, T., A. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese (2020) "Which tasks should be learned together in multi-task learning?" in *International Conference on Machine Learning*, PMLR, pp. 9120–9132.

[89] He, X., K. Zhao, and X. Chu (2021) "AutoML: A survey of the state-of-the-art," *Knowledge-Based Systems*, **212**, p. 106622.

[90] Fifty, C., E. Amid, Z. Zhao, T. Yu, R. Anil, and C. Finn (2021) "Efficiently identifying task groupings for multi-task learning," *Advances in Neural Information Processing Systems*, **34**, pp. 27503–27516.

[91] Song, X., S. Zheng, W. Cao, J. Yu, and J. Bian (2022) "Efficient and effective multi-task grouping via meta learning on task combinations," *Advances in Neural Information Processing Systems*, **35**, pp. 37647–37659.

[92] Ahn, C., E. Kim, and S. Oh (2019) "Deep elastic networks with model selection for multi-task learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6529–6538.

[93] Bragman, F. J., R. Tanno, S. Ourselin, D. C. Alexander, and J. Cardoso (2019) "Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1385–1394.

[94] Sun, X., R. Panda, R. Feris, and K. Saenko (2020) "Adashare: Learning what to share for efficient deep multi-task learning," *Advances in Neural Information Processing Systems*, **33**, pp. 8728–8740.

[95] Guo, P., C.-Y. Lee, and D. Ulbricht (2020) "Learning to branch for multi-task learning," in *International conference on machine learning*, PMLR, pp. 3854–3863.

[96] Gao, Y., H. Bai, Z. Jie, J. Ma, K. Jia, and W. Liu (2020) "Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning," in *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 11543–11552.

[97] Waring, J., C. Lindvall, and R. Umeton (2020) "Automated machine learning: Review of the state-of-the-art and opportunities for healthcare," *Artificial Intelligence in Medicine*, **104**, p. 101822.

[98] Zhang, L., X. Liu, and H. Guan (2022) "Automtl: A programming framework for automating efficient multi-task learning," *Advances in Neural Information Processing Systems*, **35**, pp. 34216–34228.

[99] Liu, S., H. Zhang, and Y. Jin (2022) "A survey on surrogate-assisted efficient neural architecture search," *arXiv preprint arXiv:2206.01520*.

[100] Xiao, C., E. Choi, and J. Sun (2018) "Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review," *Journal of the American Medical Informatics Association*, **25**(10), pp. 1419–1428.

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

[102] He, K., X. Zhang, S. Ren, and J. Sun (2016) "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

[103] Lu, Z., K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti (2020) "Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, Springer, pp. 35–51.

[104] Luo, R., F. Tian, T. Qin, E. Chen, and T.-Y. Liu (2018) "Neural architecture optimization," *Advances in neural information processing systems*, **31**.

[105] Zhang, M., S. Jiang, Z. Cui, R. Garnett, and Y. Chen (2019) "D-vae: A variational autoencoder for directed acyclic graphs," *Advances in Neural Information Processing Systems*, **32**.

[106] Auer, P., N. Cesa-Bianchi, and P. Fischer (2002) "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, **47**(2-3), pp. 235–256.

[107] Achiam, J., S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. (2023) "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*.

[108] Xu, J., J. Li, Z. Liu, N. A. V. Suryanarayanan, G. Zhou, J. Guo, H. Iba, and K. Tei (2024) "Large language models synergize with automated machine learning," *arXiv preprint arXiv:2405.03727*.

[109] Wang, L., C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, et al. (2024) "A survey on large language model based autonomous agents," *Frontiers of Computer Science*, **18**(6), p. 186345.

[110] Chen, A., D. Dohan, and D. So (2024) "EvoPrompting: language models for code-level neural architecture search," *Advances in Neural Information Processing Systems*, **36**.

[111] Liu, T., N. Astorga, N. Seedat, and M. van der Schaar (2024) "Large language models to enhance bayesian optimization," *arXiv preprint arXiv:2402.03921*.

Vita

Suhan Cui

Education

The Pennsylvania State University 2021.08 - 2025.05
Ph.D. in Informatics, Advisor: Prof. Dongwon Lee
Northeastern University, China 2017.08 - 2021.06
B.Eng. in Software Engineering

Selected Publications

1. **Suhan Cui** and Prasenjit Mitra, "Automated Multi-Task Learning for Joint Disease Prediction on Electronic Health Records", Advances in Neural Information Processing Systems (NeurIPS 2024)

2. **Suhan Cui**, Jiaqi Wang, Yuan Zhong, Han Liu, Ting Wang and Fenglong Ma, "Automated Fusion of Multimodal Electronic Health Records for Better Medical Predictions", SIAM International Conference on Data Mining (SDM 2024)

3. Muchao Ye*, **Suhan Cui***, Yaqing Wang, Junyu Luo, Cao Xiao and Fenglong Ma, "MedPath: Augmenting Health Risk Prediction via Medical Knowledge Paths", 2021 World Wide Web Conference (WWW 2021).

4. Muchao Ye*, **Suhan Cui***, Yaqing Wang, Junyu Luo, Cao Xiao and Fenglong Ma, "MedRetriever: Target-Driven Health Risk Prediction via Retrieving Unstructured Medical Text", 30th ACM International Conference on Information and Knowledge Management (CIKM 2021).

5. **Suhan Cui**, Junyu Luo, Muchao Ye, Jiaqi Wang, Ting Wang and Fenglong Ma, "MedSkim: Denoised Health Risk Prediction via Skimming Medical Claims Data", 22nd IEEE International Conference on Data Mining (ICDM 2022).

6. **Suhan Cui**, Jiaqi Wang, Xinning Gui, Ting Wang and Fenglong Ma, "AUTOMED: Automated Medical Risk Predictive Modeling on Electronic Health Records", 2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2022)

7. **Suhan Cui**, Guanhao Wei, Li Zhou, Emily Zhao, Ting Wang and Fenglong Ma, "Predicting Line of Therapy Transition via Similar Patient Augmentation", Journal of Biomedical Informatics, 2023