DOCTOR AI: INTERPRETABLE DEEP LEARNING FOR MODELING ELECTRONIC HEALTH RECORDS

A Dissertation
Presented to
The Academic Faculty

By

Edward Choi

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computational Science and Engineering

Georgia Institute of Technology

August 2018

DOCTOR AI: INTERPRETABLE DEEP LEARNING FOR MODELING ELECTRONIC HEALTH RECORDS

Approved by:

Dr. Jimeng Sun, Advisor
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Jon Duke
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Jacob Eisenstein
School of Interactive Computing
Georgia Institute of Technology

Dr. James Rehg
School of Interactive Computing
Georgia Institute of Technology

Dr. Walter F. Stewart

Date Approved: April 19, 2018

ACKNOWLEDGEMENTS

I would like to sincerely thank my advisor, Dr. Jimeng Sun for giving me an opportunity to learn how to conduct research, communicate with others, and present ideas in writing and orally. The past four years have completely changed my life. I thank the committee members, Dr. Jon Duke, Dr. Jacob Eisenstein, Dr. Jim Rehg and Dr. Walter Stewart for participating in my thesis, and providing insightful comments and helpful feedback. I would like to acknowledge all the help I received from researchers I collaborated with over the past years: Dr. Taha Bahadori, Dr. Cao Xiao, Dr. Andy Schuetz, Dr. Le Song, Dr. Nan Du, Dr. Bradley Malin, Dr. Angeliki Lazaridou, Dr. Nando de Freitas, Dr. Ariel Gordon and Dr. Elad Eban. I thank my school mates, not just for research collaboration and writing papers together, but also for simply hanging out and occasional fun we had: Robert, Kimis, Sungtae, Siddharth, Jeff, Yu, Yanbo, Sandesh, Sarah, Olivier, James, Ari, Yuyu, Hang, Zhaoming, and Kunal. I am very grateful to my personal friends who were a significant part of my PhD life: Ickhyun, Dr. Moonkyu Cho, Joonseok, Sookyung, Seunghyun, Seungyeon, Hanna, Hyojong, Taeheon, Hyemin, Hyoukjun, Hyeokhyun, Eunji and Chanho. My last thanks goes to my family for their love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Deep learning recently has been showing superior performance in complex domains such as computer vision, audio processing and natural language processing compared to traditional statistical methods. Naturally, deep learning techniques, combined with large electronic health records (EHR) data generated from healthcare organizations have potential to bring dramatic changes to the healthcare industry. However, typical deep learning models can be seen as highly expressive blackboxes, making them difficult to be adopted in real-world healthcare applications due to lack of interpretability. In order for deep learning methods to be readily adopted by real-world clinical practices, they must be interpretable without sacrificing their prediction accuracy.

In this thesis, we propose interpretable and accurate deep learning methods for modeling EHR, specifically focusing on longitudinal EHR data. We will begin with a direct application of a well-known deep learning algorithm, recurrent neural networks (RNN), to capture the temporal nature of longitudinal EHR. Then, based on the initial approach we develop interpretable deep learning models by focusing on three aspects of computational healthcare: efficient representation learning of medical concepts, code-level interpretation for sequence predictions, and leveraging domain knowledge into the model. Another important aspect that we will address in this thesis is developing a framework for effectively utilizing multiple data sources (*e.g.* diagnoses, medications, procedures), which can be extended in the future to incorporate wider data modalities such as lab values and clinical notes.

CHAPTER 1
INTRODUCTION

The recent resurgence of neural networks, or deep learning [1], has changed the way we handle data in computational data analytics. Unlike traditional statistical approaches where humans were required to study the data carefully and design useful features, deep learning places the machine in charge of learning useful features (or representations) directly from the data without human intervention. Deep learning typically uses high-capacity neural network models, which is trained by a large number of training samples. Thanks to the advances in computational resources such as graphics processing unit (GPU) and a large volume of labeled datasets, deep learning has shown superior performance compared to traditional approaches in various fields such as computer vision, natural language processing and audio processing [2, 3, 4, 5].

Electronic health records (EHR) since adopted by large healthcare organizations in the last decade, has enabled the accumulation of large electronic patient data. EHR has helped researchers use traditional statistical approaches such as logistic regression or random forests for computational healthcare [6, 7, 8]. However, with the growing size of patient records and the development of powerful computing resources, it seems that now is the right time to introduce deep learning techniques to computational healthcare. However in healthcare, interpretation of the model outcome is vital. Therefore, although typical deep learning models show impressive predictive performance, their blackbox nature makes it difficult for them to be readily used in healthcare. Deep learning models need to be interpretable without sacrificing its prediction accuracy in order to be actively adopted in healthcare.

In this thesis, we propose interpretable deep learning methods for modeling EHR, specifically focusing on predictive modeling and representation learning of longitudinal EHR data. Patients visit hospitals over time, which constitutes a sequence of visit records.

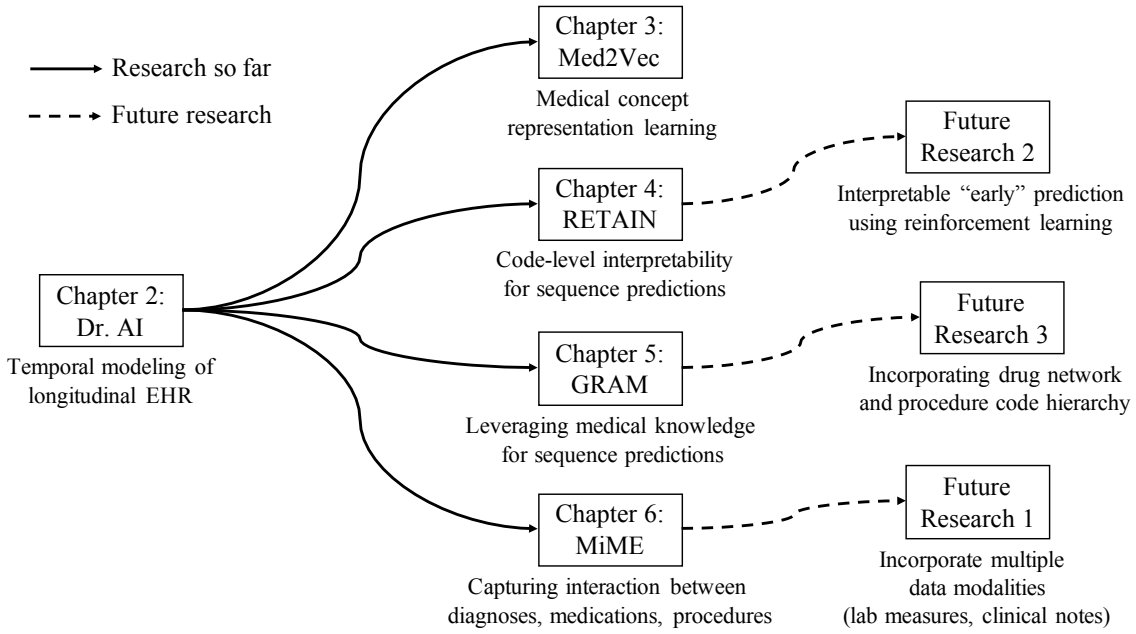


Figure 1.1: Relationship between research conducted so far, and future research directions. Each conducted research will be described in detail in the corresponding chapters.

Each visit (*i.e.* encounter) consists of medical events (*i.e.* medical codes) such as diagnosis codes, medication codes and procedure codes. Therefore it is essential to capture both the medical code relationships within a visit, and the medical code relationships across visits, and be able to provide interpretation of the model to the user. Figure 1.1 depicts how we have tackled this problem so far, and how we will continue our effort in the future.

We began with a direct application of a well-known deep learning algorithm, the recurrent neural networks (RNN), to effectively capture the intra-visit and inter-visit medical code relationships. This was motivated by the successful application of the RNN to predict the onset of heart failure for patients at Sutter Health [9]. This project, named *Dr.AI* [10], aimed to model the temporal progression of patient status. Given all previous visit records of a patient, our model tried to predict the medical codes likely to occur in the next visit, and the time duration until the next visit. Dr.AI was a meaningful first step towards introducing deep learning to EHR in that in showed impressive performance for capturing the temporal nature of EHR, predicting future diagnosis codes with 0.64 recall@10, showing superior performance to traditional statistical methods. However, due to the blackbox nature of RNN,

it was not easy to interpret the temporal representations learned by, or predictions made by Dr.AI. Dr.AI is described in detail in Chapter 2.

During our research on Dr.AI, we discovered that using a pre-trained medical code representations improved the predictive performance of the model. Specifically, we pre-trained the representation vectors for each medical code using a co-occurrence based algorithm, namely Skip-gram [11], and used the resulting vectors in the embedding layer between the input layer and the RNN. Motivated by this success, for the next project, we aimed to learn interpretable representations of medical codes by taking into account the hierarchical structure of EHR. Our model, *Med2Vec* [12], improved the predictive performance for various tasks while providing interpretation for the learned medical code representation. Specifically, each dimension of the vector representations learned by Med2Vec represented a coherent clinical concept such as *injury from playing sports* or *symptoms and medications related to sickle cell disease*. Med2Vec was an important work in that it showed deep learning methods can be both accurate and interpretable. Med2Vec is described in detail in Chapter 3.

Although Med2Vec demonstrated strength in both prediction accuracy and interpretation of the learned representation, its interpretability was only applicable to non-sequence prediction tasks. In order to overcome this limit and provide interpretation for sequence prediction tasks, we decided to use the attention mechanism from neural machine translation [3]. By encoding the visit information via linear transformation, and generating the attention weights using the RNN, our model, *RETAIN* [13], demonstrated the same level of temporal prediction accuracy as the regular RNN while providing exact interpretation as to how much each medical code in each visit contributed to the prediction outcome. RETAIN is described in more detail in Chapter 4.

Up to RETAIN, our projects have been developed in a pure data-driven fashion. Healthcare field, however, is rich with domain knowledge curated by medical experts such as the

International Classification of Diseases (ICD) diagnosis hierarchy[1] or SNOMED-CT[2]. Such domain knowledge can be very helpful when we do not have sufficient training data to train high-capacity models such as neural networks. Our next project, *GRAM* [14], aimed to incorporate such domain knowledge into predictive neural network models. Specifically we focused on domain knowledge which could be represented as a directed acyclic graph (DAG). By developing an attention mechanism that operates on DAGs, we were able to learn medical code representations that closely align with the given domain knowledge. Thanks to the highly informative code representations, RNN's predictive performance improved when modeling rare diseases or when we used smaller amount of training data. GRAM is described in more detail in Chapter 5.

EHR data consist of heterogeneous data sources such as structured codes (*e.g.* diagnosis, medication, procedure codes), lab measures, clinical notes, and demographics. From Dr.AI to GRAM, we exclusively dealt with structured codes. But detailed or complementary patient information can be collected by incorporating more data sources, which is a natural future work of this thesis. However, while closely studying longitudinal EHR to design an optimal way to incorporate more data sources, we discovered that a patient encounter is more than just a set of medical codes, which is how we modeled each visit up to GRAM. In fact, a patient encounter is associated with one or more diagnosis codes and medication/procedure orders. Each medication/procedure order in turn is associated with a single diagnosis code, meaning that doctors order medications or procedures because of a specific diagnosis. This relationship between the diagnosis code and medication/procedure codes is information we neglected so far. We therefore decided to design a framework where we can effectively capture this relationship before trying to incorporate more data sources, some of which (*e.g.* lab measures) are associated each procedure order. The new framework, *MiME*, adds to the previous patient visit representation, one more level of embedding to capture the interaction between a diagnosis and its associated medication/procedure orders. MiME

[1] https://www.cdc.gov/nchs/icd/icd9.htm
[2] https://www.snomed.org/snomed-ct

outperformed various baseline models on heart failure prediction and sequential diagnoses prediction, and showed similar predictive performance to baseline models on medication prediction. Studying the experiment results, we could conclude that MiME indeed showed better performance than baseline models because of its ability to capture the interaction between diagnosis and medications/procedures, making it a strong foundation on which we can incorporate more data sources and add interpretability in the future.

In an effort to develop interpretable, accurate deep learning models for EHR, we began our research by modeling the temporal structure of EHR by directly applying RNN. Then we addressed three aspects of computational healthcare, namely representation learning, temporal interpretability, and domain knowledge incorporation, by developing Med2Ve, RETAIN, and GRAM. Additionally, we described the new framework, MiME that can act as the foundation for incorporating more data sources in the future. There are a number of interesting future works, but extending MiME to handle heterogeneous data types (*e.g.* lab measures, clinical notes) and provide interpretable predictions is the most natural next step. Another direction is training the RETAIN framework with policy gradient methods to delegate certain decision making problems to the machine while retaining the interpretability. For example, when we want to predict the onset of a certain disease as soon as possible, it is difficult for humans to determine the optimal time to make the diagnosis. Using the reinforcement learning technique, this task could be delegated to the machine, without having to have data labeled with the ground truth optimal time. Extending GRAM to incorporate additional domain knowledge besides the disease hierarchy is another straightforward future direction. Specifically, we are interested in leveraging the drug relation network and the procedure code hierarchy. Future works will be discussed in more detail in Chapter 7.

1.1 Interpretability of machine learning models

As the focus of this thesis is deep learning models' lack of interpretability, we use this section for discussing the notion *model interpretability* and introducing a few works that

address this topic.

The precise meaning of model interpretability is somewhat debatable, as different users may expect different types of interpretation. For example, studying the feature coefficients of a logistic regression model is a popular way of interpreting what the model has learned. Projecting word embeddings to a two dimensional space and studying their clustering behavior can be another form of interpreting what the model has learned. Some might even be interested to know how a model was trained from a given dataset (*i.e.* the optimization process). But when we state that deep learning models are not as interpretable as linear models, we generally mean that it is difficult to succinctly describe the relationship between the input and the output. Before further unfolding this section, we should mention that although researchers generally agree linear models are more interpretable than deep neural networks, it is not an undisputed notion. As pointed out by [15], linear models lose their interpretability as the number of features increases, and as features themselves become unintelligible (*e.g.* preprocessed with dimensionality reduction methods). Even decision trees, a popular choice for its interpretability and non-linearity, also cannot be intuitively understood as the size of the tree grows [16]. However, to avoid having this section become too philosophical, we focus on the unintelligible relationship between the input and output of deep neural networks, and popular ways to address it.

Since deep neural networks themselves are designed to be complex and non-linear, most approaches for interpreting them are post-hoc. Visualizing the learned representation is a popular method to understand the relationship between the input and the output. [17] introduced two ways to visualize the relationship between the input and the output of deep convolutional networks, using gradient descent. First approach is to derive an input image that maximizes a specific output class by fixing the model parameters and applying gradient descent with respect to the input image initialized with random values. The derived image is a local optimum since it relies on the initial value, and therefore it might not be the most representative example of what the model recognizes as a specific class. However,

empirically this approach provides decent insight into what kind of shapes and textures the model deems important to classify the given image. The second approach is to compute a class saliency map for a given image by using the Taylor expansion of the original convolutional model. The pixel magnitude in the saliency map indicates which pixels need to be changed the least to affect the class score the most.

Explanation by example is another way to interpret the deep neural networks. For example, as briefly mentioned above, plotting word embedding vectors to a two dimensional space using t-SNE [18] or PCA can tell us which words are close to one another in the latent space [11]. This type of interpretation does not directly shed light on the relationship between the input and the output. However, when sample A is classified as a certain class, then we can infer that other samples that are close to sample A would be classified as the same class, thus indirectly describing what the model has learned from the data.

An alternative way to interpret deep neural networks is to use model-agnostic approaches [19]. Partial dependence plot [20] can be used to identify what kind of relationship (*e.g.* linear, monotonic, more complex) a feature (or a set of features) has with the model outcome. Partial dependence works by marginalizing the model over the distribution of the features except the feature of interest, so that we are left with a function that captures the relationship between the model output and the feature of interest. Shapely values [21] can be used to identify how much each feature affected the model outcome to deviate from the model's average outcome. This method provides a concrete insight for a single sample, but it requires exponential time as we use more features. More recently, [22] proposed a method called LIME, which provides similar insight for a single sample as Shapely values, but requires much less time. LIME focuses on the neighborhood of a single sample; after generating many slightly perturbed samples, it trains an interpretable model (*e.g.* linear regression, decision tree) on those samples to approximately learn the relationship between the input features and the output of a single sample.

In this thesis, we use the definition of interpretability described above; succinctly describe

the relationship between the input and the output. But we add one more condition that is relevant to healthcare; whether the knowledge (*i.e.* the parameters) learned by the model aligns well with established medical knowledge. In the following chapters, in addition to analyzing the computational performance of our works, we will also study if and how they address these two aspects of interpretability.

CHAPTER 2
DOCTOR AI: PREDICTING CLINICAL EVENTS VIA RECURRENT NEURAL NETWORKS

Leveraging large historical data in electronic health record (EHR), we developed Doctor AI, a generic predictive model that covers observed medical conditions and medication uses. Doctor AI is a temporal model using recurrent neural networks (RNN) and was developed and applied to longitudinal time stamped EHR data from 260K patients over 8 years. Encounter records (e.g. diagnosis codes, medication codes or procedure codes) were input to RNN to predict (all) the diagnosis and medication categories for a subsequent visit. Doctor AI assesses the history of patients to make multilabel predictions (one label for each diagnosis or medication category). Based on separate blind test set evaluation, Doctor AI can perform differential diagnosis with up to 79% recall@30, significantly higher than several baselines. Moreover, we demonstrate great generalizability of Doctor AI by adapting the resulting models from one institution to another without losing substantial accuracy.

2.1 Introduction

A common challenge in healthcare today is that physicians have access to massive amounts of data on patients, but little time nor tools. Intelligent clinical decision support anticipates the information at the point of care that is specific to the patient and provider needs. Electronic health records (EHR), now commonplace in U.S. healthcare, represent the longitudinal experience of both patients and doctors. These data are being used with increasing frequency to predict future events. While predictive models have been developed to anticipate needs, most existing work has focused on specialized predictive models that predict a limited set of outcomes. However, day-to-day clinical practice involves an unscheduled and heterogeneous mix of scenarios and needs different prediction models in the hundreds to thousands. It is

impractical to develop and deploy specialized models one by one.

Leveraging large historical data in EHR, we developed Doctor AI, a generic predictive model that covers observed medical conditions and medication uses. Doctor AI is a temporal model using recurrent neural networks (RNN) and was developed and applied to longitudinal time stamped EHR data. In this chapter, we are particularly interested in whether historical EHR data may be used to predict future physician diagnoses and medication orders. Applications that accurately forecast could have many uses such as anticipating the patient status at the time of visit and presenting data a physician would want to see at the moment. The primary goal of this study was to use longitudinal patient visit records to predict the physician diagnosis and medication order of the next visit. As a secondary goal we predicted the time to the patient's next visit. Predicting the visit time facilitates guidance of whether a patient may be delayed in seeking care.

The two tasks addressed in this chapter are different from sequence labeling tasks often seen in natural language processing applications, e.g., part-of-speech tagging. Our proposed model, Doctor AI, performs multilabel prediction (one for each disease or medication category) over time while sequence labeling task predicts a single label at each step. The key challenge was finding a flexible model that is capable of performing the multilabel prediction problem. The two main classes of techniques have been proposed in dealing with temporal sequences: 1) continuous-time Markov chain based models [23, 24, 25], and 2) intensity based point process modeling techniques such as Hawkes processes [26, 27, 28]. However, both classes are expensive to compute, especially for nonlinear settings. Furthermore, they often make strong assumptions about the data generation process which might not be valid for EHR data. Our modeling strategy was to develop a generalized approach to representing patient temporal healthcare experience to predict all the diagnoses, medication categories and visit time. We used recurrent neural network (RNN), considering that RNNs have been particularly successful for representation learning in sequential data, *e.g.* [29, 30, 31, 32, 33]. In particular, we make the following main contributions in this chapter:

- We demonstrate how RNNs can be used to represent the patient status and predict diagnosis, medication order and visit time. The trained RNN is able to achieve above 64% recall@10 and 79% recall@30 for diagnosis prediction, showing potential to serve as a differential diagnosis assistance.
- We propose an initialization scheme for RNNs using Skip-gram embeddings [11] and show that it improves the performance of the RNN in both accuracy and speed.
- We empirically confirm that RNN models possess great potential for transfer learning across different medical institutions. This suggests that health systems with insufficient patient data can adopt models learned from larger datasets of other health systems to improve prediction accuracy on their smaller population.

2.2 Related Work

In this section, we briefly review the common approaches to modeling multilabel event sequences with special focus on the models that have been applied to medical data. There are two main approaches to modeling multilabel event sequences: with or without discretization (binning) of time.

Discretization. When the time axis is discretized, the point process data can be converted to binary time series (or time series of count data if binning is coarse) and analyzed via time series analysis techniques [34, 35, 36]. However, this approach is inefficient as it produces long time series whose elements are mostly zero. Furthermore, discretization of time introduces noise in the time stamps of visits. Finally, these approaches are often not able to model the duration until next event. Thus, it is advantageous not to discretize the data both in terms of modeling and computation.

Continuous-time models. Among the continuous-time models, there are two main techniques: continuous-time Markov chain based models [37, 25, 38, 39] and their extension using Baysian networks [23, 40] and intensity function modeling techniques such as Cox

and Hawkes processes [26, 41, 42, 28].

Intensity function modeling techniques have been shown to have computational advantages over the continuous-time Markov chain based models. Moreover, modeling multilabel marked point processes with continuous-time Markov chains expands their state-space and make them even more expensive. However, Hawkes processes only depend linearly on the past observation times; while there are limited classes of non-linear Hawkes process [27], the temporal dynamics can be more complex. Finally, Hawkes processes are known to have a flat loss function near optimal value of the parameters which renders the gradient-based learning algorithms inefficient [43]. In this paper we address these challenges by designing a recurrent neural network which has been shown to be successful in learning complex sequential patterns.

Disease progression models. There have been active research in modeling the temporal progression of diseases [44]. Generally, most works can be divided into two groups: works that focus on a specific disease and works that focus on a broader range of diseases.

Specific-purpose progression modeling: There have been many studies that focus on modeling the temporal progression of a specific disease based on either intensive use of domain-specific knowledge [45, 46, 47] or taking advantage of advanced statistical methods [39, 48, 49, 50]. Specifically, studies have been conducted on Alzheimer's disease [46, 50, 49], glaucoma [39], chronic kidney disease [47], diabetes mellitus [45], and abdominal aortic aneurysm [48]

General-purpose progression modeling: Recently, [51, 28, 36] proposed more general approaches to modeling the progression of wider range of diseases. As discussed earlier, [28] used Hawkes process, and [36] discretized time in order to model multiple patients and multiple diseases. [51] proposed a graphical model based on Markov Jump Process to predict the stage progression of chronic obstructive pulmonary disease (COPD) and its co-morbid diseases.

One of the main challenges in using these algorithms is scalability. The datasets used in

Table 2.1: Basic statistics of the the clinical records dataset.

# of patients	263,706	Total # of codes	38,594
Avg. # of visits	54.61	Total # of 3-digit Dx codes	1,183
Avg. # of codes per visit	3.22	# of top level Rx codes	595
Max # of codes per visit	62	Avg. duration between visits	76.12 days

previous works typically contain up to a few thousands of patients and a few hundreds of codes. Even the largest dataset used by [36] contains 13,180 patients and 8,722 codes, which is significantly smaller than our dataset described in Table 2.1. Need for domain-specific knowledge is also a big challenge. For example, [51] not only used a smaller dataset (3,705 patients and 264 codes) but also used co-morbidity information to improve the performance of their algorithm. Such expert knowledge is difficult to obtain from typical EHR data.

Deep learning models for EHR. Researchers have recently begun attempting to apply neural network based methods (or deep learning) to EHR to utilize its ability to learn complex patterns from data. Previous studies such as phenotype learning [52, 53, 54] or representation learning [55, 12, 56], however, have not fully addressed the sequential nature of EHR. [57] is especially related to our work in that both studies use RNN for sequence prediction. However, while [57] uses regular times series of real-valued variables collected from ICU patients to predict diagnosis codes, we use discrete medical codes (*e.g.* diagnosis, medication, procedure) extracted from longitudinal patient visit records. Also, in each visit we make a prediction about predict diagnosis, medication order in the next visit and and the time to next visit.

2.3 Cohort

Population and source of data. The source population for this study was primary care patients from Sutter Health Palo Alto Medical Foundation. Sutter Health is a large primary care and multispecialty group practice that has used an Epic Systems Corporation EHR for more than a decade. The dataset was extracted from a density sampled case-control study for heart failure. The dataset consists of de-identified encounter orders, medication orders,

problem list records and procedure orders.

Data processing. As inputs, we use ICD-9 codes, medication codes, and procedure codes. We extracted ICD-9 codes from encounter records, medication orders, problem list records and procedure orders. Generic Product Identifier (GPI) medication codes and CPT procedure codes were extracted from medication orders and procedure orders respectively. All codes were timestamped with the patients' visit time. If a patient received multiple codes in a single visit, those codes were given the same timestamp. We excluded patients that made less than two visits. The resulting dataset consists of 263,706 patients who made on average 54.61 visits per person.

Grouping medical codes. There are more about 11,000 unique ICD-9 codes and 18,000 GPI medication codes in the dataset, many of which are very granular. For example, pulmonary tuberculosis (ICD-9 code 011) is divided into 70 subcategories (ICD-9 code 011.01, 011.02, ..., 011.95, 011.96). Simply knowing that a patient is likely to have pulmonary tuberculosis is enough to increase the doctor's awareness of the severity of the clinical situation. Therefore, to predict diagnosis and medication order, we grouped codes into higher-order categories to reduce the feature set and information overload. For the diagnosis codes, we use the 3-digit ICD-9 codes, yielding 1183 unique codes. For the medication codes, we use the Generic Product Identifier Drug Class, which groups the medication codes into 595 unique groups. The label $\mathbf{y}_i$ we use in the following sections represents the 1,778-dimensional vector (i.e., 1183 + 595) for the grouped diagnosis codes and medication codes.

2.4 Methods

This section describes the RNN model for multilabel point processes. We will also describe how we predict diagnosis, medication order and visit time using the RNN model.

Problem setting. For each patient, the observations are drawn from a multilabel point process in the form of $(t_i, \boldsymbol{x}_i)$ for $i = 1, \ldots, n$. Each pair represents an event, such as an ambulatory care visit, during which multiple medical codes such as ICD-9 diagnosis codes,

procedure codes, or medication codes are documented in the patient record. The multi-hot label vector $\boldsymbol{x}_i \in \{0,1\}^p$ represents the medical codes assigned at time t_i, where p denotes the number of unique medical codes. At each timestamp, we may extract higher-level codes for prediction purposes and denote it by $\boldsymbol{y}_i$, see the details in section 2.3. The number of events for each patient may differ.

Gated Recurrent Units Preliminaries. Specifically, we implemented our RNN with Gated Recurrent Units (GRU). Although Long Short Term Memory (LSTM) [58, 59] has drawn much attention from many researchers, GRU has recently shown to have similar performance as LSTM, while employing a simpler architecture [60]. In order to precisely describe the network used in this work, we reiterate the mathematical formulation of GRU as follows:

$$\begin{aligned}
\mathbf{z}_i &= \sigma(\mathbf{W}_z \mathbf{x}_i + \mathbf{U}_z \mathbf{h}_{i-1} + \mathbf{b}_z) \\
\mathbf{r}_i &= \sigma(\mathbf{W}_r \mathbf{x}_i + \mathbf{U}_r \mathbf{h}_{i-1} + \mathbf{b}_r) \\
\tilde{\mathbf{h}}_i &= \tanh(\mathbf{W}_h \mathbf{x}_i + \mathbf{r}_i \circ \mathbf{U}_h \mathbf{h}_{i-1} + \mathbf{b}_h) \\
\mathbf{h}_i &= \mathbf{z}_i \circ \mathbf{h}_{i-1} + (1 - \mathbf{z}_i) \circ \tilde{\mathbf{h}}_i
\end{aligned}$$

where $\mathbf{z}_i$ and $\mathbf{r}_i$ respectively represent the update gate and the reset gate, $\tilde{\mathbf{h}}_i$ the intermediate memory unit, $\mathbf{h}_i$ the hidden layer, all at timestep t_i. A detailed description of GRU is provided in Supplementary 2.7.

Description of neural network architecture. Our goal is to learn an effective vector representation for the patient status at each timestamp t_i. Using effective patient representations, we are interested in predicting diagnosis and medication categories in the next visit $\boldsymbol{y}_{i+1}$ and the time duration until the next visit $d_{i+1} = t_{i+1} - t_i$. Finally, we would like to perform all these steps jointly in a single supervised learning scheme. We use RNN to learn such patient representations, treating the hidden layer as the representation for the patient status and use it for the prediction tasks.

The proposed neural network architecture (Figure 2.1) receives input at each timestamp

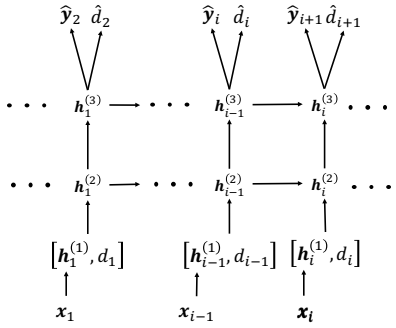


Figure 2.1: This diagram shows how we have applied RNNs to solve the problem of forecasting of next visits' time and the codes assigned during each visit. The first layer simply embeds the high-dimensional input vectors in a lower dimensional space. The next layers are the recurrent units (here two layers), which learn the status of the patient at each timestamp as a real-valued vector. Given the status vector, we use two dense layers to generate the codes observed in the next timestamp and the duration until next visit.

t_i as the concatenation of the multi-hot input vector $\boldsymbol{x}_i$ of the multilabel categories and the duration d_i since the last event. In our datasets, the input dimension is as large as $40,000$. Thus, the next layer projects the input to a lower dimensional space. Then, we pass the lower dimensional vector through RNN (implemented with GRU in our study). We can also stack multiple layers of RNN to increase the representative power of the network. Finally, we use a Softmax layer to predict the diagnosis codes and the medication codes, and a rectified linear unit (ReLU) to predict the time duration until next visit.

For predicting the diagnosis codes and the medication codes at each timestep t_i, a Softmax layer is stacked on top of the GRU, using the hidden layer $\mathbf{h}_i$ as the input: $\widehat{\mathbf{y}}_{i+1} = \mathrm{softmax}({\mathbf{W}_{code}}^{\top}\mathbf{h}_i + \mathbf{b}_{code})$. For predicting the time duration until the next visit, a rectified linear unit (ReLU) is placed on top of the GRU, again using the hidden layer $\mathbf{h}_i$ as the input: $\widehat{d}_{i+1} = \max({\mathbf{w}_{time}}^{\top}\mathbf{h}_i + b_{time}, 0)$. The objective of training our model is to learn the weights $\mathbf{W}_{\{z,r,h,code\}}$, $\mathbf{U}_{\{z,r,h\}}$, $\mathbf{b}_{\{z,r,h,code\}}$, $\mathbf{w}_{time}$ and b_{time}. The values of all $\mathbf{W}$'s and $\mathbf{U}$'s were initialized to orthonormal matrices using singular value decomposition of matrices generated from the normal distribution [61]. The initial value of $\mathbf{w}_{time}$ was chosen from the uniform distribution between -0.1 and 0.1. All $\mathbf{b}$'s and b_{time} were initialized to zeros. The joint loss function consists of the cross entropy for the code prediction and the squared loss

for the time duration prediction, as described below for a single patient:

$$\mathcal{L}(\mathbf{W}, \mathbf{U}, \mathbf{b}, \mathbf{w}_{time}, b_{time}) = \sum_{i=1}^{n-1} \left\{ \left(\mathbf{y}_{i+1} \log(\widehat{\mathbf{y}}_{i+1}) + (1 - \mathbf{y}_{i+1}) \log(1 - \widehat{\mathbf{y}}_{i+1}) \right) + \frac{1}{2} \| d_{i+1} - \widehat{d}_{i+1} \|_2^2 \right\}$$

As mentioned above, the multi-hot vectors $\mathbf{x}_i$ of almost 40,000 dimensions are first projected to a lower dimensional space, then put into the GRU. We employed two different approaches for this: (1) We put an extra layer of a certain size between the multi-hot input $\mathbf{x}_i$ and the GRU, and call it the embedding layer. We denote the weight matrix between the multi-hot input vector and the embedding layer as $\mathbf{W}_{emb}$. Then we learn the weight $\mathbf{W}_{emb}$ as we train the entire model. (2) We initialize the weight $\mathbf{W}_{emb}$ with a matrix generated by Skip-gram algorithm [11], then refine the weight $\mathbf{W}_{emb}$ as we train the entire model. This can be seen as using the pre-trained Skip-gram vectors as the input to the RNN and fine-tuning them with the joint prediction task. The brief description of learning the Skip-gram vectors from the EHR is provided in Supplementary 2.8. The first and second approach can be formulated as follows:

$$\boldsymbol{h}_i^{(1)} = [\tanh(\mathbf{x}_i{}^\top \mathbf{W}_{emb} + \mathbf{b}_{emb}),\ d_i] \tag{2.1}$$

$$\boldsymbol{h}_i^{(1)} = [\mathbf{x}_i{}^\top \mathbf{W}_{emb},\ d_i] \tag{2.2}$$

where $[\cdot, \cdot]$ is the concatenation operation used for appending the time duration to the multi-hot vector $\boldsymbol{h}_i^{(1)}$ to make it an input vector to the GRU.

2.5 Results

We now describe the details of our experiments in the proposed RNN approach to forecasting the future clinical events. The source code of Doctor AI is publicly available at `https://github.com/mp2893/doctorai`.

2.5.1 Experiment Setup

For training all models including the baselines, we used 85% of the patients as the training set and 15% as the test set. We trained the RNN models for 20 epochs (*i.e.*, 20 iterations over the entire training data) and then evaluated the final performance against the test set. To avoid overfitting, we used dropout between the GRU layer and the prediction layer (*i.e.* code prediction and time duration prediction). Dropout was also used between GRU layers if we were using a multi-layer GRU. We also applied norm-2 regularization on both $\mathbf{W}_{code}$ and $\mathbf{w}_{time}$. Both regularization coefficients were set to 0.001. The size of the hidden layer $\mathbf{h}_i$ of the GRU was set to 2000 to guarantee a sufficient expressive power. After running sets of preliminary experiments where we tried the size from 100 to 2000, we noticed that the code prediction performance started to saturate around 1600$\sim$1800. All models were implemented with Theano [62] and trained on a machine equipped with two Nvidia Tesla K80 GPUs.

We train total four different variation of Doctor AI as follows,

- **RNN-1**: RNN with a single hidden layer initialized with a random orthogonal matrix for $\mathbf{W}_{emb}$.
- **RNN-2**: RNN with two hidden layers initialized with a random orthogonal matrix for $\mathbf{W}_{emb}$.
- **RNN-1-IR**: RNN using a single hidden layer initialized embedding matrix $\mathbf{W}_{emb}$ with the Skip-gram vectors trained on the entire dataset.
- **RNN-2-IR**: RNN with two hidden layers initialized embedding matrix $\mathbf{W}_{emb}$ with the Skip-gram vectors trained on the entire dataset. dataset.

2.5.2 Evaluation metrics

The performance of algorithms in predicting diagnoses and medication codes was evaluated using the Top-k recall defined as:

$$\text{top-}k \text{ recall} = \frac{\text{\# of true positives in the top } k \text{ predictions}}{\text{\# of true positives}}$$

Top-k recall mimics the behavior of doctors conducting differential diagnosis, where doctors list most probable diagnoses and treat patients accordingly to identify the patient status. Therefore, a machine with a high Top-k recall translates to a doctor with an effective diagnostic skill. This makes Top-k recall an attractive performance metric for our problem.

We select the maximum k to be 30 to evaluate the performance of the models not only for simple cases but also for complex cases. Near 50.7% of the patients have been assigned with more than 10 diagnosis and medication codes at least once. Since it is those complex cases that are of interest to predict and analyze, we choose k to be large enough (i.e., 3 times of the mean).

Coefficient of determination (R^2) was used to evaluate the predictive performance of regression and forecasting algorithms. It compares the accuracy of the prediction with respect to the simple prediction by mean of the target variable.

$$R^2 = 1 - \frac{\sum_i (y_i - \widehat{y_i})^2}{\sum_i (y_i - \overline{y_i})^2}$$

Because time to the next visit can be highly skewed, we measure the R^2 performance of the algorithms in predicting $\log(d_i)$ to lower the impact of anomalous long durations in the performance metric. In the same spirit, we train all models to predict the logarithm of the time duration between visits.

2.5.3 Baselines

We compare our model against several baselines as described below. Some of the existing techniques based on continuous-time Markov chain and latent space models were not scalable enough to be trained using the entire dataset in a reasonable amount of time; thus comparison is not feasible.

Frequency baselines. We compare our algorithms against simple baselines that are based on experts' intuition about the dynamics of events in clinical settings. The first baseline uses a patient's medical codes in the last visit as the prediction for the current visit. This baseline is competitive when the status of a patient with a chronic condition stabilizes over time. We enhanced this baseline using the top-k most frequent labels observed in visits prior to the current visits. In the experiments we observe that the baseline of top-k most frequent labels is quite competitive.

Logistic and Neural Network time series models. A common way to perform prediction task is to use $\mathbf{x}_{i-1}$ to predict the codes in the next visit $\mathbf{x}_i$ using logistic regression or multilayer perceptron (MLP). To enhance the baseline further, we can use the data from L time lags before and aggregate them $\mathbf{x}_{i-1} + \mathbf{x}_{i-2} + \ldots, +\mathbf{x}_{i-L}$ for some duration L to create the features for prediction of $\mathbf{x}_i$. Similarly, we can have a model that predicts the time until next visit using rectified linear units (ReLU) as the output activation. We set the lag $L = 5$ so that the logistic regression and MLP can use information from maximum five past visits. The details of MLP design are described in Supplementary 2.9.

2.5.4 Prediction performance

Table 2.2 compares the results of different algorithms with RNN based Doctor AI. We report the results in three settings: when we are interested in (1) predicting only diagnosis codes (Dx), (2) predicting only medication codes (Rx), and (3) jointly predicting Dx codes, Rx codes, and the time duration to next visit. The results confirm that the proposed approach is able to outperform the baseline algorithms by a large margin. Note that the recall values for

Table 2.2: Accuracy of algorithms in forecasting future medical activities. Embedding matrices $\mathbf{W}_{emb}$ of both RNN-1 (using one hidden layer) and RNN-2 (using two hidden layers) are initialized with random orthogonal vectors. Embedding matrices $\mathbf{W}_{emb}$ of both RNN-1-IR (using one hidden layer) and RNN-2-IR (using two hidden layers) are initialized with Skip-gram vectors trained on the entire dataset.

	Dx Only Recall @k			Rx Only Recall @k			Dx,Rx,Time Recall @k			
Algorithms	$k=10$	$k=20$	$k=30$	$k=10$	$k=20$	$k=30$	$k=10$	$k=20$	$k=30$	R^2
Last visit	29.17			13.81			26.25			—
Most freq.	56.63	67.39	71.68	62.99	69.02	70.07	48.11	60.23	66.00	—
Logistic	43.24	54.04	60.76	45.80	60.02	68.93	36.04	46.32	52.53	0.0726
MLP	46.66	57.38	64.03	47.62	61.72	70.92	38.82	49.09	55.74	0.1221
RNN-1	63.12	73.11	78.49	67.99	79.55	**85.53**	53.86	65.10	71.24	0.2519
RNN-2	63.32	73.32	78.71	67.87	79.47	85.43	53.61	64.93	71.14	0.2528
RNN-1-IR	63.24	73.33	78.73	**68.31**	**79.77**	85.52	54.37	65.68	71.85	0.2492
RNN-2-IR	**64.30**	**74.31**	**79.58**	68.16	79.74	85.48	**54.96**	**66.31**	**72.48**	**0.2534**

the joint task are lower than those for Dx code prediction or Rx code prediction because the hypothesis space is larger for the joint prediction task. The superior performance of RNN based approaches can be attributed to the efficient representation that they learn for patients at each visit [63, 64]. RNNs are able to learn succinct feature representations of patients by accumulating the relevant information from their history and the current set of codes, which outperformed hand-picked features of frequency baselines.

Table 2.2 confirms that learning patient representation with RNN is easier with the input vectors that are already efficient representations of the medical codes. The RNN trained with the Skip-gram vectors (denoted by RNN-IR) consistently outperforms the RNN that learns the weight matrix $\mathbf{W}_{emb}$ directly from the data, with only one exception, the medication prediction Recall@30, although the differences are insignificant. The results also confirm that having multiple layers when using RNN improves its ability to learn more efficient representations. The results also indicate that a single layer RNN might have enough representative power to capture the dynamics of medications, and adding more layers may not improve the performance.

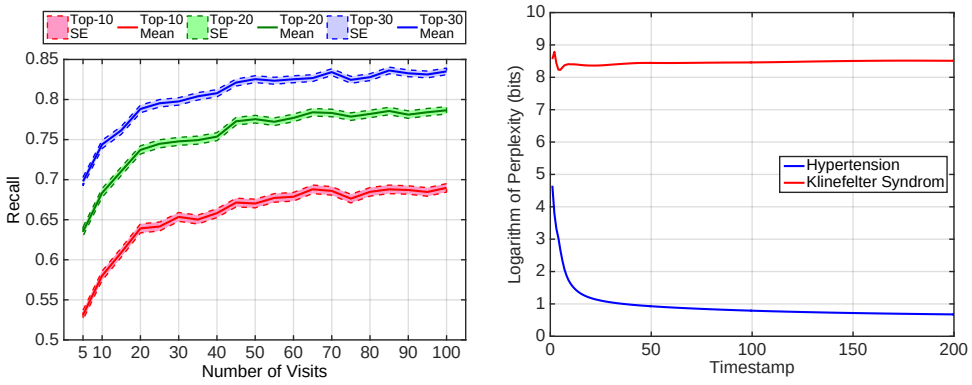
The results also indicate that our approach significantly improves the accuracy of predicting the time duration until the next visit compared to the baselines. However, the

absolute value of R^2 metric shows that accurate prediction of time intervals remains as a challenge. We believe achieving significantly better time prediction without extra features should be difficult because the timing of a clinical visit can be affected by many personal factors such as financial status, location of residence, means of transportation, and life style, to name a few. Thus, without such sensitive personal information, which is rarely included in the EHR, accurate prediction of time intervals should be unlikely.

2.5.5 Understanding the behavior of the network

To study the applicability of our model in a real-world setting where patients have varying length of medical records, we conducted an additional experiment to study the relationship between the length of the patient medical history and the prediction performance. To this end, we selected 5,800 patients from the test set who had more than 100 visits. We used the best performing model to predict the diagnosis codes at visits at different times and found the mean and standard error of recall across the selected patients. Figure 2.2a shows the result of the experiment. We believe that the increase in performance can be due to two reasons: (1) RNN is able to learn a better estimate of the patient status as it sees longer patient records and (2) Visits are correlated with poor health. Those with high visit count are more likely to be severely ill, and therefore their future is easier to predict.

Another experiment was conducted to understand the behavior of the network by giving synthetic inputs. We chose hypertension (ICD-9 code 401.9) as an example of a frequently observed diagnosis, and Klinefelter's syndrome (ICD-9 code 758.7) as an example of an infrequent diagnosis. We created two synthetic patients who respectively have 200 visits of 401.9 and 758.7. Then we used the best performing model to predict the diagnosis codes for the next visits. Figure 2.2b shows contrasting patterns: when the input is one of the frequent codes such as hypertension, the network quickly learns a more specific set of output codes as next disease. When we select an infrequent code like Klinefelter's syndrome as the input, the network's output is more diverse and mostly the frequently observed codes. The top 30



(a) (b)

Figure 2.2: Characterizing behavior of the trained network: (a) Prediction performance of Doctor AI as it sees a longer history of the patients. (b) Change in the perplexity of response to a frequent code (hypertension) and an infrequent code (Klinefelter's syndrome).

codes after convergence shown in Table 2.4 in Supplementary 2.10 confirm the disparity of the diversity of the predicted codes for the two cases.

2.5.6 Knowledge transfer across hospitals

As we observed from the previous experiments, the dynamics of clinical events are complex, which requires models with a high representative power. However, many institutions have not yet collected large scale datasets, and training such models could easily lead to overfitting. To address this challenge, we resort to the recent advances in domain adaptation techniques for deep neural networks [65, 66, 67, 68].

A different dataset, MIMIC II, which is a publicly available clinical dataset collected from ICU patients over 7 years of observation, was chosen to conduct the experiment. This dataset differs from the Sutter dataset in that it consists of demographically and diagnostically different patients. The number of patients who made at least two visits is 2,695, and the number of unique diagnosis code (3-digit ICD-9 code) is 767, which is a subset of the Sutter dataset. From the dataset, we extracted sequences of 3-digit ICD-9 codes. We chose 2,290 patients for training, 405 for testing. We chose the 2-layer RNN with 1000 dimensional hidden layer, and performed two experiments: 1) We trained the model only on the MIMIC II dataset. 2) We initialized the coefficients of the model with the values learned from the

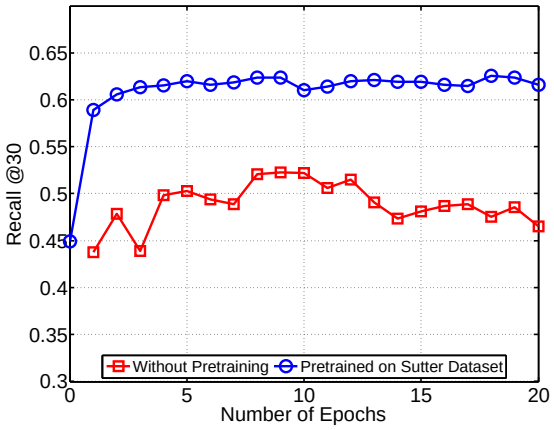


Figure 2.3: The impact of pre-training on improving the performance on smaller datasets. In the first experiment, we first train the model on a small dataset (red curve). In the second experiment, we pre-train the model on our large dataset and use it for initializing the training of the smaller dataset. This procedure results in more than 10% improvement in the performance.

3-digit ICD-9 sequences of the Sutter data, then we refined the coefficients with the MIMIC II dataset. Figure 2.3 shows the vast improvement of the prediction performance induced by the knowledge transfer from the Sutter data.

2.6 Conclusion

In this chapter, we proposed Doctor AI system, which is a RNN-based model that can learn efficient patient representation from a large amount of longitidinal patient records and predict future events of patients. We tested Doctor AI on a large real-world EHR datasets, which achieved 79.58% recall@30 and significantly outperformed many baselines. We have also shown that the patient's visit count and the rarity of medical codes highly influence the performance. We have also demonstrated that knowledge learned from one hospital could be adapted to another hospital. The empirical analysis by a medical expert confirmed that Doctor AI not only mimics the predictive power of human doctors, but also provides diagnostic results that are clinically meaningful.

One limitation of Doctor AI is that, in medical practice, incorrect predictions can sometimes be more important than correct predictions as they can degrade patient health.

Also, although Doctor AI has shown that it can mimic physicians' average behavior, it would be more useful to learn to perform better than average. We set as our future work to address these issues so that Doctor AI can provide practical help to physicians in the future.

Supplementary

2.7 Description of Gated Recurrent Units

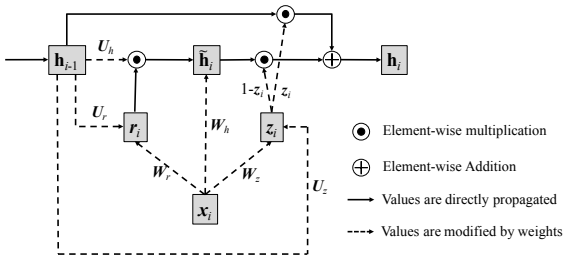


Figure 2.4: Architecture of GRU

We first reiterate the mathematical formulation of GRU so that the reader can see Figure 2.4 and the formulations together.

$$\mathbf{z}_i = \sigma(\mathbf{W}_z\mathbf{x}_i + \mathbf{U}_z\mathbf{h}_{i-1} + \mathbf{b}_z)$$

$$\mathbf{r}_i = \sigma(\mathbf{W}_r\mathbf{x}_i + \mathbf{U}_r\mathbf{h}_{i-1} + \mathbf{b}_r)$$

$$\tilde{\mathbf{h}}_i = \tanh(\mathbf{W}_h\mathbf{x}_i + \mathbf{r}_i \circ \mathbf{U}_h\mathbf{h}_{i-1} + \mathbf{b}_h)$$

$$\mathbf{h}_i = \mathbf{z}_i \circ \mathbf{h}_{i-1} + (1 - \mathbf{z}_i) \circ \tilde{\mathbf{h}}_i$$

Figure 2.4 depicts the architecture of the GRU, where $\mathbf{x}_i$, $\mathbf{z}_i$ and $\mathbf{r}_i$ respectively represent the input, update gate and the reset gate, $\tilde{\mathbf{h}}_i$ the intermediate memory unit, $\mathbf{h}_i$ the hidden layer, all at timestep t_i. $\mathbf{W}_h, \mathbf{W}_z, \mathbf{W}_r, \mathbf{U}_h, \mathbf{U}_z, \mathbf{U}_r$ are the weight matrices to be learned. Note that the bias vectors $\mathbf{b}_h, \mathbf{b}_z, \mathbf{b}_r$ are omitted in Figure 2.4.

The outstanding difference between the classical RNN (Elman Network) and GRU is that the previous hidden layer $\mathbf{h}_{i-1}$ and the current input $\mathbf{x}_i$ do not directly change the value of the current hidden layer $\mathbf{h}_i$. Instead, they change the values of both gates $\mathbf{z}_i$, $\mathbf{r}_i$ and the

intermediate memory unit $\tilde{\mathbf{h}}_i$. Then the current hidden layer $\mathbf{h}_i$ is updated by $\tilde{\mathbf{h}}_i$ and $\mathbf{z}_i$. Due to the σ function, both gates $\mathbf{z}_i$ and $\mathbf{r}_i$ have values between 0 and 1. Therefore if the reset gate $\mathbf{r}_i$ is close to zero, the intermediate memory unit $\tilde{\mathbf{h}}_i$ will disregard the past values of the hidden layer $\mathbf{h}_{i-1}$. If the update gate $\mathbf{z}_i$ is close to one, the current hidden layer $\mathbf{h}_i$ will disregard the current input $\mathbf{x}_i$, and retain the value from the previous timestep $\mathbf{h}_{i-1}$.

Simply put, the reset gate allows the hidden layer to drop any information that is not useful in making a prediction, and the updated gate controls how much information from the previous hidden layer should be propagated to the current hidden layer. This characteristic of GRU is especially useful as it is not easy to identify information essential to predicting the future diagnosis, medication or the time duration until the next visit.

2.8 Learning the Skip-gram vectors from the EHR

Learning efficient representations of medical codes (*e.g.* diagnosis codes, medication codes, and procedure codes) may lead to improved performance of many clinical applications. We specifically used Skip-gram [11] to learn real-valued multidimensional vectors to capture the latent representation of medical codes from the EHR.

We processed the private dataset so that diagnosis codes, medication codes, procedure codes are laid out in a temporal order. If there are multiple codes at a single visit, they were laid out in a random order. Then using the context window size of 5 to the left and 5 to the right, and applying Skip-gram, we were able to project diagnosis codes, medication codes and procedure codes into the same lower dimensional space, where similar or related codes are embedded close to one another. For example, hypertension, obesity, hyperlipidemia all share similar values compared to pneumonia or bronchitis. The trained Skip-gram vectors are then plugged into RNN so that a multi-hot vector can be converted to vector representations of medical codes.

2.9 Details of the training procedure of multilayer perceptron

We use a multilayer perceptron with a hidden layer of width 2,000. We apply L_2 regularization to all of the weight matrices. The activation functions in the first and output layers are selected to be tanh and softmax functions respectively. For prediction of time intervals, we used rectified linear units.

2.10 Case study

The detailed results are shown in Table 2.3. To take a closer look at the performance of Doctor AI, in Table 2.3 (in Supplementary 2.10) we list the predicted, true, and historical diagnosis codes for five visits of different patients. The blue items represent the correct predictions. The results are promising and show that, given the history of the patient, the Doctor AI can predict the true diagnostic codes. The results highly mimic the way a human doctor will interpret the disease predictions from the history. For all five of the cases shown in Table 2.3, the set of predicted diseases contain most, if not all of the true diseases. For example, in the first case, the top 3 predicted diseases match the true diseases. A human doctor would likely predict similar diseases to the ones predicted with Doctor AI, since old myocardial infarction and chronic ischemic heart disease can be associated with infections and diabetes [69].

In the fourth case, visual disturbances can be associated with migraines and essential hypertension [70]. Further, essential hypertension may be linked to cognitive function [71], which plays a role in anxiety disorders and dissociative and somatoform disorders. Regarding codes that are guessed incorrectly with the fourth case, they can still be plausible given the history. For example, cataracts, and disorders of refraction and accommodation could have been guessed based on a history of visual disturbances, as well as strabismus and disorders of binocular eye movements. Allergic rhinitis could have been guessed, because there was a history of allergic rhinitis. In summary, Doctor AI is able to very accurately

predict the true diagnoses in the sample patients. The results are promising and should motivate future studies involving the application of Doctor AI on different datasets exhibiting other populations of patients.

Table 2.3: Comparison of the diagnoses by Doctor AI and the true future diagnoses.

Predicted		True		History	
ICD9	Description	ICD9	Description	ICD9	Description
412	**Old myocardial infarction**				
V58	**Encounter for other and unspecified procedures**				
414	**Other forms of chronic ischemic heart disease**				
272	Disorders of lipoid metabolism			465	Acute upper respiratory infec. of multiple or unspec. sites
250	Diabetes mellitus	**414**	**Other forms of chronic ischemic heart disease**	250	Diabetes mellitus
585	Chronic kidney disease (CKD)	**412**	**Old myocardial infarction**	366	Cataract
428	Heart failure	**V58**	**Encounter for other and unspecified procedures**	V58	Encounter for other and unspecified procedures
285	Other and unspecified anemias			362	Other retinal disorders
V04	Need for prophylactic vaccin. and inocul. against certain diseases				
V76	Special screening for malignant neoplasms				
V07	**Need for isolation and other prophylactic measures**				
477	Allergic rhinitis				
780	General symptoms			782	Symptoms involving skin and other integumentary tissue
401	**Essential hypertension**	**V07**	**Need for isolation and other prophylactic measures**	477	Allergic rhinitis
786	**Symptoms involving respiratory system**	**401**	**Essential hypertension**	V07	Need for isolation and other prophylactic measures
493	Asthma	**786**	**Symptoms involving respiratory system**	564	Functional digestive disorders, not elsewhere classified
300	Anxiety, dissociative and somatoform disorders	782	Symptoms involving skin and other integumentary tissue	401	Essential hypertension
461	Acute sinusitis				
530	Diseases of esophagus				
719	Other and unspecified disorders of joint				
453	Other venous embolism and thrombosis				
V58	**Encounter for other and unspecified procedures**				
719	**Other and unspecified disorders of joint**				
V12	**Personal history of certain other diseases**	**715**	**Osteoarthrosis and allied disorders**	453	Other venous embolism and thrombosis
V43	Organ or tissue replaced by other means	**V12**	**Personal history of certain other diseases**	956	Injury to peripheral nerve(s) of pelvic girdle and lower limb
729	Other disorders of soft tissues	**719**	**Other and unspecified disorders of joint**	V43	Organ or tissue replaced by other means
715	**Osteoarthrosis and allied disorders**	**V58**	**Encounter for other and unspecified procedures**		
733	Other disorders of bone and cartilage				
726	Peripheral enthesopathies and allied syndromes				
451	Phlebitis and thrombophlebitis				
477	Allergic rhinitis				
780	**General symptoms**				
300	**Anxiety, dissociative and somatoform disorders**				
401	**Essential hypertension**	**401**	**Essential hypertension**	782	Symptoms involving skin and other integumentary tissue
346	**Migraine**	**780**	**General symptoms**	477	Allergic rhinitis
366	Cataract	**346**	**Migraine**	692	Contact dermatitis and other eczema
V43	Organ or tissue replaced by other means	**300**	**Anxiety, dissociative and somatoform disorders**	368	Visual disturbances
367	Disorders of refraction and accommodation			378	Strabismus and other disorders of binocular eye movements
368	Visual disturbances				
272	Disorders of lipoid metabolism				
428	**Heart failure**				
427	**Cardiac dysrhythmias**				
272	**Disorders of lipoid metabolism**				
401	Essential hypertension	**250**	**Diabetes mellitus**	466	Acute bronchitis and bronchiolitis
786	Symptoms involving respiratory system	402	Hypertensive heart disease	428	Heart failure
185	Malignant neoplasm of prostate	**428**	**Heart failure**	786	Symptoms involving respiratory system
250	**Diabetes mellitus**	**272**	**Disorders of lipoid metabolism**	785	Symptoms involving cardiovascular system
414	Other forms of chronic ischemic heart disease	**427**	**Cardiac dysrhythmias**	250	Diabetes mellitus
788	Symptoms involving urinary system				
424	Other diseases of endocardium				

Table 2.4: Comparison of the diagnoses by Doctor AI for a frequent and an infrequent disease code after 200 time step.

Hypertension		**Klinefelter's syndrome**	
ICD9	Description	ICD9	Description
401	Essential hypertension	272	Disorders of lipoid metabolism
272	Disorders of lipoid metabolism	V70	General medical examination
786	Symptoms involving respiratory system and other chest symptoms	V04	Need for prophylactic vaccination and inoculation against certain diseases
V06	Need for prophylactic vaccination and inoculation against combinations of diseases	730	Osteomyelitis, periostitis, and other infections involving bone
790	Nonspecific findings on examination of blood	780	General symptoms
V76	Special screening for malignant neoplasms	783	Symptoms concerning nutrition, metabolism, and development
V04	Need for prophylactic vaccination and inoculation against certain diseases	295	Schizophrenic disorders
V70	General medical examination	V76	Special screening for malignant neoplasms
780	General symptoms	141	Malignant neoplasm of tongue
276	Disorders of fluid, electrolyte, and acid-base balance	V06	Need for prophylactic vaccination and inoculation against combinations of diseases
782	Symptoms involving skin and other integumentary tissue	250	Diabetes mellitus
268	Vitamin D deficiency	782	Symptoms involving skin and other integumentary tissue
719	Other and unspecified disorders of joint	786	Symptoms involving respiratory system and other chest symptoms
427	Cardiac dysrhythmias	208	Leukemia of unspecified cell type
380	Disorders of external ear	401	Essential hypertension
250	Diabetes mellitus	790	Nonspecific findings on examination of blood
599	Other disorders of urethra and urinary tract	280	Iron deficiency anemias
V72	Special investigations and examinations	607	Disorders of penis
789	Other symptoms involving abdomen and pelvis	281	Other deficiency anemias
729	Other disorders of soft tissues	V03	Need for prophylactic vaccination and inoculation against bacterial diseases
682	Other cellulitis and abscess	332	Parkinson's disease
V03	Need for prophylactic vaccination and inoculation against bacterial diseases	255	Disorders of adrenal glands
724	Other and unspecified disorders of back	799	Other ill-defined and unknown causes of morbidity and mortality
V58	Encounter for other and unspecified procedures and aftercare	244	Acquired hypothyroidism
278	Overweight, obesity and other hyperalimentation	V58	Encounter for other and unspecified procedures and aftercare
V82	Special screening for other conditions	151	Malignant neoplasm of stomach
V65	Other persons seeking consultation	294	Persistent mental disorders due to conditions classified elsewhere
585	Chronic kidney disease (CKD)	V72	Special investigations and examinations
274	Gout	344	Other paralytic syndromes
V49	Other conditions influencing health status	146	Malignant neoplasm of oropharynx

CHAPTER 3

MULTI-LAYER REPRESENTATION LEARNING FOR MEDICAL CONCEPTS

Learning efficient representations for concepts is an important basis for many applications such as machine translation or document classification. Proper representations of medical concepts such as diagnosis, medication, procedure codes and visits from Electronic Health Records (EHR) has broad applications in healthcare analytics. Patient EHR data consists of a sequence of visits over time, where each visit includes multiple medical concepts, e.g., diagnosis, procedure, and medication codes. This hierarchical structure provides two types of relational information, namely sequential order of visits and co-occurrence of the codes within a visit. In this chapter, we propose `Med2Vec`, which not only learns the representations for both medical codes and visits from large EHR datasets with over million visits, but also allows us to interpret the learned representations confirmed positively by clinical experts. In the experiments, `Med2Vec` shows significant improvement in prediction accuracy in clinical applications compared to baselines such as Skip-gram, GloVe, and stacked autoencoder, while providing clinically meaningful interpretation.

3.1 Introduction

Discovering efficient representations of high dimensional concepts has been a key challenge in a variety of applications recently [63]. Using various types of neural networks, high-dimensional data can be transformed to continuous real-valued concept vectors that efficiently capture their latent relationship from data. Such succinct representations have been shown to improve the performance of various complex tasks across domains spanning from image processing [72, 73, 74], language modeling [75, 76], word embedding [11, 77], music information retrieval [78], sentiment analysis [79], and multi-modal learning of images and text [80].

Efficient representations for medical concepts is an important, if not essential, element in healthcare applications as well. Medical concepts contain rich latent relationships that cannot be represented by simple one-hot coding [81, Chapter 2.3.2]. For example, pneumonia and bronchitis are clearly more related than pneumonia and obesity. In one-hot coding, such relationship between different codes are not represented. Despite its limitation, many healthcare applications [82, 83] still use the simple sum over one-hot vectors to derive patient feature vectors. To overcome this limitation, it is common in healthcare applications, to rely on carefully designed feature representations [84, 85, 6]. However, this process often involves ad-hoc feature engineering that requires considerable expert medical knowledge and is not scalable nor comprehensive in general.

Recently, studies have shown that it is possible to learn efficient representations of healthcare concepts without medical expertise while significantly improving the performance of various healthcare predictive models. [86, 55, 10, 57, 87] Despite this progress, learning efficient representations of healthcare concepts, however, is still an open challenge. The

difficulty stems from several aspects:

1. Electronic Health Record (EHR) data have a unique structure where the visits are temporally ordered but the medical codes within a visit form an unordered set. A sequence of visits possesses sequential relationship among them which cannot be captured by simply aggregating code-level representations. Moreover, given the demographic information for patients, the structure of EHR becomes more complex.

2. Learned representations should be interpretable. While the interpretability of the representation in healthcare applications is essential, many of the state-of-the-art representation learning methods such as recurrent neural networks (RNN) are difficult to interpret.

3. The algorithm should be scalable enough to handle large EHR datasets with hundreds of thousands of patients and millions of visits.

To address such challenges in healthcare concept representation learning, we propose `Med2Vec` and make the following contributions.

- We propose `Med2Vec`, a simple and robust algorithm to efficiently learn succinct code-, and visit-level representations by using real-world EHR datasets, without depending on expert medical knowledge.

- `Med2Vec` learns interpretable representations and enables clinical applications to offer more than just improved performances. We conducted a detailed user study with clinical experts to validate the interpretability of the resulting representation.

- We conduct experiments to demonstrate the scalability of `Med2Vec`, and show that our model can be readily applied to near 30K medical codes over two large datasets with 3 million and 5.5 million visits, respectively.

- We apply the learned representations to multiple real-world healthcare prediction problems and demonstrate the improved performance enabled by `Med2Vec` compared to several baselines.

In the following section, we discuss related works, then describe our method in section 3. In section 4, we explain experiment design and interpretation method in detail. We present the results and discussion in section 5. Then we conclude this chapter with future work in section 6.

3.2 Preliminaries and Related Work

In this section, we first describe the preliminary ideas used in learning representation for words. Then, we review the algorithms developed for representing healthcare data.

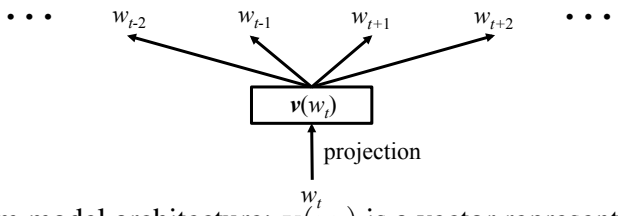


Figure 3.1: Skip-gram model architecture: $\mathbf{v}(w_t)$ is a vector representation for the word w_t. The goal of Skip-gram is to learn vector representations of words that are good at predicting neighboring words.

3.2.1 Learning representation for words

Representation learning of words using neural network based methods have been studied since the early 2000's [75, 88, 89, 90]. Among these techniques, Skip-gram [11] is the basis of many concept representation learning methods, including our own. Skip-gram is able to capture the subtle relationships between words, thus outperforming the previous works in a word analogy task[91].

Given a sequence of words $w_1, w_2, \ldots, w_T$, Skip-gram learns the word representations based on the co-occurrence information of words inside a context window of a predefined size. The key principle of Skip-gram is that a word's representation should be able to predict the neighboring words. The objective of Skip-gram is to maximize the following average log probability.

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

where c is the size of the context window. The conditional probability is defined by the softmax function:

$$p(w_O|w_I) = \frac{\exp\left(v'^{\top}_{w_O} v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left(v'^{\top}_{w} v_{w_I}\right)}$$

where v_w and v'_w are the *input* and *output* vector representations of word w. W is the number of words in the vocabulary. Basically, Skip-gram tries to maximize the softmax probability of the inner product of the center word's vector and its context word's vectors.[1]

Pennington et al. proposed GloVe, [77] which learns another word representations by using a similar principle as Skip-gram. GloVe uses the global word co-occurrence matrix to learn the word representations. Since the global co-occurrence matrix is often sparse, GloVe can be computationally less demanding than Skip-gram, which is a neural network model using the sliding context window. On the other hand, GloVe employs a weighting function that could require a considerable amount tuning effort.

Beyond one level representation like Skip-gram and GloVe, researchers also proposed hierarchical learning representations for the text corpus, which has some analogy to our healthcare setting with two level concepts namely: codes and visits. Le and Mikolov [92]

[1] Mikolov et al. [11] also use hierarchical softmax and negative sampling to speed up the learning process. We focus on the original simple formulation.

proposed to learn representations for paragraphs and words simultaneously by treating paragraphs indicators as words. However, their algorithm assigns a fixed set of vectors for both words and paragraphs in the training data. Moreover, their approach does not capture the sequential order among paragraphs. Skip-thought [93] proposed an encoder-decoder structure: an encoder (Gated Recurrent Units (GRU) in their case) learns a representation for a sentence that is able to regenerate its surrounding sentences (via GRU again). Skip-thought cannot be applied directly to EHR data because unlike words in sentences, the codes in a visit are unordered. Also, the interpretation of Skip-thought model is difficult, as they rely on complex RNNs.

3.2.2 Representation learning in healthcare

Recently researchers start to explore the possibility of efficient representation learning in the medical domain.

Medical text analysis Minarro et al. [94] learns the representations of medical terms by applying Skip-gram to various medical text collected from PubMed, Merck Manuals, Medscape and Wikipedia. De Vine et al. [95] learns the representations of UMLS concepts from free-text patient records and medical journal abstracts. They first replaced the words in documents to UMLS concepts, then applied Skip-gram to learn the distributed representations of the concepts. However, none of them studied longitudinal EHR data with a large number of medical codes.

Structured visit records analysis Skip-gram was directly applied to structured longitudinal visit records to learn the representation of medical codes (*e.g.* diagnosis, medication, procedure codes) [86, 55]. In [86], the authors demonstrated that simply aggregating the learned representation of medical codes to create a visit representation leads to improved predictive performance. However, simply aggregating the code representations is not the optimal method to generate a visit representation as it completely ignores the temporal relations across adjacent visits. We believe that taking advantage of the two-level information (the co-occurrence of codes within a visit and the sequential nature of visits) and the demographic information of patients will give us better representation for both medical codes and patient visits. RNNs have been applied to analysis of longitudinal patient records [10, 57] and can generate both code and patient representations. However, despite their outstanding predictive performance, RNNs are difficult to interpret [87] which limits their applications in healthcare.

3.3 Method

In this section, we describe the proposed algorithm `Med2Vec`. We start by mathematically formulating the EHR data structure and our goal. Then we describe our approach in a top-down fashion. We also explain how to interpret the learned representations. We conclude this section with complexity analysis.

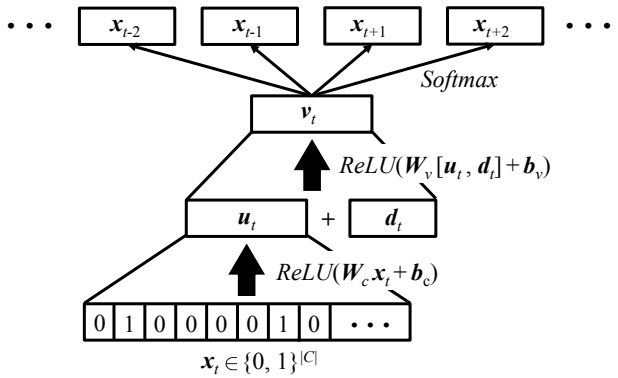


Figure 3.2: Structure of `Med2Vec`: A visit comprised of several medical codes is converted to a binary vector $\mathbf{x}_t \in \{0,1\}^{|\mathcal{C}|}$. The binary vector is then converted to an intermediate visit representation $\mathbf{u}_t$. $\mathbf{u}_t$ is concatenated with a vector of demographic information $\mathbf{d}_t$, and converted to the final visit representation $\mathbf{v}_t$, which is trained to predict its neighboring visits $\ldots, \mathbf{x}_{t-2}, \mathbf{x}_{t-1}, \mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \ldots$

EHR structure and our notation We denote the set of all medical codes $c_1, c_2, \ldots, c_{|\mathcal{C}|}$ in our EHR dataset by $\mathcal{C}$ with size $|\mathcal{C}|$. EHR data for each patient is in the form of a sequence of visits $V_1, \ldots, V_T$ where each visit contains a subset of medical codes $V_t \subseteq \mathcal{C}$. Without loss of generality, all algorithms will be presented for a single patient to avoid cluttered notations. The goal of `Med2Vec` is to learn two types of representations:

Code representations We aim to learn an embedding function $f_C : \mathcal{C} \mapsto \mathbb{R}^m_+$ that maps every code in the set of all medical codes $\mathcal{C}$ to non-negative real-valued vectors of dimension m. The non-negativity constraint is introduced to improve interpretability, as discussed in details in Section 3.3.5.

Visit representations Our second task is to learn another embedding function $f_V : \mathcal{V} \mapsto \mathbb{R}^n$ that maps every visit (a set of medical codes) to a real-valued vector of dimension n. The set $\mathcal{V}$ is the power set of the set of codes $\mathcal{C}$.

3.3.1 Med2Vec architecture

Figure 3.2 depicts the architecture of `Med2Vec`. Given a visit V_t, we use a multi-layer perceptron (MLP) to generate the corresponding visit representation $\mathbf{v}_t$. First, visit V_t is represented by a binary vector $\mathbf{x}_t \in \{0,1\}^{|\mathcal{C}|}$ where the i-th entry is 1 only if $c_i \in V_t$. Then $\mathbf{x}_t$ is converted to an intermediate visit representation $\mathbf{u}_t \in \mathbb{R}^m$ as follows,

$$\mathbf{u}_t = ReLU(\mathbf{W}_c \mathbf{x}_t + \mathbf{b}_c) \tag{3.1}$$

using the code weight matrix $\mathbf{W}_c \in \mathbb{R}^{m \times |\mathcal{C}|}$ and the bias vector $\mathbf{b}_c \in \mathbb{R}^m$. The rectified linear unit is defined as $ReLU(\mathbf{v}) = \max(\mathbf{v}, \mathbf{0})$. Note that $\max()$ applies element-wise to vectors. We use the rectified linear unit (ReLU) as the activation function to enable interpretability, which will be discussed in section 3.3.3.

We concatenate the demographic information $\mathbf{d}_t \in \mathbb{R}^d$, where d is the size of the demographic information vector, to the intermediate visit representation $\mathbf{u}_t$ and create the final visit representation $\mathbf{v}_t \in \mathbb{R}^n$ as follows,

$$\mathbf{v}_t = ReLU(\mathbf{W}_v[\mathbf{u}_t, \mathbf{d}_t] + \mathbf{b}_v)$$

using the visit weight matrix $\mathbf{W}_v \in \mathbb{R}^{n \times (m+d)}$ and the bias vector $\mathbf{b}_v \in \mathbb{R}^n$, where n is the predefined size of the visit representation. We use ReLU once again as the activation function. We discuss our efficient training procedure of the parameters $\mathbf{W}_c, \mathbf{b}_c, \mathbf{W}_v$ and $\mathbf{b}_v$ in the next subsection.

3.3.2 Learning from the visit-level information

As mentioned in the introduction, the sequential information of visits can be exploited for learning efficient representations of visits and potentially codes. We train the MLP using a very straightforward intuition as follows: a visit describes a state in a continuous process that is a patient's clinical experience. Therefore, given a visit representation, we should be able to predict what has happened in the past, and what will happen in the future. Specifically, given a visit representation $\mathbf{v}_t$, we train a softmax classifier that predicts the medical codes of the visits within a context window[2]. We minimize the cross entropy error as follows,

$$\min_{\mathbf{W}_s, \mathbf{b}_s} \frac{1}{T} \sum_{t=1}^{T} \sum_{-w \leq i \leq w, i \neq 0} -\mathbf{x}_{t+i}{}^\top \log \hat{\mathbf{y}}_t - (\mathbf{1} - \mathbf{x}_{t+i})^\top \log(\mathbf{1} - \hat{\mathbf{y}}_t), \tag{3.2}$$

$$\text{where} \qquad \hat{\mathbf{y}}_t = \frac{\exp(\mathbf{W}_s \mathbf{v}_t + \mathbf{b}_s)}{\sum_{j=1}^{|\mathcal{C}|} \exp(\mathbf{W}_s[j,:]\mathbf{v}_t + \mathbf{b}_s[j])}$$

where $\mathbf{W}_s \in \mathbb{R}^{|\mathcal{C}| \times n}$ and $\mathbf{b}_s \in \mathbb{R}^{|\mathcal{C}|}$ are the weight matrix and bias vector for the softmax classifier, w the predefined context window size, $\exp$ the element-wise exponential function, and $\mathbf{1}$ denotes an all one vector. We have used MATLAB's notation for selecting a row in $\mathbf{W}_s$ and a coordinate of $\mathbf{b}_s$.

[2]We also tried predicting the visit representations $\ldots, \mathbf{v}_{t-1}, \mathbf{v}_{t+1}, \ldots$ instead of the medical codes, but obtained poor results.

3.3.3 Learning from the code-level information

As we described in the introduction, healthcare datasets contain two-level information: visit-level sequence information and code-level co-occurrence information. Since the loss function in Eq. (3.2) can efficiently capture the sequence level information, now we need to find a way to use the second source of information, i.e., the intra-visit co-occurrence of the codes.

A natural choice to capture the code co-occurrence information is to use Skip-gram. The main idea would be that the representations for the codes that occur in the same visit should predict each other. To embed Skip-gram in `Med2Vec`, we can train $\mathbf{W}_c \in \mathbb{R}^{m \times |\mathcal{C}|}$ (which also produces intermediate visit level representations) so that the i-th column of $\mathbf{W}_c$ will be the representation for the i-th medical code among total $|\mathcal{C}|$ codes. Note that given the unordered nature of the codes inside a visit, unlike the original Skip-gram, we do not distinguish between the "input" medical code and the "output" medical code. In text, it is sensible to assume that a word can serve a different role as a center word and a context word, whereas in EHR datasets, we cannot classify codes as center or context codes. It is also desirable to learn the representations of different types of codes (*e.g.* diagnosis, medication, procedure code) in the same latent space so that we can capture the hidden relationships between them.

However, coordinate-wise interpretation of Skip-gram codes is not straightforward because the positive and negative values of $\mathbf{W}_c$ make it hard for each coordinate to focus on a single coherent medical concept. For intuitive interpretation, we should learn code representations with non-negative values. Note that in Eq.(3.1), if the binary vector $\mathbf{x}_t$ is a one-hot vector, then the intermediate visit representation $\mathbf{u}_t$ becomes a code representation. Therefore, using the Skip-gram algorithm, we train the non-negative weight $ReLU(\mathbf{W}_c)$ instead of $\mathbf{W}_c$. This will not only use the intra-visit co-occurrence information, but also guarantee non-negative code representations. Moreover, ReLU produces sparse code representations, which further facilitates easier interpretation of the codes.

The code representations to be learned is denoted as a matrix $\mathbf{W}'_c = ReLU(\mathbf{W}_c) \in \mathbb{R}^{m \times |\mathcal{C}|}$. From a sequence of visits $V_1, V_2, \ldots, V_T$, the code-level representations can be learned by maximizing the following objective function,

$$\max_{\mathbf{W}'_c} \quad \frac{1}{T} \sum_{t=1}^{T} \sum_{i:c_i \in V_t} \sum_{j:c_j \in V_t, j \neq i} \log p(c_j|c_i), \tag{3.3}$$

$$\text{where} \quad p(c_j|c_i) = \frac{\exp\left(\mathbf{W}'_c[:,j]^\top \mathbf{W}'_c[:,i]\right)}{\sum_{k=1}^{|\mathcal{C}|} \exp\left(\mathbf{W}'_c[:,k]^\top \mathbf{W}'_c[:,i]\right)}. \tag{3.4}$$

3.3.4 Unified training

The single unified framework can be obtained by adding the two objective functions (3.3) and (3.2) as follows,

$$\begin{aligned}\underset{\mathbf{W}_{c,v,s}, \mathbf{b}_{c,v,s}}{\operatorname{argmin}} \frac{1}{T} \sum_{t=1}^{T} \Big\{ &- \sum_{i:c_i \in V_t} \sum_{j:c_j \in V_t, j \neq i} \log p(c_j|c_i) \\ &+ \sum_{-w \leq k \leq w, k \neq 0} -\mathbf{x}_{t+k}^\top \log \hat{\mathbf{y}}_t - (\mathbf{1} - \mathbf{x}_{t+k})^\top \log(\mathbf{1} - \hat{\mathbf{y}}_t) \Big\}\end{aligned}$$

By combining the two objective functions we learn both code representations and visit representations from the same source of patient visit records, exploiting both intra-visit co-occurrence information as well as inter-visit sequential information at the same time.

3.3.5 Interpretation of learned representations

While the original Skip-gram learns code representations that have interesting properties such as additivity, in healthcare we need stronger interpretability. We need to be able to associate clinical meaning to each dimension of both code and visit representations. Interpreting the learned representations is based on analyzing each coordinate in both code and visit embedding spaces.

Interpreting code representations If information is properly embedded into a lower dimensional non-negative space, each coordinate of the lower dimension can be readily interpreted. Non-negative matrix factorization (NMF) is a good example. Since we trained $ReLU(\mathbf{W}_c) \in \mathbb{R}^{m \times |\mathcal{C}|}$, a non-negative matrix, to represent the medical codes, we can employ a simple method to interpret the meaning of each coordinate of the m-dimensional code embedding space. We can find the top k codes that have the largest values for the i-th coordinate of the code embedding space as follows,

$$\text{argsort}(\mathbf{W}_c[i,:])[1:k]$$

where argsort returns the indices of a vector that index its values in a descending order. By studying the returned medical codes, we can view each coordinate as a disease group. Detailed examples are given in section 3.5.1

Interpreting visit representations To interpret the learned visit vectors, we can use the same principle we used for interpreting the code representation. For the i-th coordinate of the n-dimensional visit embedding space, we can find the top k coordinates of the code embedding space that have the strongest values as follows,

$$\text{argsort}(\mathbf{W}_v[i,:])[1:k]$$

where we use the same argsort as before. Once we obtain a set of code coordinates, we can use the knowledge learned from interpreting the code representations to understand how each visit coordinate is associated with a group of diseases. This simple interpretation is possible because the intermediate visit representation $\mathbf{u}_t$ is a non-negative vector, due to the $ReLU$ activation function.

In the experiments, we also tried to find the input vector $\mathbf{x}_t$ that most activates the target visit coordinate [96, 97]. However, the results were very sensitive to the initial value of $\mathbf{x}_t$,

and even averaging over multiple samples were producing unreliable results.

3.3.6 Complexity analysis

We first analyze the computational complexity of the code-level objective function Eq. (3.3). Without loss of generality, we assume the visit records of all patients are concatenated into a single sequence of visits. Then the complexity for Eq. (3.3) is as follows,

$$\mathcal{O}(T\overline{M}^2|\mathcal{C}|m)$$

where T is the number of visits, $\overline{M}^2$ is the average of squared number of medical codes within a visit, $|\mathcal{C}|$ the number of unique medical codes, m the size of the code representation. The M^2 factor comes from iterating over all possible pairs of codes within a visit. The complexity of the visit-level objective function Eq.(3.2) is as follows,

$$\mathcal{O}(Tw(|\mathcal{C}|(m+n)+mn))$$

where w is the size of the context window, n the size of the visit representation. The added terms come from generating a visit representation via MLP. Since size of code representation m and size of visit representation n generally have the same order of magnitude, we can replace n with m. Furthermore, m is generally smaller than $|\mathcal{C}|$ by at least two orders of magnitude. Therefore the overall complexity of `Med2Vec` can be simplified as follows.

$$\mathcal{O}(T|\mathcal{C}|m(\overline{M}^2+w))$$

Here we notice that $\overline{M}^2$ is generally larger than w. In our work, the average number of codes $\overline{M}$ per visit for two datasets are 7.88 and 3.19 according to Tables 3.1, respectively, whereas we select the window size w to be at most 5 in our experiments. Therefore the complexity of `Med2Vec` is dominated by the code representation learning process, for which we use the

Skip-gram algorithm. This means that exploiting visit-level information to learn efficient representations for both visits and codes does not incur much additional cost.

3.4 Experiments

In this section, we evaluate the performance of `Med2Vec` in both public and proprietary datasets. First we describe the datasets. Then we describe evaluation strategies for code and visit representations, along with implementation details. Then we present the experiment results of code and visit representations with discussion. We conclude with convergence and scalability study. We make the source code of `Med2Vec` publicly available at `https://github.com/mp2893/med2vec`.

3.4.1 Dataset description

We evaluate performance of `Med2Vec` on a dataset provided by Children's Healthcare of Atlanta (CHOA)[3]. We extract visit records from the dataset, where each visit contains several medical codes (*e.g.* diagnosis, medication, procedure codes). The diagnosis codes follow ICD-9 codes, the medication codes are denoted by National Drug Codes (NDC), and the procedure codes follow Category I of Current Procedural Terminology (CPT). We exclude patients who had less that two visits to showcase `Med2Vec`'s ability to use sequential information of visits. The basic statistics of the dataset are summarized in Table 3.1. The data are fully de-identified and do not include any personal health information (PHI).

We divide the dataset into two groups in a 4:1 ratio. The former is used to train `Med2Vec`. The latter is held out for evaluating the visit-level representations, where we train models to predict visit-related labels.

We also use CMS dataset, a publicly available [4] synthetic medical claims dataset. The basic information of CMS is also given in Table 3.1. Compared to CHOA dataset, the CMS

[3] `http://www.choa.org/`

[4] `https://www.cms.gov/Medicare/Quality-Initiatives-Patient-Assessment-Instruments/OASIS/DataSet.html`

Table 3.1: Basic statistics of CHOA and CMS dataset.

Dataset	**CHOA**	**CMS**
# of patients	550,339	831,210
# of visits	3,359,240	5,464,950
Avg. # of visits per patient	6.1	6.57
# of unique medical codes	28,840	21,033
- # of unique diagnosis codes	10,414	14,111
- # of unique medication codes	12,892	N/A
- # of unique procedure codes	5,534	6,922
Avg. # of codes per visit	7.88	3.19
Max # of codes per visit	440	44
(95%, 99%) percentile # of codes per visit	(22, 53)	(9, 13)

dataset has more patients but fewer unique medical codes. The average number of codes per visit is also smaller than that of CHOA dataset. Since CMS dataset is synthetic, we use it only for testing the scalability in section 3.4.7.

3.4.2 Evaluation Strategy of code representations

Qualitative evaluation by medical experts For a comprehensive qualitative evaluation, we perform a *relatedness* test by selecting 100 most frequent diagnosis codes and their 5 closest diagnoses, medications and procedures in terms of cosine similarity. This will allow us to know if the learned representations effectively capture the latent relationships among them. Two medical experts from CHOA check each item and assign *related*, *possible* and *unrelated* labels.

Quantitative evaluation with baselines We use medical code groupers to quantitatively evaluate the code representations. Code groupers are used to collapse individual medical codes into clinically meaningful categories. For example, Clinical Classifications Software (CCS) groups ICD9 diagnosis codes into 283 categories such as tuberculosis, bacterial infection, and viral infection.

We apply K-means clustering to the learned code representations and calculate the normalized mutual information (NMI) based on the group label of each code. We use the

CCS as the ground truth for evaluating the code representation for diagnosis. For medication code evaluation, we use American Hospital Formulary Service (AHFS) pharmacologic-therapeutic classification, which groups NDC codes into 165 categories. For procedure code evaluation, we use the second-level grouping of CPT category I, which groups CPT codes into 115 categories.Thus, we set the number of clusters k to 283, 165, 115 respectively for the diagnosis, medication, procedure code evaluation, which matches the numbers of groups from individual groupers.

For baselines, we use popular methods that efficiently exploit co-occurrence information. Skip-gram (which is used in learning representations of medical concepts by [55, 86]) is trained using Eq. (3.3). GloVe will be trained on the co-occurrence matrix of medical codes, for which we counted the codes co-occurring within a visit. Additionally, we also report well-known baselines such as singular value decomposition on the co-occurrence matrix.

3.4.3 Evaluation strategy of visit representation

We evaluate the quality of the visit representations by performing two visit-level prediction tasks: predicting the future visit and predicting the present status. The former will evaluate a visit representation's potential effectiveness in predictive healthcare while the latter will evaluate the how well it captures the information in the given visit. The details of the two tasks are given below.

Predicting future medical codes: We predict the medical codes that will occur in the next visit using the visit representations. Specifically, given two consecutive visits V_i and V_j, the medical codes $c \in V_j$ will be the target $\mathbf{y}$, the medical codes $c \in V_i$ will be the input $\mathbf{x}$, and we use softmax to predict $\mathbf{y}$ given $\mathbf{x}$. The predictive performance will be measured by Recall@k due to its similarity to the differential diagnosis. Doctors iteratively perform differential diagnosis by generating a list of most likely diseases for an undiagnosed patient based on the available information. We set $k = 30$ to cover the complex cases of CHOA dataset, as over 167,000 visits are assigned with more than 20 medical codes according to

Table 3.1. We predict the grouped medical codes, obtained by the medical groupers used in Section 3.4.2.

Predicting Clinical Risk Groups (CRG) level: A patient's CRG level indicates his severity level. It ranges from 1 to 9, including 5a and 5b. The CRG levels can be divided into two groups: non-severe (CRG 1-5a) and severe (CRG 5b-9). Given a visit, we use logistic regression to predict the binary CRG class associated with the visit. We use Area Under The Curve (AUC) to measure the classification accuracy, as it is more robust to class imbalance in data.

Baselines For baselines, we use the following methods.

Binary vector model (One-hot+): In order to compare with the raw input data, we use the binary vector $\mathbf{x}_t$ as the visit representation.

Stacked autoencoder (SA): Stacked autoencoder is one of the most popular unsupervised representation learning algorithms [98]. Using the binary vector $\mathbf{x}_t$ concatenated with patient demographic information as the input, we train a 3-layer stacked autoencoder (SA) [99] to minimize the reconstruction error.The trained SA will then be used to generate visit representations.

Sum of Skip-gram vectors (Skip-gram+): We first learn the code-level representations with Skip-gram only (Eq. (3.3)). Then for the visit-level representation, we simply add the representations of the codes within the visit. This approach was proven very effective for heart failure prediction in [86]. We append patient demographic information at the end.

Sum of GloVe vectors (GloVe+): We perform the same process as Skip-gram+, but use GloVe vectors instead of Skip-gram vectors. We use the recommended hyperparameter setting from [77].

Evaluation details We use the held-off dataset, which was *not* used to learn the code and visit representations, to perform the two prediction tasks. The held-off dataset contains 672,110 visits assigned with CRG levels. In order to train the predictors, we divide the held-

out data to training and testing folds with ration 4:1. Both softmax and logistic regression are trained for 10 epochs on the training fold. We perform 5-fold cross validation for each task and report the average performance. For all baseline models and `Med2Vec`, we use age, sex and ethnicity as the demographic information in the input data.

3.4.4 Implementation and training details

For learning code and visit representations using `Med2Vec` and all baselines, we use Adadelta [100] in a mini-batch fashion. For Skip-gram, SA and `Med2Vec`, we use 1,000 visits[5] per batch. For GloVe, we use 1,000 non-zero entries of the co-occurrence matrix per batch. The optimization terminates after a fixed number of epochs. In section 3.4.6, we show the relationship between training epochs and the performance. We also show the convergence behavior of `Med2Vec` and the baselines in section 3.4.7.

`Med2Vec`, Skip-gram, GloVe and SA are implemented with Theano 0.7.0 [101]. K-means clustering for the code-level evaluation and SVD are performed using Scikit-learn 0.14.1. Softmax and logistic regression models for the visit-level evaluation are implemented with Keras 0.3.1, and trained for 10 epochs. All tasks are executed on a machine equipped with Intel Xeon E5-2697v3, 256GB memory and two Nvidia K80 Tesla cards.

We train multiple models using various hyperparameter settings. For all models we vary the size of the code representations m (or the size of the hidden layer for SA), and the number of training epochs. Additionally for `Med2Vec`, we vary the size of the visit representations n, and the size of the visit context window w.

To alleviate the curse of dimensionality when training the softmax classifier (Eq.(3.2)) of `Med2Vec`, we always use the medical code groupers of section 3.4.2 so that the softmax classifier is trained to predict the grouped medical codes instead of the exact medical codes. To confirm the impact of this strategy, we train an additional `Med2Vec` without using the medical code groupers.

[5] for efficient computation, we preprocessed the EHR dataset so that the visit records of all patients are concatenated into a single sequence of visits.

Table 3.2: Average score of the medical codes from the relatedness test. 2 was assigned for *related*, 1 for *possible* and 0 for *unrelated*

Average	Diagnosis	Medication	Procedure
1.34	1.59	0.95	1.47

Table 3.3: Clustering NMI of the diagnosis, medication and procedure code representations of various models. All models learned 200 dimensional code vectors. All models except SVD were trained for 10 epochs.

Model	Diagnosis	Medication	Procedure
SVD ($\sigma\mathbf{V}^\top$)	0.1824	0.0843	0.1781
Skip-gram	0.2251	0.1216	0.2432
GloVe	0.4205	0.2163	0.3499
Med2Vec	0.2328	0.1089	0.21

3.4.5 Results of the code-level evaluation

Table 3.2 shows the average score of the medical codes from the qualitative code evaluation. On average, `Med2Vec` successfully captures the relationship between medical codes. However, `Med2Vec` seems to have a hard time capturing proper representation of medications. This is due to the precise nature the medication prescription. For example, `Med2Vec` calculated that *Ofloxacin*, an antibiotic sometimes used to treat middle-ear infection, was related to *sensorineural hearling loss* (SNHL), an inner-ear problem. On the surface level, this is a wrong relationship. But `Med2Vec` can be seen as capturing the deeper relationship between medical concepts that is not always clear on the surface level.

Table 3.3 shows the clustering NMI of diagnosis, medication and procedure codes, measured for various models. `Med2Vec` shows more or less similar conformity to the existing groupers as Skip-gram. SVD shows the weakest conformity among all models. GloVe exhibits significantly stronger conformity than any other models. Exploiting the global co-occurrence matrix seems to help learn code representations where similar codes are closer to each other in terms of Euclidean distance.

However, the degree of conformity of the code representations to the groupers does not necessarily indicate how well the code representations capture the hidden relationships. For

example, CCS categorizes ICD9 224.4 *Benign neoplasm of cornea* as CCS 47 *Other and unspecified benign neoplasm*, and ICD9 370.00 *Unspecified corneal ulcer* as CCS 91 *Other eye disorders*. But the two diagnosis codes are both eye related problems, and they could be considered related in that sense. Therefore we recommend the readers use the evaluation results for comparing the performance between `Med2Vec` and other baselines, rather than for measuring the absolute performance.

In the following visit-level evaluation, we show the dominant predictive performance of `Med2Vec` indicates that code representations' strong conformity to the groupers does not necessarily imply good visit representations.

3.4.6 Results of the visit-level evaluation

The first row of Figure 3.3 shows the Recall@30 for predicting the future medical codes. First, in all of the experiments, `Med2Vec` achieves the highest performance, despite the fact that it is constrained to be positive and interpretable. The second observation is that `Med2Vec`'s performance is robust to choice of the hyperparameters in a wide range of values. Comparing to a more volatile performance of Skip-gram, we can see that including the visit information in training not only improves the performance, but also stabilizes it too.

Another fascinating aspect of the results is the overfitting pattern in different algorithms. Increasing the code representation size degrades the performance of all of the algorithms, as it leads to overfitting. Similar behavior can be seen as we train GloVe+ for more epochs which suggests early stopping technique should be used in representation learning [102]. For `Med2Vec`, increasing the visit representation size n seems to have the strongest influence to its predictive performance.

The bottom row of figures in Figure 3.3 shows the AUC for predicting the CRG class of the given visit. The overfitting patterns are not as prominent as the previous task. This is due to the different nature of the two prediction tasks. While the goal of CRG prediction is to predict a value related to the current visit, predicting the future codes is taking a step away

Table 3.4: Performance comparison of two Med2Vec models. The top row was trained with the grouped code as mentioned in section 3.4.4. The bottom row was trained without using the groupers. Both models were trained for 10 epochs with $m, n = 200$, $w = 1$.

Model	Future code prediction	CRG prediction
Grouped codes	0.7605	0.9150
Exact codes	0.7574	0.9155

from the current visit. This different nature of the two tasks also contributes to the better performance of One-hot+ on the CRG prediction. One-hot+ contains the entire information of the given visit, although in a very high-dimensional space. Therefore predicting the CRG level, which has a tight relationship with the medical codes within a visit, is an easier task for One-hot+ than predicting the future codes.

Table 3.4 shows the performance comparison between two different `Med2Vec` models. The top model is trained with the grouped codes as explained in section 3.4.4, while the bottom models is trained with the exact codes. Considering the marginal difference of the CRG prediction AUC, it is evident that our strategy to alleviate the curse of dimensionality was beneficial. Moreover, using the grouped codes will improve the training speed as the softmax function will require less computation.

3.4.7 Convergence behavior and scalability

We compare the convergence behavior of `Med2Vec` with Skip-gram (Eq. (3.3)), GloVe and SA. For SA, we measure the convergence behavior of a single-layer. We train the models for 50 epochs and plot the normalized difference of the loss value $\frac{\mathcal{L}_t - \mathcal{L}_{t-1}}{\mathcal{L}_t}$, where $\mathcal{L}_t$ denotes the loss value at time t. We also study the scalability of all models except One-hot+, as there is no representation learning in it. We vary the size of the training data and plot the time taken for each model to run one epoch.

The left figure of Fig 3.4 shows the convergence behavior of all models when trained on the CHOA dataset. SA shows the most stable convergence behavior, which is natural given that we used a single-layer SA, a much less complex model compared to GloVe,

Skip-gram and Med2Vec. All models except SA seem to reach convergence after 10 epochs of training. Note that Med2Vec shows similar, if not better convergence behavior compared to Skip-gram even with added complexity.

The center figure of Fig 3.4 shows the minutes taken to train all models for one epoch using the CHOA dataset. As we have analzyed in section ssec:complexity, Med2Vec takes essentially the same time to train for one epoch. Both Skip-gram and Med2Vec, however, takes longer than SA and GloVe. This is mainly due to having the softmax function for training the code representations. GloVe, which is trained on the very sparse co-occurrence matrix naturally takes the least time to train.

The right figure of Fig 3.4 shows the training time when using the CMS dataset. Note that Med2Vec and Skip-gram takes similar time to train as SA. This is due to the smaller number of codes per visit, which is the computationally dominating factor of both Med2Vec and Skip-gram. GloVe takes less time as the number of unique codes are smaller in the CMS dataset. SA, on the other hand, takes more time because the number of visits have doubled while the the number of unique codes is about 73% of that of the CHOA dataset.

3.5 Interpretation

Given the importance of interpretability in healthcare, we demonstrate three stages of interpretability for our model in collaboration with the medical experts from CHOA. First, to analyze the learned code representations we show top five medical codes for each of six coordinates of the code embedding space and explain the characteristic of each coordinate. This way, we show how we can annotate each dimension of the code embedding space with clinical concepts. The six coordinates are specifically chosen so that they can be used in the later stages. Second, we demonstrate the interpretability of Med2Vec's visit representations by analyzing the meaning of two coordinates in the visit embedding space.

Finally, we extend the interpretability of Med2Vec to a real-world task, the CRG prediction, and analyze the medical codes that have strong influence on the CRG level.

Table 3.5: Medical codes with the strongest value in six different coordinates of the 200 dimensional code embedding space. We choose ten medical codes per coordinate. Shortened descriptions of diagnosis codes are compensated by their ICD9 codes. Medications and procedures are appended with (R) and (P) respectively.

Coordinate 112	Coordinate 152	Coordinate 141
Kidney replaced by transplant (V42.0)	X-ray, knee (P)	Cystic fibrosis (277.02)
Hb-SS disease without crisis (282.61)	X-ray, thoracolumbar (P)	Intracranial injury (854.00)
Heart replaced by transplant (V42.1)	Accidents in public building (E849.6)	Persistent mental disorders (294.9)
RBC antibody screening (P)	Activities involving gymnastics (E005.2)	Subdural hemorrhage (432.1)
Complications of transplanted	Struck by objects/persons in sports (E917.0)	Neurofibromatosis (237.71)
bone marrow (996.85)	Encounter for removal of sutures (V58.32)	Other conditions of brain (348.89)
Sickle-cell disease (282.60)	Struck by object in sports (E917.5)	Conductive hearing loss (389.05)
Liver replaced by transplant (V42.7)	Unspecified fracture of ankle (824.8)	Unspecified causes of encephalitis,
Hb-SS disease with crisis (282.62)	Accidents occurring in place for	myelitis, encephalomyelitis (323.9)
Prograf PO (R)	recreation and sport (E849.4)	Sensorineural hearing loss (389.15)
Complications of transplanted heart (996.83)	Activities involving basketball (E007.6)	Intracerebral hemorrhage (431)
Coordinate 184	**Coordinate 190**	**Coordinate 199**
	Down's syndrome (758.0)	
Pain in joint, shoulder region (719.41)	Congenital anomalies (759.89)	Infantile cerebral palsy (343.9)
Pain in joint, lower leg (719.46)	Tuberous sclerosis (759.5)	Congenital quadriplegia (343.2)
Pain in joint, ankle and foot (719.47)	Anomalies of larynx, trachea,	Congenital diplegia (343.0)
Pain in joint, multiple sites (719.49)	and bronchus (748.3)	Quadriplegia (344.00)
Generalized convulsive epilepsy (345.10)	Autosomal deletions (758.39)	Congenital hemiplegia (343.1)
Pain in joint, upper arm (719.42)	Conditions due to anomaly of unspecified	Baclofen 10mg tablet (R)
Cerebral artery occlusion (434.91)	chromosome (758.9)	Wheelchair management (P)
MRI, brain (780.59)	Acquired hypothyroidism (244.9)	Tracheostomy status (V44.0)
Other joint derangement (718.81)	Conditions due to chromosome anomalies (758.89)	Paraplegia (344.1)
Fecal occult blood (790.6)	Anomalies of spleen (759.0)	Baclofen 5mg/ml liquid (R)
	Conditions due to autosomal anomalies (758.5)	

Once we learn the logistic regression weight $\mathbf{w}_{LR}$ for the CRG prediction, we can extract knowledge from the learned weights by analyzing the visit coordinates to which the weights are strongly connected.

Instead of analyzing the visit coordinates, however, we propose an approximate way of directly finding out which code coordinate plays an important role in predicting the CRG class. Our goal is to find $\mathbf{u}_t$ such that maximizes the output activation as follows[6]

$$\mathbf{u}_t^\star = \underset{\mathbf{u}_t, \|\mathbf{u}_t\|_2=1, \mathbf{u}_t \succeq 0}{\operatorname{argmax}} \left[ReLU(\mathbf{W}_v \mathbf{u}_t + \mathbf{b}_v)\right]^\top \mathbf{w}_{LR} \tag{3.5}$$

Given the fact that $ReLU(\cdot)$ is an increasing function (not-strictly though), we make an approximation and find the solution without the $ReLU(\cdot)$ term. The approximate solution can be found in closed form $\mathbf{u}_t^\star \propto (\mathbf{W}_v^\top \mathbf{w}_{LR})_+$. Finally, we calculate the element-wise product of $\mathbf{u}_t^\star$ and $\max(\mathbf{W}_c + \mathbf{b}_c)$. This is to take into account the fact that each code

[6] As we are interested in influential codes, we assume the demographic information vector is zero vector and omit it for ease of notation.

coordinate has different maximum value. Therefore, instead of simply selecting the code coordinate with the strongest connection to the CRG level, we consider each coordinate's maximum ability to activate the positive CRG prediction.

The resulting vector will show the maximum influence each code coordinate can have on the CRG prediction.

3.5.1 Results

Table 3.5 shows top ten codes with the largest value in each of the six coordinates of the code embedding space. The coordinate 112 is clearly related to sickle-cell disease and organ transplant. The two are closely related in that sickle cell disease can be treated with bone-marrow transplant. Prograf is a medication used for preventing organ rejection. Coordinate 152 groups medical codes related to sports-related injuries, specifically broken bones. Coordinate 141 is related to brain injuries and hearing loss due to the brain injuries. Neurofibromatosis(NF) is also related to this coordinate because it can cause tumors along the nerves in the brain. Cystic fibrosis(CF) seems to be a weak link in this group as it is only related to NF in the sense that both NF and CF are genetically inherited. Coordinate 184 clearly represents medical codes related to epilepsy. Epilepsy is often accompanied by convulsions, which can cause joint pain. Cerebral artery occlusion is related epilepsy in the sense that epileptic seizures can be a manifestation of cerebral arterial occlusive diseases[103]. Also, both blood in feces and the joint pain can be attributed to Henoch–Schönlein purpura, a disease primarily found in children. Coordinate 190 groups diseases that are caused by congenital chromosome anomalies, especially the autosome. Acquired hypothyroidism seems to be an outlier of this coordinate. Coordinate 199 is strongly related to congenital paralysis. Baclofen is a medication used as a muscle relaxer. Quadraplegia patients can have weakened respiratory function due to impaired abdominal muscles[104], in which case tracheostomy could be required.

We now analyze two visit coordinates: coordinate 50 and 41. Both visit coordinates

have the strongest connection to the logistic regression learned for the CRG prediction. For visit coordinate 50, the two strongest code coordinates connected to it are code coordinates 112 and 152. Then naturally, from our analysis above, we can easily see that visit coordinate 50 is strongly activated by sickle-cell disease and sports-related injuries. For visit coordinate 41, code coordinates 141 and 184 have the strongest connection. Again from the analysis above, we can directly infer that visit coordinate 41 can be seen as a patient group consisting of brain damage & hearing loss patients and epilepsy patients. By repeating this process, we can find the code coordinates that are likely to strongly influence the CRG level.

However, finding the influential code coordinates for CRG level can be achieved without analyzing the visit representation if we use Eq.(3.5). Applying Eq.(3.5) to the logistic regression weight of the CRG prediction, we learned that code coordinates 190 and 199 are the two strongest influencer of the CRG level. Using the analysis from above, we can naturally conclude that patients suffering from congenital chromosome anomalies or congenital paralysis are most likely to be considered to be in severe states, which is obviously true in any clinical setting.

The results indicate that interpretable visit representations learned by `Med2Vec` not only improve the prediction accuracy, but also identify the influential clinical concepts.

3.6 Conclusion

In this chapter, we proposed `Med2Vec`, a scalable two layer neural network for learning lower dimensional representations for medical concepts. `Med2Vec` incorporates both code co-occurence information and visit sequence information of the EHR data which improves the accuracy of both code and visit representations. Throughout several experiments, we successfully demonstrated the superior performance of `Med2Vec` in two predictive tasks and provided clinical interpretation of the learned representations.

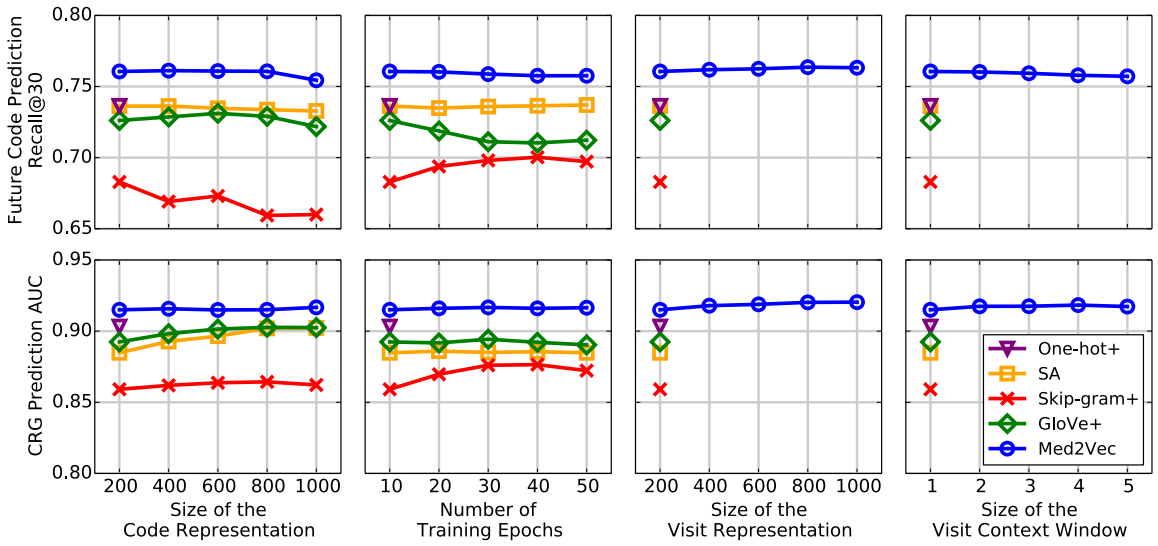


Figure 3.3: The top row and the bottom row respectively show the Recall@30 for predicting the future medical codes and the AUC for predicting the CRG class when changing different hyperparameters. The basic configuration for Med2Vec is $m, n = 200$, $w = 1$, and the training epoch set to 10. The basic configuration for all baseline models is 200 for code representation size (or hidden layer size) and training epoch also set to 10. In each column, we change one hyperparameter while fixing others to the basic configuration.

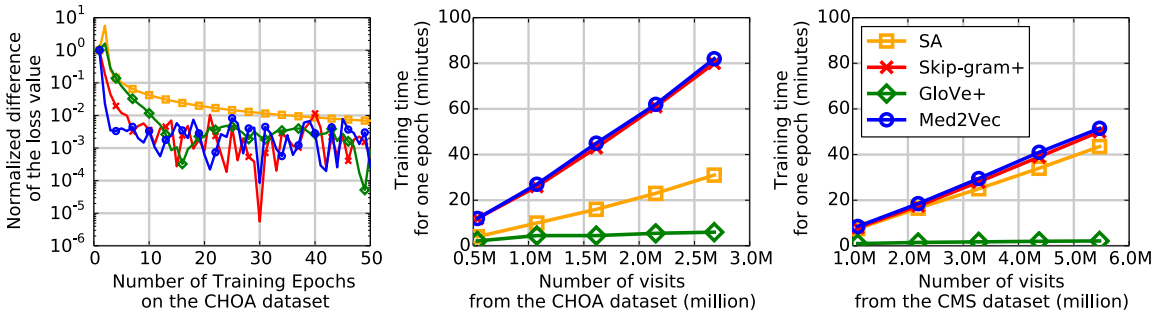


Figure 3.4: The first figure shows the convergence behavior of all models on the CHOA dataset. The second and third figures show the relationship between the training time and the dataset size for all models respectively using the CHOA dataset and the CMS dataset.

CHAPTER 4

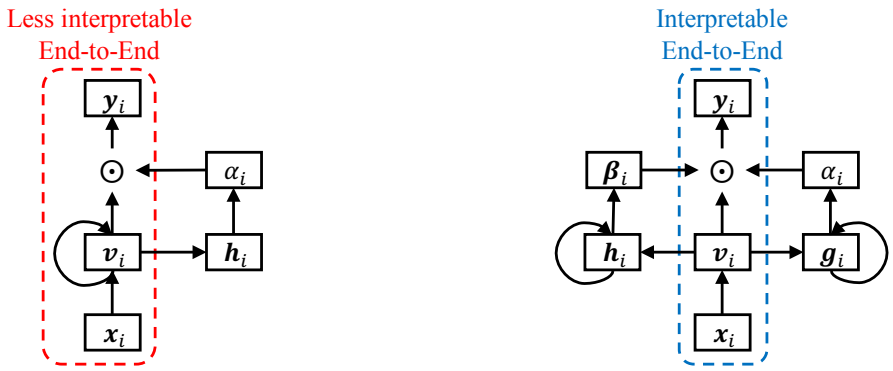
RETAIN: AN INTERPRETABLE PREDICTIVE MODEL FOR HEALTHCARE USING REVERSE TIME ATTENTION MECHANISM

4.1 Introduction

The broad adoption of Electronic Health Record (EHR) systems has opened the possibility of applying clinical predictive models to improve the quality of clinical care. Several systematic reviews have underlined the care quality improvement using predictive analysis [105, 106, 107, 108]. EHR data can be represented as temporal sequences of high-dimensional clinical variables (e.g., diagnoses, medications and procedures), where the sequence ensemble represents the documented content of medical visits from a single patient. Traditional machine learning tools summarize this ensemble into aggregate features, ignoring the temporal and sequence relationships among the feature elements. The opportunity to improve both predictive accuracy and interpretability is likely to derive from effectively modeling temporality and high-dimensionality of these event sequences.

Accuracy and interpretability are two dominant features of successful predictive models. There is a common belief that one has to trade accuracy for interpretability using one of three types of traditional models [109]: 1) identifying a set of rules (*e.g.* via decision trees [110]), 2) case-based reasoning by finding similar patients (*e.g.* via k-nearest neighbors [111] and distance metric learning [84]), and 3) identifying a list of risk factors (*e.g.* via LASSO coefficients [112]). While interpretable, all of these models rely on aggregated features, ignoring the temporal relation among features inherent to EHR data. As a consequence, model accuracy is sub-optimal. Latent-variable time-series models, such as [113, 114], account for temporality, but often have limited interpretation due to abstract state variables.

Recently, recurrent neural networks (RNN) have been successfully applied in modeling



(a) Standard attention model (b) RETAIN model

Figure 4.1: Common attention models vs. RETAIN, using folded diagrams of RNNs. (a) Standard attention mechanism: the recurrence on the hidden state vector $\mathbf{v}_i$ hinders interpretation of the model. (b) Attention mechanism in RETAIN: The recurrence is on the attention generation components ($\mathbf{h}_i$ or $\mathbf{g}_i$) while the hidden state $\mathbf{v}_i$ is generated by a simpler more interpretable output.

sequential EHR data to predict diagnoses [57] and model encounter sequences [10, 115]. But, the gain in accuracy from use of RNNs is at the cost of model output that is notoriously difficult to interpret. While there have been several attempts at directly interpreting RNNs [116, 117, 87], these methods are not sufficiently developed for application in clinical care.

We have addressed this limitation using a modeling strategy known as RETAIN, a two-level neural attention model for sequential data that provides detailed interpretation of the prediction results while retaining the prediction accuracy comparable to RNN. To this end, RETAIN relies on an attention mechanism modeled to represent the behavior of physicians during an encounter. A distinguishing feature of RETAIN (see Figure 4.1) is to leverage sequence information using an attention generation mechanism, while learning an interpretable representation. And emulating physician behaviors, RETAIN examines a patient's past visits in reverse time order, facilitating a more stable attention generation. As a result, RETAIN identifies the most meaningful visits and quantifies visit specific features that contribute to the prediction.

RETAIN was tested on a large health system EHR dataset with 14 million visits completed by 263K patients over an 8 year period. We compared predictive accuracy of RETAIN

to traditional machine learning methods and to RNN variants using a case-control dataset to predict a future diagnosis of heart failure. The comparative analysis demonstrates that RETAIN achieves comparable performance to RNN in both accuracy and speed and significantly outperforms traditional models. Moreover, using a concrete case study and visualization method, we demonstrate how RETAIN offers an intuitive interpretation.

4.2 Methodology

We first describe the structure of sequential EHR data and our notation, then follow with a general framework for predictive analysis in healthcare using EHR, followed by details of the RETAIN method.

EHR Structure and our Notation. The EHR data of each patient can be represented as a time-labeled sequence of multivariate observations. Assuming we use r different variables, the n-th patient of N total patients can be represented by a sequence of $T^{(n)}$ tuples $(t_i^{(n)}, \mathbf{x}_i^{(n)}) \in \mathbb{R} \times \mathbb{R}^r, i = 1, \ldots, T^{(n)}$. The timestamps $t_i^{(n)}$ denotes the time of the i-th visit of the n-th patient and $T^{(n)}$ the number of visits of the n-th patient. To minimize clutter, we describe the algorithms for a single patient and have dropped the superscript (n) whenever it is unambiguous. The goal of predictive modeling is to predict the label at each time step $\mathbf{y}_i \in \{0, 1\}^s$ or at the end of the sequence $\mathbf{y} \in \{0, 1\}^s$. The number of labels s can be more than one.

For example, in encounter sequence modeling (ESM) [10], each visit (*e.g.* encounter) of a patient's visit sequence is represented by a set of varying number of medical codes $\{c_1, c_2, \ldots, c_n\}$. c_j is the j-th code from the vocabulary $\mathcal{C}$. Therefore, in ESM, the number of variables $r = |\mathcal{C}|$ and input $\mathbf{x}_i \in \{0, 1\}^{|\mathcal{C}|}$ is a binary vector where the value one in the j-th coordinate indicates that c_j was documented in i-th visit. Given a sequence of visits $\mathbf{x}_1, \ldots, \mathbf{x}_T$, the goal of ESM is, for each time step i, to predict the codes occurring at the next visit $\mathbf{x}_2, \ldots, \mathbf{x}_{T+1}$, with the number of labels $s = |\mathcal{C}|$.

In case of learning to diagnose (L2D) [57], the input vector $\mathbf{x}_i$ consists of continuous

clinical measures. If there are r different measurements, then $\mathbf{x}_i \in \mathbb{R}^r$. The goal of L2D is, given an input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$, to predict the occurrence of a specific disease ($s = 1$) or multiple diseases ($s > 1$). Without loss of generality, we will describe the algorithm for ESM, as L2D can be seen as a special case of ESM where we make a single prediction at the end of the visit sequence.

In the rest of this section, we will use the abstract symbol RNN to denote any recurrent neural network variants that can cope with the vanishing gradient problem [118], such as LSTM [58], GRU [119], and IRNN [120], with any depth (number of hidden layers).

4.2.1 Preliminaries on Neural Attention Models

Attention based neural network models are being successfully applied to image processing [121, 122, 123, 124], natural language processing [3, 125, 126] and speech recognition [127]. The utility of the attention mechanism can be seen in the language translation task [3] where it is inefficient to represent an entire sentence with one fixed-size vector because neural translation machines finds it difficult to translate the given sentence represented by a single vector.

Intuitively, the attention mechanism for language translation works as follows: given a sentence of length S in the original language, we generate $\mathbf{h}_1, \ldots, \mathbf{h}_S$, to represent the words in the sentence. To find the j-th word in the target language, we generate attentions α_i^j for $i = 1, \ldots, S$ for each word in the original sentence. Then, we compute the context vector $\mathbf{c}_j = \sum_i \alpha_i^j \mathbf{h}_i$ and use it to predict the j-th word in the target language. In general, the attention mechanism allows the model to focus on a specific word (or words) in the given sentence when generating each word in the target language.

We rely on a conceptually similar temporal attention mechanism to generate interpretable prediction models using EHR data. Our model framework is motivated by and mimics how doctors attend to a patient's needs and explore the patient record, where there is a focus on specific clinical information (e.g., key risk factors) working from the present to the past.

4.2.2 Reverse Time Attention Model RETAIN

Figure 4.2 shows the high-level overview of our model, where a central feature is to delegate a considerable portion of the prediction responsibility to the process for generating attention weights. This is intended to address, in part, the difficulty with interpreting RNNs where the recurrent weights feed past information to the hidden layer. Therefore, to consider both the visit-level and the variable-level (individual coordinates of $\mathbf{x}_i$) influence, we use a linear embedding of the input vector $\mathbf{x}_i$. That is, we define

$$\mathbf{v}_i = \mathbf{W}_{emb}\mathbf{x}_i, \qquad \text{(Step 1)}$$

where $\mathbf{v}_i \in \mathbb{R}^m$ denotes the embedding of the input vector $\mathbf{x}_i \in \mathbb{R}^r$, m the size of the embedding dimension, $\mathbf{W}_{emb} \in \mathbb{R}^{m \times r}$ the embedding matrix to learn. We can alternatively use more sophisticated yet interpretable representations such as those derived from multilayer perceptron (MLP) [96, 97]. MLP has been used for representation learning in EHR data [12].

We use two sets of weights, one for the visit-level attention and the other for variable-level attention, respectively. The scalars $\alpha_1, \ldots, \alpha_i$ are the visit-level attention weights that govern the influence of each visit embedding $\mathbf{v}_1, \ldots, \mathbf{v}_i$. The vectors $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_i$ are the variable-level attention weights that focus on each coordinate of the visit embedding $v_{1,1}, v_{1,2}, \ldots, v_{1,m}, \ldots, v_{i,1}, v_{i,2}, \ldots, v_{i,m}$.

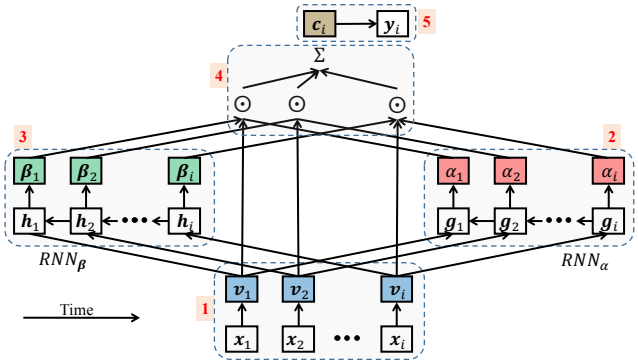


Figure 4.2: Unfolded view of RETAIN's architecture: Given input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_i$, we predict the label $\mathbf{y}_i$. **Step 1**: Embedding, **Step 2**: generating α values using RNN_α, **Step 3**: generating $\boldsymbol{\beta}$ values using $\mathrm{RNN}_{\boldsymbol{\beta}}$, **Step 4**: Generating the context vector using attention and representation vectors, and **Step 5**: Making prediction. Note that in Steps 2 and 3 we use RNN in the reversed time.

We use two RNNs, RNN_α and $\mathrm{RNN}_{\boldsymbol{\beta}}$, to separately generate α's and $\boldsymbol{\beta}$'s as follows,

$$
\begin{aligned}
\mathbf{g}_i, \mathbf{g}_{i-1}, \ldots, \mathbf{g}_1 &= \mathrm{RNN}_\alpha(\mathbf{v}_i, \mathbf{v}_{i-1}, \ldots, \mathbf{v}_1), \\
e_j &= \mathbf{w}_\alpha^\top \mathbf{g}_j + b_\alpha, \quad \text{for} \quad j = 1, \ldots, i \\
\alpha_1, \alpha_2, \ldots, \alpha_i &= \mathrm{Softmax}(e_1, e_2, \ldots, e_i) \qquad \text{(Step 2)} \\
\mathbf{h}_i, \mathbf{h}_{i-1}, \ldots, \mathbf{h}_1 &= \mathrm{RNN}_{\boldsymbol{\beta}}(\mathbf{v}_i, \mathbf{v}_{i-1}, \ldots, \mathbf{v}_1) \\
\boldsymbol{\beta}_j &= \tanh\left(\mathbf{W}_{\boldsymbol{\beta}} \mathbf{h}_j + \mathbf{b}_{\boldsymbol{\beta}}\right) \quad \text{for} \quad j = 1, \ldots, i, \qquad \text{(Step 3)}
\end{aligned}
$$

where $\mathbf{g}_i \in \mathbb{R}^p$ is the hidden layer of RNN_α at time step i, $\mathbf{h}_i \in \mathbb{R}^q$ the hidden layer of $\mathrm{RNN}_{\boldsymbol{\beta}}$ at time step i and $\mathbf{w}_\alpha \in \mathbb{R}^p, b_\alpha \in \mathbb{R}, \mathbf{W}_{\boldsymbol{\beta}} \in \mathbb{R}^{m \times q}$ and $\mathbf{b}_{\boldsymbol{\beta}} \in \mathbb{R}^m$ are the parameters to learn. The hyperparameters p and q determine the hidden layer size of RNN_α and $\mathrm{RNN}_{\boldsymbol{\beta}}$, respectively. Note that for prediction at each timestamp, we generate a new set of attention vectors α and $\boldsymbol{\beta}$. For simplicity of notation, we do not include the index for predicting at different time steps. In Step 2, we can use Sparsemax [128] instead of Softmax for sparser attention weights.

As noted, RETAIN generates the attention vectors by running the RNNs backward in time; i.e., RNN_α and $\mathrm{RNN}_{\boldsymbol{\beta}}$ both take the visit embeddings in a reverse order $\mathbf{v}_i, \mathbf{v}_{i-1}, \ldots, \mathbf{v}_1$.

Running the RNN in reversed time order also offers computational advantages since the reverse time order allows us to generate e's and $\boldsymbol{\beta}$'s that dynamically change their values when making predictions at different time steps $i = 1, 2, \ldots, T$. This ensures that the attention vectors are modified at each time step, increasing the computational stability of the attention generation process.[1]

Using the generated attentions, we obtain the context vector $\mathbf{c}_i$ for a patient up to the i-th visit as follows,

$$\mathbf{c}_i = \sum_{j=1}^{i} \alpha_j \boldsymbol{\beta}_j \odot \mathbf{v}_j, \tag{Step 4}$$

where $\odot$ denotes element-wise multiplication. We use the context vector $\mathbf{c}_i \in \mathbb{R}^m$ to predict the true label $\mathbf{y}_i \in \{0, 1\}^s$ as follows,

$$\widehat{\mathbf{y}}_i = \mathrm{Softmax}(\mathbf{W}\mathbf{c}_i + \mathbf{b}), \tag{Step 5}$$

where $\mathbf{W} \in \mathbb{R}^{s \times m}$ and $\mathbf{b} \in \mathbb{R}^s$ are parameters to learn. We use the cross-entropy to calculate the classification loss as follows,

$$\mathcal{L}(\mathbf{x}_1, \ldots, \mathbf{x}_T) = -\frac{1}{N} \sum_{n=1}^{N} \frac{1}{T^{(n)}} \sum_{i=1}^{T^{(n)}} \left(\mathbf{y}_i^\top \log(\widehat{\mathbf{y}}_i) + (\mathbf{1} - \mathbf{y}_i)^\top \log(\mathbf{1} - \widehat{\mathbf{y}}_i) \right) \tag{4.1}$$

where we sum the cross entropy errors from all dimensions of $\widehat{\mathbf{y}}_i$. In case of real-valued output $\mathbf{y}_i \in \mathbb{R}^s$, we can change the cross-entropy in Eq. (4.1) to, for example, mean squared error.

Overall, our attention mechanism can be viewed as the inverted architecture of the standard attention mechanism for NLP [3] where the words are encoded by RNN and the attention weights are generated by MLP. In contrast, our method uses MLP to embed the visit information to preserve interpretability and uses RNN to generate two sets of

[1] For example, feeding visit embeddings in the original order to RNN_α and $\mathrm{RNN}_{\boldsymbol{\beta}}$ will generate the same e_1 and $\boldsymbol{\beta}_1$ for every time step $i = 1, 2, \ldots, T$. Moreover, in many cases, a patient's recent visit records deserve more attention than the old records. Then we need to have $e_{j+1} > e_j$ which makes the process computationally unstable for long sequences.

attention weights, recovering the sequential information as well as mimicking the behavior of physicians. Note that we did not use the timestamp of each visit in our formulation. Using timestamps, however, provides a small improvement in the prediction performance. We propose a method to use timestamps in Supplementary 4.6.

4.3 Interpreting RETAIN

Finding the visits that contribute to prediction are derived using the largest α_i, which is straightforward. However, finding influential variables is slightly more involved as a visit is represented by an ensemble of medical variables, each of which can vary in its predictive contribution. The contribution of each variable is determined by $\mathbf{v}$, $\boldsymbol{\beta}$ and α, and interpretation of α alone informs which visit is influential in prediction but not why.

We propose a method to interpret the end-to-end behavior of RETAIN. By keeping α and $\boldsymbol{\beta}$ values fixed as the attention of doctors, we analyze changes in the probability of each label $y_{i,1}, \ldots, y_{i,s}$ in relation to changes in the original input $x_{1,1}, \ldots, x_{1,r}, \ldots, x_{i,1}, \ldots, x_{i,r}$. The $x_{j,k}$ that yields the largest change in $y_{i,d}$ will be the input variable with highest contribution. More formally, given the sequence $\mathbf{x}_1, \ldots, \mathbf{x}_i$, we are trying to predict the probability of the output vector $\mathbf{y}_i \in \{0, 1\}^s$, which can be expressed as follows

$$p(\mathbf{y}_i|\mathbf{x}_1, \ldots, \mathbf{x}_i) = p(\mathbf{y}_i|\mathbf{c}_i) = \text{Softmax}\left(\mathbf{W}\mathbf{c}_i + \mathbf{b}\right) \tag{4.2}$$

where $\mathbf{c}_i \in \mathbb{R}^m$ denotes the context vector. According to Step 4, $\mathbf{c}_i$ is the sum of the visit embeddings $\mathbf{v}_1, \ldots, \mathbf{v}_i$ weighted by the attentions α's and $\boldsymbol{\beta}$'s. Therefore Eq (4.2) can be rewritten as follows,

$$p(\mathbf{y}_i|\mathbf{x}_1, \ldots, \mathbf{x}_i) = p(\mathbf{y}_i|\mathbf{c}_i) = \text{Softmax}\left(\mathbf{W}\Big(\sum_{j=1}^{i} \alpha_j \boldsymbol{\beta}_j \odot \mathbf{v}_j\Big) + \mathbf{b}\right) \tag{4.3}$$

Using the fact that the visit embedding $\mathbf{v}_i$ is the sum of the columns of $\mathbf{W}_{emb}$ weighted by

each element of $\mathbf{x}_i$, Eq (4.3) can be rewritten as follows,

$$p(\mathbf{y}_i|\mathbf{x}_1,\ldots,\mathbf{x}_i) = \text{Softmax}\Big(\mathbf{W}\Big(\sum_{j=1}^{i}\alpha_j\boldsymbol{\beta}_j \odot \sum_{k=1}^{r} x_{j,k}\mathbf{W}_{emb}[:,k]\Big) + \mathbf{b}\Big)$$
$$= \text{Softmax}\Big(\sum_{j=1}^{i}\sum_{k=1}^{r} x_{j,k}\,\alpha_j\mathbf{W}\Big(\boldsymbol{\beta}_j \odot \mathbf{W}_{emb}[:,k]\Big) + \mathbf{b}\Big) \tag{4.4}$$

where $x_{j,k}$ is the k-th element of the input vector $\mathbf{x}_j$. Eq (4.4) can be completely deconstructed to the variables at each input $\mathbf{x}_1,\ldots,\mathbf{x}_i$, which allows for calculating the contribution ω of the k-th variable of the input $\mathbf{x}_j$ at time step $j \leq i$, for predicting $\mathbf{y}_i$ as follows,

$$\omega(\mathbf{y}_i, x_{j,k}) = \underbrace{\alpha_j\mathbf{W}(\boldsymbol{\beta}_j \odot \mathbf{W}_{emb}[:,k])}_{\text{Contribution coefficient}}\ \underbrace{x_{j,k}}_{\text{Input value}}\ , \tag{4.5}$$

where the index i of $\mathbf{y}_i$ is omitted in the α_j and $\boldsymbol{\beta}_j$. As we have described in Section 4.2.2, we are generating α's and $\boldsymbol{\beta}$'s at time step i in the visit sequence $\mathbf{x}_1,\ldots,\mathbf{x}_T$. Therefore the index i is always assumed for α's and $\boldsymbol{\beta}$'s. Additionally, Eq (4.5) shows that when we are using a binary input value, the coefficient itself is the contribution. However, when we are using a non-binary input value, we need to multiply the coefficient and the input value $x_{j,k}$ to correctly calculate the contribution.

4.4 Experiments

We compared performance of `RETAIN` to RNNs and traditional machine learning methods. Given space constraints, we only report the results on the learning to diagnose (L2D) task and summarize the encounter sequence modeling (ESM) in Supplementary 4.8. The `RETAIN` source code is publicly available at `https://github.com/mp2893/retain`.

4.4.1 Experimental setting

Source of data: The dataset consists of electronic health records from Sutter Health. The patients are 50 to 80 years old adults chosen for a heart failure prediction model study.

Table 4.1: Statistics of EHR dataset. (D:Diagnosis, R:Medication, P:Procedure)

# of patients	263,683	Avg. # of codes in a visit	3.03
# of visits	14,366,030	Max # of codes in a visit	62
Avg. # of visits per patient	54.48	Avg. # of Dx codes in a visit	1.83
# of medical code groups	615 (D:283, R:94, P:238)	Max # of Dx in a visit	42

From the encounter records, medication orders, procedure orders and problem lists, we extracted visit records consisting of diagnosis, medication and procedure codes. To reduce the dimensionality while preserving the clinical information, we used existing medical groupers to aggregate the codes into input variables. The details of the medical groupers are given in the Supplementary 4.7. A profile of the dataset is summarized in Table 4.1.

Implementation details: We implemented `RETAIN` with Theano 0.8 [101]. For training the model, we used Adadelta [100] with the mini-batch of 100 patients. The training was done in a machine equipped with Intel Xeon E5-2630, 256GB RAM, two Nvidia Tesla K80's and CUDA 7.5.

Baselines: For comparison, we completed the following models.

- **Logistic regression (LR)**: We compute the counts of medical codes for each patient based on all her visits as input variables and normalize the vector to zero mean and unit variance. We use the resulting vector to train the logistic regression.
- **MLP**: We use the same feature construction as **LR**, but put a hidden layer of size 256 between the input and output.
- **RNN**: RNN with two hidden layers of size 256 implemented by the GRU. Input sequences $\mathbf{x}_1, \ldots, \mathbf{x}_i$ are used. Logistic regression is applied to the top hidden layer. We use two layers of RNN of to match the model complexity of `RETAIN`.
- **RNN+$\boldsymbol{\alpha}_M$**: One layer single directional RNN (hidden layer size 256) along time to generate the input embeddings $\mathbf{v}_1, \ldots, \mathbf{v}_i$. We use the MLP with a single hidden layer of size 256 to generate the visit-level attentions $\alpha_1, \ldots, \alpha_i$. We use the input embeddings $\mathbf{v}_1, \ldots, \mathbf{v}_i$ as the input to the MLP. This baseline corresponds to Figure 4.1a.

- **RNN+α_R**: This is similar to **RNN+α_M** but uses the reverse-order RNN (hidden layer size 256) to generate the visit-level attentions $\alpha_1, \ldots, \alpha_i$. We use this baseline to confirm the effectiveness of generating the attentions using reverse time order.

The comparative visualization of the baselines are provided in Supplementary 4.9. We use the same implementation and training method for the baselines as described above. The details on the hyper-parameters, regularization and drop-out strategies for the baselines are described in Supplementary 4.7.

Evaluation measures: Model accuracy was measured by:

- **Negative log-likelihood** that measures the model loss on the test set. The loss can be calculated by Eq (4.1).

- **Area Under the ROC Curve (AUC)** of comparing $\widehat{y}_i$ with the true label y_i. AUC is more robust to imbalanced positive/negative prediction labels, making it appropriate for evaluation of classification accuracy in the heart failure prediction task.

We also report the bootstrap (10,000 runs) estimate of the standard deviation of the evaluation measures.

4.4.2 Heart Failure Prediction

Objective: Given a visit sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$, we predicted if a primary care patient will be diagnosed with heart failure (HF). This is a special case of ESM with a single disease outcome at the end of the sequence. Since this is a binary prediction task, we use the logistic sigmoid function instead of the Softmax in Step 5.

Cohort construction: From the source dataset, 3,884 cases are selected and approximately 10 controls are selected for each case (28,903 controls). The case/control selection criteria are fully described in the supplementary section. Cases have index dates to denote the date they are diagnosed with HF. Controls have the same index dates as their corre-

Table 4.2: Heart failure prediction performance of RETAIN and the baselines

Model	Test Neg Log Likelihood	AUC	Train Time / epoch	Test Time
LR	0.3269 ± 0.0105	0.7900 ± 0.0111	0.15s	0.11s
MLP	0.2959 ± 0.0083	0.8256 ± 0.0096	0.25s	0.11s
RNN	0.2577 ± 0.0082	0.8706 ± 0.0080	10.3s	0.57s
RNN+α_M	0.2691 ± 0.0082	0.8624 ± 0.0079	6.7s	0.48s
RNN+α_R	0.2605 ± 0.0088	$\mathbf{0.8717} \pm 0.0080$	10.4s	0.62s
RETAIN	$\mathbf{0.2562} \pm 0.0083$	0.8705 ± 0.0081	10.8s	0.63s

sponding cases. We extract diagnosis codes, medication codes and procedure codes in the 18-months window before the index date.

Training details: The patient cohort was divided into the training, validation and test sets in a 0.75:0.1:0.15 ratio. The validation set was used to determine the values of the hyper-parameters. See Appendix 4.7 for details of hyper-parameter tuning.

Results: Logistic regression and MLP underperformed compared to the four temporal learning algorithms (Table 4.2). RETAIN is comparable to the other RNN variants in terms of prediction performance while offering the interpretation benefit.

Note that RNN+α_R model are a degenerated version of RETAIN with only scalar attention, which is still a competitive model as shown in table 4.2. This confirms the efficiency of generating attention weights using the RNN. However, RNN+α_R model only provides scalar visit-level attention, which is not sufficient for healthcare applications. Patients often receives several medical codes at a single visit, and it will be important to distinguish their relative importance to the target. We show such a case study in section 4.4.3.

Table 4.2 also shows the scalability of RETAIN, as its training time (the number of seconds to train the model over the entire training set once) is comparable to RNN. The test time is the number of seconds to generate the prediction output for the entire test set. We use the mini-batch of 100 patients when assessing both training and test times. RNN takes longer than RNN+α_M because of its two-layer structure, whereas RNN+α_M uses a

single layer RNN. The models that use two RNNs (RNN, RNN+α_R, RETAIN)[2] take similar time to train for one epoch. However, each model required a different number of epochs to converge. RNN typically takes approximately 10 epochs, RNN+α_M and RNN+α_R 15 epochs and RETAIN 30 epochs. Lastly, training the attention models (RNN+α_M, RNN+α_R and RETAIN) for ESM would take considerably longer than L2D, because ESM modeling generates context vectors at each time step. RNN, on the other hand, does not require additional computation other than embedding the visit to its hidden layer to predict target labels at each time step. Therefore, in ESM, the training time of the attention models will increase linearly in relation to the length of the input sequence.

4.4.3 Model Interpretation for Heart Failure Prediction

We evaluated the interpretability of RETAIN in the HF prediction task by choosing a HF patient from the test set and calculating the contribution of the variables (medical codes in this case) to diagnostic prediction. The patient suffered from skin problems, *skin disorder* (SD), *benign neoplasm* (BN), *excision of skin lesion* (ESL), for some time before showing symptoms of HF, *cardiac dysrhythmia* (CD), *heart valve disease* (HVD) and *coronary atherosclerosis* (CA), and then a diagnosis of HF (Figure 4.3). We can see that skin-related codes from the earlier visits made little contribution to HF prediction as expected. RETAIN properly puts much attention to the HF-related codes that occurred in recent visits.

To confirm RETAIN's ability to exploit the sequence information of the EHR data, we reverse the visit sequence of Figure 4.3a and feed it to RETAIN. Figure 4.3b shows the contribution of the medical codes of the reversed visit record. HF-related codes in the past are still making positive contributions, but not as much as they did in Figure 4.3a. Figure 4.3b also emphasizes RETAIN's superiority to interpretable, but stationary models such as logistic regression. Stationary models often aggregate past information and remove the temporality from the input data, which can mistakenly lead to the same risk prediction for

[2]The RNN baseline uses two layers of RNN, RNN+α_R uses one for visit embedding and one for generating α, RETAIN uses each for generating α and $\boldsymbol{\beta}$

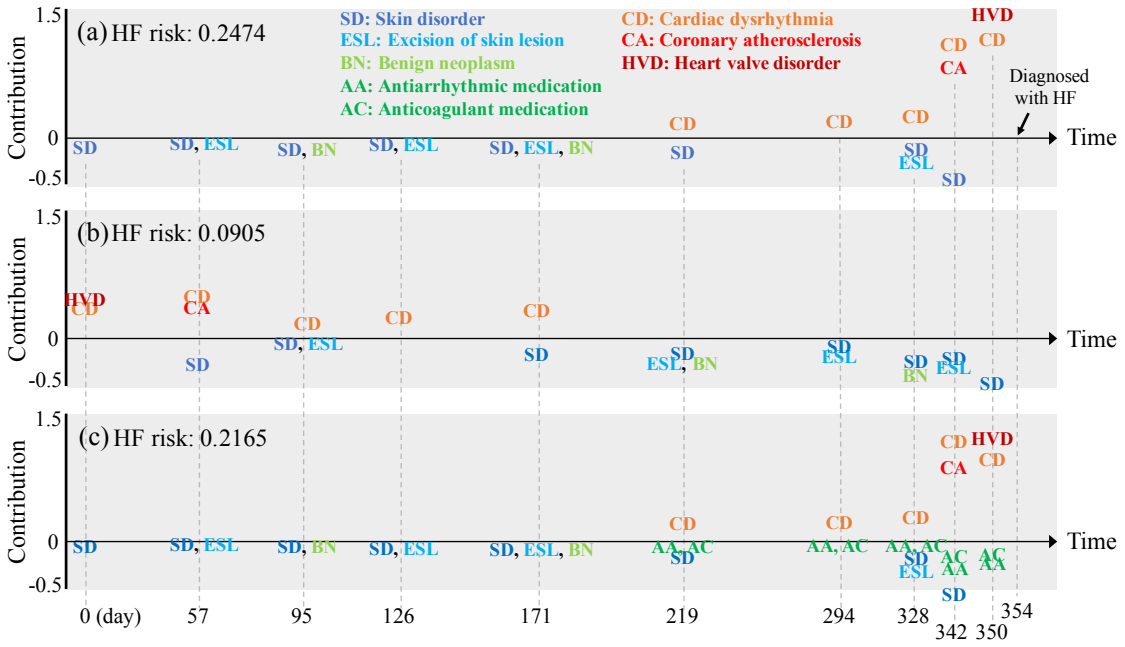


Figure 4.3: (a) Temporal visualization of a patient's visit records where the contribution of variables for diagnosis of heart failure (HF) is summarized along the x-axis (*i.e.* time) with the y-axis indicating the magnitude of visit and code specific contributions to HF diagnosis. (b) We reverse the order of the visit sequence to see if `RETAIN` can properly take into account the modified sequence information. (c) Medication codes are added to the visit record to see how it changes the behavior of `RETAIN`.

Figure 4.3a and 4.3b. `RETAIN`, however, can correctly digest the sequence information and calculates the HF risk score of 9.0%, which is significantly lower than that of Figure 4.3a.

Figure 4.3c shows how the contributions of codes change when selected medication data are used in the model. We added two medications from day 219: *antiarrhythmics* (AA) and *anticoagulants* (AC), both of which are used to treat *cardiac dysrhythmia* (CD). The two medications make a negative contributions, especially towards the end of the record. The medications decreased the positive contributions of *heart valve disease* and *cardiac dysrhythmia* in the last visit. Indeed, the HF risk prediction (0.2165) of Figure 4.3c is lower than that of Figure 4.3a (0.2474). This suggests that taking proper medications can help the patient in reducing their HF risk.

4.5 Conclusion

Our approach to modeling event sequences as predictors of HF diagnosis suggest that complex models can offer both superior predictive accuracy and more precise interpretability. Given the power of RNNs for analyzing sequential data, we proposed RETAIN, which preserves RNN's predictive power while allowing a higher degree of interpretation. The key idea of RETAIN is to improve the prediction accuracy through a sophisticated attention generation process, while keeping the representation learning part simple for interpretation, making the entire algorithm accurate and interpretable. RETAIN trains two RNN in a reverse time order to efficiently generate the appropriate attention variables. For future work, we plan to develop an interactive visualization system for RETAIN and evaluating RETAIN in other healthcare applications.

Supplementary

4.6 A method to use the timestamps

As before, we use $t_i^{(n)}$ to represent the timestamp of the i-th visit of the n-th patient. In the following, we suppress the superscript (n) to avoid cluttered notation. Note that the timestamp t_i can be anything that provides the temporal information of the i-th visit: the number of days from the first visit, the number of days between two consecutive visits, or the number of days until the index date of some event such as heart failure diagnosis.

In order to use the timestamps, we modify Step 2 and Step 3 in Section 4.2.2 as follows:

$$
\begin{aligned}
\mathbf{g}_i, \mathbf{g}_{i-1}, \ldots, \mathbf{g}_1 &= \mathrm{RNN}_\alpha(\mathbf{v}_i^{'}, \mathbf{v}_{i-1}^{'}, \ldots, \mathbf{v}_1^{'}), \\
e_j &= \mathbf{w}_\alpha^\top \mathbf{g}_j + b_\alpha, \quad \text{for} \quad j = 1, \ldots, i \\
\alpha_1, \alpha_2, \ldots, \alpha_i &= \mathrm{Softmax}(e_1, e_2, \ldots, e_i) \\
\mathbf{h}_i, \mathbf{h}_{i-1}, \ldots, \mathbf{h}_1 &= \mathrm{RNN}_{\boldsymbol{\beta}}(\mathbf{v}_i^{'}, \mathbf{v}_{i-1}^{'}, \ldots, \mathbf{v}_1^{'}) \\
\boldsymbol{\beta}_j &= \tanh\left(\mathbf{W}_{\boldsymbol{\beta}} \mathbf{h}_j + \mathbf{b}_{\boldsymbol{\beta}}\right) \quad \text{for} \quad j = 1, \ldots, i, \\
\text{where } \mathbf{v}_i^{'} &= [\mathbf{v}_i, t_i]
\end{aligned}
$$

where we use $\mathbf{v}_i^{'}$, the concatenation of the visit embedding $\mathbf{v}_i$ and the timestamp t_i, to generate the attentions α and $\boldsymbol{\beta}$. However, when obtaining the context vector $\mathbf{c}_i$ as per Step 4, we use $\mathbf{v}_i$, not $\mathbf{v}_i^{'}$ to match the dimensionality. The entire process could be understood such that we use the temporal information not to embed each visit, but to calculate the attentions for the entire visit sequence. This is consistent with our modeling approach where we lose the sequential information in embedding the visit with MLP, then recover the sequential information by generating the attentions using the RNN. By using the temporal information, specifically the log of the number of days from the first visit, we were able to improve the heart failure prediction AUC by 0.003 without any hyper-parameter tuning.

4.7 Details of the experiment settings

4.7.1 Hyper-parameter Tuning

We used the validation set to tune the hyper-parameters: visit embedding size m, RNN_α's hidden layer size p, $\mathrm{RNN}_{\boldsymbol{\beta}}$'s hidden layer size q, L_2 regularization coefficient, and drop-out rates.

L_2 regularization was applied to all weights except the ones in RNN_α and $\mathrm{RNN}_{\boldsymbol{\beta}}$. Two separate drop-outs were used on the visit embedding $\mathbf{v}_i$ and the context vector $\mathbf{c}_i$. We

performed the random search with predefined ranges $m, p, q \in \{32, 64, 128, 200, 256\}$, $L_2 \in \{0.1, 0.01, 0.001, 0.0001\}$, $dropout_{\mathbf{v}_i}, dropout_{\mathbf{c}_i} \in \{0.0, 0.2, 0.4, 0.6, 0.8\}$. We also performed the random search with m, p and q fixed to 256.

The final value we used to train `RETAIN` for heart failure prediction is $m, p, q = 128$, $dropout_{\mathbf{v}_i} = 0.6$, $dropout_{\mathbf{c}_i} = 0.6$ and 0.0001 for the L_2 regularization coefficient.

4.7.2 Code Grouper

Diagnosis codes, medication codes and procedure codes in the dataset are respectively using International Classification of Diseases (ICD-9), Generic Product Identifier (GPI) and Current Procedural Terminology (CPT).

Diagnosis codes are grouped by Clinical Classifications Software for ICD-9[3] which reduces the number of diagnosis code from approximately 14,000 to 283. Medication codes are grouped by Generic Product Identifier Drug Group[4] which reduces the dimension to from approximately 151,000 to 96. Procedure codes are grouped by Clinical Classifications Software for CPT[5], which reduces the number of CPT codes from approximately 9,000 to 238.

4.7.3 Training Specifics of the Basline Models

- **LR**: We use 0.01 L_2 regularization coefficient for the logistic regression weight.
- **MLP**: We use drop-out rate 0.6 on the output of the hidden layer. We use 0.0001 L_2 regularization coefficient for the hidden layer weight and the logistic regression weight.
- **RNN**: We use drop-out rate 0.6 on the outputs of both hidden layers. We use 0.0001 L_2 regularization coefficient for the logistic regression weight. The dimension size of both hidden layers is 256.

[3] https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp
[4] http://www.wolterskluwercdi.com/drug-data/medi-span-electronic-drug-file/
[5] https://www.hcup-us.ahrq.gov/toolssoftware/ccs_svcsproc/ccssvcproc.jsp

Table 4.3: Qualifying ICD-9 codes for heart failure

ICD-9 Code	Description
398.91	Rheumatic heart failure (congestive)
402.01	Malignant hypertensive heart disease with heart failure
402.11	Benign hypertensive heart disease with heart failure
402.91	Unspecified hypertensive heart disease with heart failure
404.01	Hypertensive heart and chronic kidney disease, malignant, with heart failure and with chronic kidney disease stage I through stage IV, or unspecified
404.03	Hypertensive heart and chronic kidney disease, malignant, with heart failure and with chronic kidney disease stage V or end stage renal disease
404.11	Hypertensive heart and chronic kidney disease, benign, with heart failure and with chronic kidney disease stage I through stage IV, or unspecified
404.13	Hypertensive heart and chronic kidney disease, benign, with heart failure and chronic kidney disease stage V or end stage renal disease
404.91	Hypertensive heart and chronic kidney disease, unspecified, with heart failure and with chronic kidney disease stage I through stage IV, or unspecified
404.93	Hypertensive heart and chronic kidney disease, unspecified, with heart failure and chronic kidney disease stage V or end stage renal disease
428.0	Congestive heart failure, unspecified
428.1	Left heart failure
428.20	Systolic heart failure, unspecified
428.21	Acute systolic heart failure
428.22	Chronic systolic heart failure
428.23	Acute on chronic systolic heart failure
428.30	Diastolic heart failure, unspecified
428.31	Acute diastolic heart failure
428.32	Chronic diastolic heart failure
428.33	Acute on chronic diastolic heart failure
428.40	Combined systolic and diastolic heart failure, unspecified
428.41	Acute combined systolic and diastolic heart failure
428.42	Chronic combined systolic and diastolic heart failure
428.43	Acute on chronic combined systolic and diastolic heart failure
428.9	Heart failure, unspecified

- **RNN+α_M**: We use drop-out rate 0.4 on the output of the hidden layer and 0.6 on the output of the context vector $\sum_i \alpha_i \mathbf{v}_i$. We use 0.0001 L_2 regularization coefficient for the hidden layer weight of the MLP that generates α's and the logistic regression weight. The dimension size of the hidden layers in both RNN and MLP is 256.

- **RNN+α_R**: We use drop-out rate 0.4 on the output of the hidden layer and 0.6 on the output of the context vector $\sum_i \alpha_i \mathbf{v}_i$. We use 0.0001 L_2 regularization coefficient for the hidden layer weight of the RNN that generates α's and the logistic regression weight. The dimension size of the hidden layers in both RNNs is 256.

4.7.4 Heart Failure Case/Control Selection Criteria

Case patients were 40 to 85 years of age at the time of HF diagnosis. HF diagnosis (HFDx) is defined as: 1) Qualifying ICD-9 codes for HF appeared in the encounter records or medication orders. Qualifying ICD-9 codes are displayed in Table 4.3. 2) a minimum of three clinical encounters with qualifying ICD-9 codes had to occur within 12 months of each other, where the date of diagnosis was assigned to the earliest of the three dates. If the time span between the first and second appearances of the HF diagnostic code was greater than 12 months, the date of the second encounter was used as the first qualifying encounter. The date at which HF diagnosis was given to the case is denoted as HFDx. Up to ten eligible controls (in terms of sex, age, location) were selected for each case, yielding an overall ratio of 9 controls per case. Each control was also assigned an index date, which is the HFDx of the matched case. Controls are selected such that they did not meet the operational criteria for HF diagnosis prior to the HFDx plus 182 days of their corresponding case. Control subjects were required to have their first office encounter within one year of the matching HF case patient's first office visit, and have at least one office encounter 30 days before or any time after the case's HF diagnosis date to ensure similar duration of observations among cases and controls.

4.8 Results on encounter sequence modeling

Objective: Given a sequence of visits $\mathbf{x}_1, \ldots, \mathbf{x}_T$, the goal of encounter sequence modeling is, for each time step i, to predict the codes occurring at the next visit $\mathbf{x}_2, \ldots, \mathbf{x}_{T+1}$. In this experiment, we focus on predicting the diagnosis codes in the encounter sequence, so we create a separate set of labels $\mathbf{y}_1, \ldots, \mathbf{y}_T$ that do not contain non-diagnosis codes such as medication codes or procedure codes. Therefore $\mathbf{y}_i$ will contain diagnosis codes from the next visit $\mathbf{x}_{i+1}$.

Dataset: We divide the entire dataset described in Table 4.1 into 0.75:0.10:0.15 ratio,

respectively for training set, validation set, and test set.

Baseline: We use the same baseline models we used for HF prediction. However, since we are predicting 283 binary labels now, we replace the logistic regression function with the Softmax function. The drop-out and L_2 regularization policies remain the same.

For LR and MLP, at each step i, we aggregate maximum ten past input vectors[6] $\mathbf{x}_{i-9}, \ldots, \mathbf{x}_i$ to create a pseudo-context vector $\widehat{\mathbf{c}}_i$. LR applies the Softmax function on top of $\widehat{\mathbf{c}}_i$. MLP places a hidden layer on top of $\widehat{\mathbf{c}}_i$ then applies the Softmax function.

Evaluation metric: We use the negative log likelihood Eq (4.1) on the test set to evaluate the model performance. We also use Recall@$\boldsymbol{k}$ as an additional metric to measure the prediction accuracy.

- **Recall@$\boldsymbol{k}$:** Given a sequence of visits $\mathbf{x}_1, \ldots, \mathbf{x}_T$, we evaluate the model performance based on how accurately it can predict the diagnosis codes $\mathbf{y}_1, \ldots, \mathbf{y}_T$. We use the average Recall@$\boldsymbol{k}$, which is expressed as below,

$$\frac{1}{N}\sum_{n=1}^{N}\frac{1}{T^{(n)}}\sum_{i=1}^{T^{(n)}}\text{Recall@}\boldsymbol{k}(\widehat{\mathbf{y}}_i), \quad \textit{where} \quad \text{Recall@}\boldsymbol{k}(\widehat{\mathbf{y}}_i) = \frac{|\text{argsort}(\widehat{\mathbf{y}}_i)[:k] \cap nonzero(\mathbf{y}_i)|}{|nonzero(\mathbf{y}_i)|}$$

  where $argsort$ returns a list of indices that will decrementally sort a given vector and $nonzero$ returns a list of indices of the coordinates with non-zero values. We use Recall@$\boldsymbol{k}$ because of its similar nature to the way a human physician performs the differential diagnostic procedure, which is to generate a list of most likely diseases for an undiagnosed patient, then perform medical practice until the true disease, or diseases are determined.

Prediction accuracy: Table 4.4 displays the prediction performance of `RETAIN` and the baselines. We use $k = 5, 10$ for Recall@$\boldsymbol{k}$ to allow a reasonable number of prediction trials, as well as cover complex patients who often receive multiple diagnosis codes at a single visit.

[6]We also tried aggregating all past input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_i$, but the performance was slightly worse than using just ten.

Table 4.4: Encounter diagnosis prediction performance of RETAIN and the baselines

Model	Negative Likelihood	Recall@5	Recall@10
LR	0.0288	43.15	55.84
MLP	0.0267	50.72	65.02
RNN	**0.0258**	**55.18**	**69.09**
RNN+α_M	0.0262	52.15	65.81
RNN+α_R	0.0259	53.89	67.45
RETAIN	0.0259	54.25	67.74

RNN shows the best prediction accuracy for encounter diagnosis prediction. However, considering the purpose of encounter diagnosis prediction, which is to assist doctors to provide quality care for the patient, black-box behavior of RNN makes it unattractive as a clinical tool. On the other hand, RETAIN performs as well as other attention models, only slightly inferior to RNN, provides full interpretation of its prediction behavior, making it a feasible solution for clinical applications.

The interesting finding in Table 4.4 is that MLP is able to perform as accurately as RNN+α_M in terms of Recall@10. Considering the fact that MLP uses aggregated information of past ten visits, we can assume that encounter diagnosis prediction depends more on the frequency of disease occurrences rather than the order in which they occurred. This is quite different from the HF prediction task, where stationary models (LF, MLP) performed significantly worse than sequential models.

4.9 Illustration and comparison of the baselines

Figure 4.4 illustrates the baselines used in the experiments and shows the relationship among them.

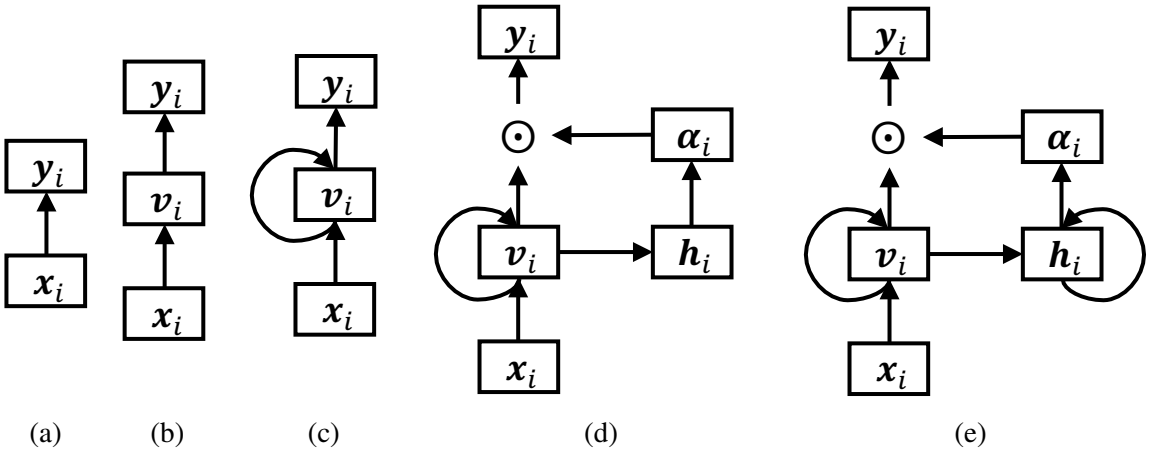


Figure 4.4: Graphical illustration of the baselines: (a) Logistic regression (LR), (b) Multilayer Perceptron (MLP), (c) Recurrent neural network (RNN), (d) RNN with attention vectors generated via an MLP (RNN+$\boldsymbol{\alpha}_M$), (e) RNN with attention vectors generated via an RNN (RNN+$\boldsymbol{\alpha}_R$). `RETAIN` is given in Figure 4.1b.

CHAPTER 5

GRAM: GRAPH-BASED ATTENTION MODEL FOR HEALTHCARE REPRESENTATION LEARNING

Deep learning methods exhibit promising performance for predictive modeling in healthcare, but two important challenges remain:

- *Data insufficiency:* Often in healthcare predictive modeling, the sample size is insufficient for deep learning methods to achieve satisfactory results.

- *Interpretation:* The representations learned by deep learning methods should align with medical knowledge.

To address these challenges, we propose GRaph-based Attention Model (GRAM) that supplements electronic health records (EHR) with hierarchical information inherent to medical ontologies. Based on the data volume and the ontology structure, GRAM represents a medical concept as a combination of its ancestors in the ontology via an attention mechanism.

We compared predictive performance (*i.e.* accuracy, data needs, interpretability) of GRAM to various methods including the recurrent neural network (RNN) in two sequential diagnoses prediction tasks and one heart failure prediction task. Compared to the basic RNN, GRAM achieved 10% higher accuracy for predicting diseases rarely observed in the training data and 3% improved area under the ROC curve for predicting heart failure using an order of magnitude less training data. Additionally, unlike other methods, the medical concept representations learned by GRAM are well aligned with the medical ontology. Finally, GRAM exhibits intuitive attention behaviors by adaptively generalizing to higher level concepts when facing data insufficiency at the lower level concepts.

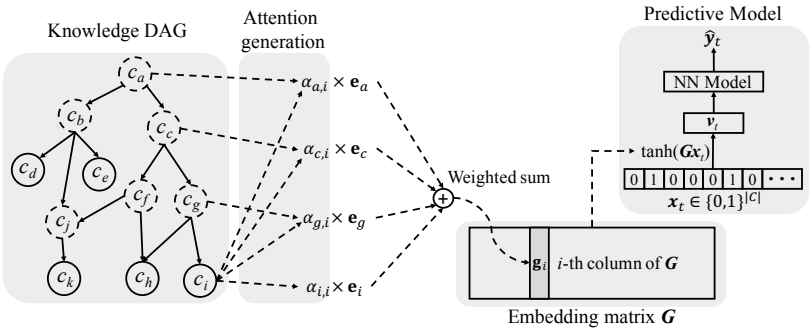
5.1 Introduction

The rapid growth in volume and diversity of healthcare data from electronic health records (EHR) and other sources is motivating the use of predictive modeling to improve care for individual patients. In particular, novel applications are emerging that use deep learning methods such as word embedding [12, 55], recurrent neural networks (RNN) [129, 10, 13, 130], convolutional neural networks (CNN) [131] or stacked denoising autoencoders (SDA) [53, 56], demonstrating significant performance enhancement for diverse prediction tasks. Deep learning models appear to perform significantly better than logistic regression or multilayer perceptron (MLP) models that depend, to some degree, on expert feature construction [57, 132].

Training deep learning models typically requires large amounts of data that often cannot be met by a single health system or provider organization. Sub-optimal model performance can be particularly challenging when the focus of interest is predicting onset of a rare disease. For example, using Doctor AI [10], we discovered that RNN alone was ineffective at predicting the onset of diseases such as cerebral degenerations (e.g. Leukodystrophy, Cerebral lipidoses) or developmental disorders (e.g. autistic disorder, Heller's syndrome). In part, the low incidence of these diseases in the training data provided little learning opportunity to the flexible models like RNN.

Deep learning models require high volumes of data because of the exponential number of feature combinations that must be assessed for the model to learn. The demand for high volume can be reduced by exploiting medical ontologies to encode hierarchical clinical constructs and relationships among medical concepts, effectively reducing the search space without loss of information. Fortunately, there are many well-organized ontologies in healthcare such as the International Classification of Diseases (ICD), Clinical Classifications Software (CCS) [133] or Systematized Nomenclature of Medicine-Clinical Terms (SNOMED-CT) [134]. Nodes (*i.e.* medical concepts) close to one another in medical

Figure 5.1: The illustration of GRAM. Leaf nodes (solid circles) represents a medical concept in the EHR, while the non-leaf nodes (dotted circles) represent more general concepts. The final representation $\mathbf{g}_i$ of the leaf concept c_i is computed by combining the basic embeddings $\mathbf{e}_i$ of c_i and $\mathbf{e}_g, \mathbf{e}_c$ and $\mathbf{e}_a$ of its ancestors c_g, c_c and c_a via an attention mechanism. The final representations form the embedding matrix $\mathbf{G}$ for all leaf concepts. After that, we use $\mathbf{G}$ to embed patient visit vector $\mathbf{x}_t$ to a visit representation $\mathbf{v}_t$, which is then fed to a neural network model to make the final prediction $\hat{\mathbf{y}}_t$.



ontologies are likely to be associated with similar patients, allowing us to transfer knowledge among them. Use of medical ontologies are likely to be helpful when data volume is insufficient to train deep learning models, and possibly even when data volume is sufficient as a means to improve model parsimony without loss of information and by learning more interpretable representations that are consistent with the ontology structure.

In this chapter, we propose GRAM, a method that infuses information from medical ontologies into deep learning models via neural attention. Considering the frequency of a medical concept in the EHR data and its ancestors in the ontology, GRAM optimizes the medical concept by adaptively combining its ancestors via attention mechanism (*i.e.* weighted sum of the representations of ancestors). The attention mechanism is trained in an end-to-end fashion with the neural network model that predicts the onset of disease(s). We also propose an effective initialization technique to better guide the representation learning process.

We compare predictive performance (i.e. accuracy, data needs, interpretability) of GRAM to various models including the recurrent neural network (RNN) in two sequential diagnoses prediction tasks and one heart failure (HF) prediction task. We demonstrate that GRAM is up

to 10% more accurate than the basic RNN for predicting diseases less observed in the training data. After discussing GRAM's scalability, we visualize the representations learned from various models, where GRAM provides more intuitive representations by grouping similar medical concepts close to one another. Finally, we show GRAM's attention mechanism can be interpreted to understand how it assigns the right amount of attention to the ancestors of each medical concept by considering the data availability and the ontology structure.

5.2 Methodology

We first define the notations describing EHR data and medical ontologies, followed by a description of GRAM (Section 5.2.2), the end-to-end training of the attention generation and predictive modeling (Section 5.2.3), and the efficient initialization scheme (Section 5.2.4).

5.2.1 Basic Notation

We denote the set of entire medical codes from the EHR as $c_1, c_2, \ldots, c_{|\mathcal{C}|} \in \mathcal{C}$ with the vocabulary size $|\mathcal{C}|$. The clinical record of each patient can be viewed as a sequence of visits $V_1, \ldots, V_T$ where each visit contains a subset of medical codes $V_t \subseteq \mathcal{C}$. V_t can be represented as a binary vector $\mathbf{x}_t \in \{0,1\}^{|\mathcal{C}|}$ where the i-th element is 1 only if V_t contains the code c_i. To avoid clutter, all algorithms will be presented for a single patient.

We assume that a given medical ontology $\mathcal{G}$ typically expresses the hierarchy of various medical concepts in the form of a *parent-child* relationship, where the medical codes $\mathcal{C}$ form the leaf nodes. Ontology $\mathcal{G}$ is represented as a directed acyclic graph (DAG) whose nodes form a set $\mathcal{D} = \mathcal{C} + \mathcal{C}'$. The set $\mathcal{C}' = \{c_{|\mathcal{C}|+1}, c_{|\mathcal{C}|+2}, \ldots,$
$c_{|\mathcal{C}|+|\mathcal{C}'|}\}$ consists of all non-leaf nodes (*i.e.* ancestors of the leaf nodes), where $|\mathcal{C}'|$ represents the number of all non-leaf nodes. We use *knowledge DAG* to refer to $\mathcal{G}$. A parent in the knowledge DAG $\mathcal{G}$ represents a related but more general concept over its children. Therefore, $\mathcal{G}$ provides a multi-resolution view of medical concepts with different degrees of specificity. While some ontologies are exclusively expressed as parent-child hierarchies

(e.g. ICD-9, CCS), others are not. For example, in some instances SNOMED-CT also links medical concepts to causal or treatment relationships, but a majority of the relationships in SNOMED-CT are still parent-child. Therefore, we focus on the parent-child relationships in this chapter.

5.2.2 Knowledge DAG and the Attention Mechanism

GRAM leverages the *parent-child* relationship of $\mathcal{G}$ to learn robust representations when data volume is constrained. GRAM balances the use of ontology information in relation to data volume in determining the level of specificity for a medical concept. When a medical concept is less frequent in the data, more weight is given to its ancestors as they can be learned more accurately and offer general (coarse-grained) information about their children. The process of resorting to the parent concepts can be automated via the attention mechanism and the end-to-end training as described in Figure 5.1.

In the knowledge DAG, each node c_i is assigned a basic embedding vector $\mathbf{e}_i \in \mathbb{R}^m$, where m represents the dimensionality. Then $\mathbf{e}_1, \ldots, \mathbf{e}_{|\mathcal{C}|}$ are the basic embeddings of the codes $c_1, \ldots, c_{|\mathcal{C}|}$ while $\mathbf{e}_{|\mathcal{C}|+1}, \ldots, \mathbf{e}_{|\mathcal{C}|+|\mathcal{C}'|}$ represent the basic embeddings of the internal nodes $c_{|\mathcal{C}|+1}, \ldots, c_{|\mathcal{C}|+|\mathcal{C}'|}$. The initialization of these basic embeddings is described in Section 5.2.4. We formulate a leaf node's final representation as a convex combination of the basic embeddings of itself and its ancestors:

$$\mathbf{g}_i = \sum_{j \in \mathcal{A}(i)} \alpha_{ij} \mathbf{e}_j, \qquad \sum_{j \in \mathcal{A}(i)} \alpha_{ij} = 1, \;\; \alpha_{ij} \geq 0 \;\text{ for } j \in \mathcal{A}(i), \tag{5.1}$$

where $\mathbf{g}_i \in \mathbb{R}^m$ denotes the final representation of the code c_i, $\mathcal{A}(i)$ the indices of the code c_i and c_i's ancestors, $\mathbf{e}_j$ the basic embedding of the code c_j and $\alpha_{ij} \in \mathbb{R}^+$ the attention weight on the embedding $\mathbf{e}_j$ when calculating $\mathbf{g}_i$. The attention weight α_{ij} in Eq. (5.1) is

calculated by the following Softmax function,

$$\alpha_{ij} = \frac{\exp(f(\mathbf{e}_i, \mathbf{e}_j))}{\sum_{k \in \mathcal{A}(i)} \exp(f(\mathbf{e}_i, \mathbf{e}_k))} \tag{5.2}$$

$f(\mathbf{e}_i, \mathbf{e}_j)$ is a scalar value representing the compatibility between the basic embeddings of $\mathbf{e}_i$ and $\mathbf{e}_k$. We compute $f(\mathbf{e}_i, \mathbf{e}_j)$ via the following feed-forward network with a single hidden layer (MLP),

$$f(\mathbf{e}_i, \mathbf{e}_j) = \mathbf{u}_a^\top \tanh(\mathbf{W}_a \begin{bmatrix} \mathbf{e}_i \\ \mathbf{e}_j \end{bmatrix} + \mathbf{b}_a) \tag{5.3}$$

where $\mathbf{W}_a \in \mathbb{R}^{l \times 2m}$ is the weight matrix for the concatenation of $\mathbf{e}_i$ and $\mathbf{e}_j$, $\mathbf{b} \in \mathbb{R}^l$ the bias vector, and $\mathbf{u}_a \in \mathbb{R}^l$ the weight vector for generating the scalar value. The constant l represents the dimension size of the hidden layer of $f(\cdot, \cdot)$. We concatenate $\mathbf{e}_i$ and $\mathbf{e}_j$ in the child-ancestor order. Note that the compatibility function f is an MLP, because MLP is well known to be a sufficient approximator for an arbitrary function, and we empirically found that our formulation performed better in our use cases than alternatives such as inner product and Bahdanau et al.'s [3].

Remarks: The example in Figure 5.1 is derived based on a single path from c_i to c_a. However, the same mechanism can be applicable to multiple paths as well. For example, code c_k has two paths to the root c_a, containing five ancestors in total. Another scenario is where the EHR data contain both leaf codes and some ancestor codes. We can move those ancestors present in EHR data from the set $\mathcal{C}'$ to $\mathcal{C}$ and apply the same process as Eq. (5.1) to obtain the final representations for them.

5.2.3 End-to-End Training with a Predictive Model

We train the attention mechanism together with a predictive model such that the attention mechanism improves the predictive performance. By concatenating final representation $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_{|\mathcal{C}|}$ of all medical codes, we have the embedding matrix $\mathbf{G} \in \mathcal{R}^{m \times |\mathcal{C}|}$ where $\mathbf{g}_i$

is the i-th column of $\mathbf{G}$. As shown in the right side of Figure 5.1, we can convert a visit V_t to a visit representation $\mathbf{v}_t$ by multiplying the embedding matrix $\mathbf{G}$ with a multi-hot (*i.e.* multi-label binary) vector $\mathbf{x}_t$ indicating the clinical events in the visit V_t, followed by a nonlinear activation via $\tanh$. Finally the visit representation $\mathbf{v}_t$ will be used as an input to the neural network model for predicting the target label $\mathbf{y}_t$. In this chapter, we use RNN as the choice of the NN model to perform sequential diagnoses prediction [10, 13]. That is, we are interested in predicting the disease codes of the next visit V_{t+1} given the visit records up to the current timestep $V_1, V_2, \ldots, V_t$, which can be expressed as follows,

$$\begin{aligned}
\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t &= \tanh(\mathbf{G}[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t]), \\
\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_t &= \text{RNN}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t, \theta_r), \\
\widehat{\mathbf{y}}_t = \widehat{\mathbf{x}}_{t+1} &= \text{Softmax}(\mathbf{W}\mathbf{h}_t + \mathbf{b}),
\end{aligned} \tag{5.4}$$

where $\mathbf{x}_t \in \mathbb{R}^{|\mathcal{C}|}$ denotes the multi-hot vector for the t-th visit; $\mathbf{v}_t \in \mathbb{R}^m$ the t-th visit representation; $\mathbf{h}_t \in \mathbb{R}^r$ the RNN's hidden layer at the t-th time step (*i.e.* t-th visit); θ_r RNN's parameters; $\mathbf{W} \in \mathbb{R}^{|\mathcal{C}| \times r}$ and $\mathbf{b} \in \mathbb{R}^{|\mathcal{C}|}$ the weight matrices and the bias vector of the final Softmax function (r denotes the dimension size of the hidden layer). Note that we use Softmax instead of dimension-wise sigmoid for predicting multiple disease codes in the next visit V_{t+1} because it showed better performance. Here we use "RNN" to denote any recurrent neural network variants that can cope with the vanishing gradient problem [118], such as LSTM [58], GRU [119], and IRNN [120]. The prediction loss for all time steps is calculated using the binary cross entropy as follows,

$$\mathcal{L}(\mathbf{x}_1, \mathbf{x}_2 \ldots, \mathbf{x}_T) = -\frac{1}{T-1} \sum_{t=1}^{T-1} \Big({\mathbf{y}_t}^\top \log(\widehat{\mathbf{y}}_t) + (\mathbf{1} - \mathbf{y}_t)^\top \log(\mathbf{1} - \widehat{\mathbf{y}}_t) \Big) \tag{5.5}$$

where we sum the cross entropy errors from all timestamps of $\widehat{\mathbf{y}}_t$, T denotes the number of timestamps of the visit sequence. Note that the above loss is defined for a single patient. In actual implementation, we will take the average of the individual loss for multiple patients.

Algorithm 1 GRAM Optimization

Randomly initialize basic embedding matrix $\mathbf{E}$, attention parameters $\mathbf{u}_a, \mathbf{W}_a, \mathbf{b}_a$, RNN parameter θ_r, softmax parameters $\mathbf{W}, \mathbf{b}$.
repeat
  Update $\mathbf{E}$ with GloVe objective function (see Section 5.2.4)
until convergence
repeat
  $\mathbf{X} \leftarrow$ random patient from dataset
  for visit V_t **in** $\mathbf{X}$ **do**
    for code c_i **in** V_t **do**
      Refer $\mathcal{G}$ to find c_i's ancestors C'
      for code c_j **in** C' **do**
        Calculate attention weight α_{ij} using Eq. (5.2).
      end for
      Obtain final representation $\mathbf{g}_i$ using Eq. (5.1).
    end for
    $\mathbf{v}_t \leftarrow \tanh(\sum_{i:c_i \in V_t} \mathbf{g}_i)$
    Make prediction $\widehat{\mathbf{y}}_t$ using Eq. (5.4)
  end for
  Calculate prediction loss $\mathcal{L}$ using Eq .(5.5)
  Update parameters according to the gradient of $\mathcal{L}$
until convergence

Algorithm 1 describes the overall GRAM training procedure assuming that we are performing the sequential diagnoses prediction task using an RNN. Note that Algorithm 1 describes stochastic gradient update to avoid clutter, but it can be easily extended to other gradient based optimization such as mini-batch gradient update.

5.2.4 Initializing Basic Embeddings

The attention generation mechanism in Section 5.2.2 requires basic embeddings $\mathbf{e}_i$ of each node in the knowledge DAG. The basic embeddings of ancestors, however, are not usually observed in the data. To properly initialize them, we use co-occurrence information to learn the basic embeddings of medical codes and their ancestors. Co-occurrence has proven to be an important source of information when learning representations of words or medical concepts [11, 12, 55]. To train the basic embeddings, we employ GloVe [77], which uses the global co-occurrence matrix of words to learn their representations. In our case, the co-

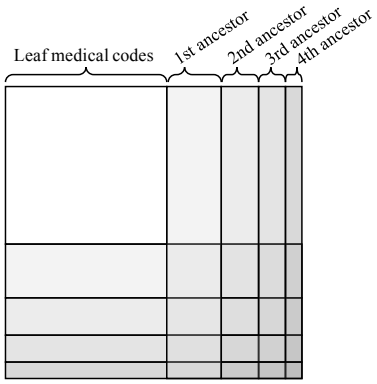


Figure 5.2: Creating the co-occurrence matrix together with the ancestors. The n-th ancestors are the group of nodes that are n hops away from any leaf node in $\mathcal{G}$. Here we exclude the root node, which will be just a single row (column).

occurrence matrix of the codes and the ancestors was generated by counting co-occurrences within each visit V_t, where we then augment each visit with the ancestors of the codes in the visit. We describe the initialization algorithm with an example knowledge DAG of Figure 5.1. Given a visit V_t,

$$V_t = \{c_d, c_i, c_k\}$$

we augment the leaf codes with their ancestors to obtain the augmented visit V_t',

$$V_t' = \{c_d, \underline{c_b}, \underline{c_a}, c_i, \underline{c_g}, \underline{c_c}, \underline{c_a}, c_k, \underline{c_j}, \underline{c_f}, \underline{c_c}, \underline{c_b}, \underline{c_a}\}$$

where the augmented ancestors are underlined. Note that a single ancestor can appear multiple times in V_t'. In fact, the higher the ancestor is in the knowledge DAG, the more times it is likely to appear in V_t'. Co-occurrence of two codes in V_t' are counted as follows,

$$\textit{co-occurrence}(c_i, c_j, V_t') = \textit{count}(c_i, V_t') \times \textit{count}(c_j, V_t')$$

where $\textit{count}(c_i, V_t')$ is the number of times the code c_i appears in the augmented visit V_t'. For example, the co-occurrence between the leaf code c_i and the root c_a is 3. However, the

co-occurrence between the ancestor c_c and the root c_a is 6. Therefore our algorithm will make the higher ancestor codes more likely to be involved in all medical events (*i.e.* visits), which is natural in healthcare applications as those general concepts are often reliable. We repeat this calculation for all pairs of codes in all augmented visits of all patients to obtain the co-occurrence matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{D}|\times|\mathcal{D}|}$ depicted by Figure 5.2. For training the embedding vectors $\mathbf{e}_i$'s using $\mathbf{M}$, we minimize the following loss function as described in [77].

$$J = \sum_{i,j=1}^{|\mathcal{D}|} f(\mathbf{M}_{ij})(\mathbf{e}_i^\top \mathbf{e}_j + b_i + b_j - \log \mathbf{M}_{ij})^2$$

$$\text{where } f(x) = \begin{cases} (x < x_{max})^\alpha & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases}$$

and the hyperparameters x_{max} and α are respectively set to 100 and 0.75 as the original paper [77]. Note that, after the initialization, the basic embeddings $\mathbf{e}_i$'s of both leaf nodes (*i.e.* medical codes) and non-leaf nodes (*i.e.* ancestors) are fine-tuned during model training via backpropagation.

5.3 Experiments

We conducted three experiments to determine if `GRAM` offered superior prediction performance when facing data insufficiency. We first describe the experimental setup followed by results comparing predictive performance of `GRAM` with various baseline models. Then we present `GRAM`'s scalability results. Finally, we qualitatively show the intuitive interpretation of `GRAM`. The source code of `GRAM` is publicly available at `https://github.com/mp2893/gram`.

5.3.1 Experiment Setup

Prediction tasks and source of data: We conducted two sequential diagnoses prediction (SDP) tasks using two different datasets. The overall aim of the experiments was to use

Table 5.1: Basic statistics of Sutter PAMF, MIMIC-III and Sutter heart failure (HF) cohort.

Dataset	Sutter PAMF	MIMIC-III	Sutter HF cohort
# of patients	258,555†	7,499†	30,727† (3,408 cases)
# of visits	13,920,759	19,911	572,551
Avg. # of visits per patient	53.8	2.66	38.38
# of unique ICD9 codes	10,437	4,893	5,689
Avg. # of codes per visit	1.98	13.1	2.06
Max # of codes per visit	54	39	29

† For all datasets, we chose patients who made at least two visits.

all information prior to a next visit to predict all diagnosis codes that would be in that next visit. The first dataset was from the Sutter Palo Alto Medical Foundation (PAMF), which consisted of 10-years longitudinal medical records of 258K primary care patients between 50 to 89 years of age. This will determine `GRAM`'s performance for general adult population with many hospital visits. The second dataset was MIMIC-III [135, 136], which is a publicly available dataset consisting of medical records of 7.5K intensive care unit (ICU) patients over 11 years. This will determine `GRAM`'s performance for high-risk patients with very few hospital visits. We utilized all the patients with at least 2 visits. We prepared the true labels $\mathbf{y}_t$ by grouping the ICD9 codes into 283 groups using CCS single-level diagnosis grouper[1]. This is to improve the training speed and predictive performance for easier analysis, while preserving sufficient granularity for each diagnosis. Each diagnosis code's varying frequency in the training data can be viewed as different degrees of data insufficiency. Model performance was assessed by *Accuracy@k* for each of CCS single-level diagnosis codes such that, given a visit V_t, we get 1 if the target diagnosis is in the top k guesses and 0 otherwise.

We also conducted a heart failure (HF) prediction task, which is a binary prediction task for predicting a future HF onset where the prediction is made only once at the last visit $\mathbf{x}_T$. The key difference between sequential diagnoses prediction and HF prediction is that the prediction target for the former can already occur in patient's prior visits while the prediction target for the latter is a new diagnosis of HF that has not appeared before. HF prediction was conducted on Sutter heart failure (HF) cohort, which is a subset of Sutter PAMF data

[1] https://www.hcup-us.ahrq.gov/toolssoftware/ccs/AppendixASingleDX.txt

for a heart failure onset prediction study with 3.4K HF cases chosen by a set of criteria described in [137, 138] and 27K matching controls chosen by a set of criteria described in [9]. This will determine GRAM's performance for a different prediction task where we predict the onset of one specific condition. We randomly downsampled the training data to create different degrees of data insufficiency. We used area under the ROC curve (AUC) to measure the performance.

A summary of the datasets are provided in Table 5.1.We used CCS multi-level diagnoses hierarchy[2] as our knowledge DAG $\mathcal{G}$. We also tested the ICD9 code hierarchy[3], but the performance was similar to using CCS multi-level hierarchy. For all three tasks, we randomly divide the dataset into the training, validation and test set by .75:.10:.15 ratio, and use the validation set to tune the hyper-parameters. Further details regarding the hyper-parameter tuning are provided below. The test set performance is reported in the paper.

Implementation details: We implemented GRAM with Theano 0.8.2 [139]. For training models, we used Adadelta [100] with a mini-batch of 100 patients, on a machine equipped with Intel Xeon E5-2640, 256GB RAM, four Nvidia Titan X's and CUDA 7.5.

Models for comparison are the following. The first two GRAM+ and GRAM are the proposed methods and the rest are baselines. Hyper-parameter tuning is configured so that the number of parameters for the baselines would be comparable to GRAM's. Further details are provided below.

- **GRAM:** Input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$ is first transformed by the embedding matrix $\mathbf{G}$, then fed to the GRU with a single hidden layer, which in turn makes the prediction, as described by Eq. (5.4). The basic embeddings $\mathbf{e}_i$'s are randomly initialized.

- **GRAM+:** We use the same setup as **GRAM**, but the basic embeddings $\mathbf{e}_i$'s are initialized according to Section 5.2.4.

- **RandomDAG:** We use the same setup as **GRAM**, but each leaf concept has five randomly

[2] https://www.hcup-us.ahrq.gov/toolssoftware/ccs/AppendixCMultiDX.txt
[3] http://www.icd9data.com/2015/Volume1/default.htm

assigned ancestors from the CCS multi-level hierarchy to test the effect of correct domain knowledge.

- **RNN**: Input $\mathbf{x}_t$ is transformed by an embedding matrix $\mathbf{W}_{emb} \in \mathbb{R}^{k \times |\mathcal{C}|}$, then fed to the GRU with a single hidden layer. The embedding size k is a hyper-parameter. $\mathbf{W}_{emb}$ is randomly initialized and trained together with the GRU.

- **RNN+:** We use the **RNN** model with the same setup as before, but we initialize the embedding matrix $\mathbf{W}_{emb}$ with GloVe vectors trained only with the co-occurrence of leaf concepts. This is to compare `GRAM` with a similar weight initialization technique.

- **SimpleRollUp:** We use the **RNN** model with the same setup as before. But for input $\mathbf{x}_t$, we replace all diagnosis codes with their direct parent codes in the CCS multi-level hierarchy, giving us 578, 526 and 517 input codes respectively for Sutter data, MIMIC-III and Sutter HF cohort. This is to compare the performance of `GRAM` with a common grouping technique.

- **RollUpRare:** We use the **RNN** model with the same setup as before, but we replace any diagnosis code whose frequency is less than a certain threshold in the dataset with its direct parent. We set the threshold to 100 for Sutter data and Sutter HF cohort, and 10 for MIMIC-III, giving us 4,408, 935 and 1,538 input codes respectively for Sutter data, MIMIC-III and Sutter HF cohort. This is an intuitive way of dealing with infrequent medical codes.

Hyper-parameter Tuning: We define five hyper-parameters for `GRAM`:

- dimensionality m of the basic embedding $\mathbf{e}_i$: [100, 200, 300, 400, 500]

- dimensionality r of the RNN hidden layer $\mathbf{h}_t$ from Eq. (5.4): [100, 200, 300, 400, 500]

- dimensionality l of $\mathbf{W}_a$ and $\mathbf{b}_a$ from Eq. (5.3): [100, 200, 300, 400, 500]

- L_2 regularization coefficient for all weights except RNN weights: [0.1, 0.01, 0.001, 0.0001]
- dropout rate for the dropout on the RNN hidden layer: [0.0, 0.2, 0.4, 0.6, 0.8]

We performed 100 iterations of the random search by using the above ranges for each of the three prediction experiments. In order to fairly compare the model performances, we matched the number of model parameters to be similar for all baseline methods. To facilitate reproducibility, final hyper-parameter settings we used for all models for each prediction experiments are described at the source code repository, `https://github.com/mp2893/gram`, along with the detailed steps we used to tune the hyper-parameters.

5.3.2 Prediction performance

Tables 5.2a and 5.2b show the sequential diagnoses prediction performance on Sutter data and MIMIC-III. Both tables show that GRAM+ outperforms other models when predicting labels with significant data insufficiency (*i.e.* less observed in the training data).The performance gain is greater for MIMIC-III, where GRAM+ outperforms the basic RNN by 10% in the 20th-40th percentile range. This seems to come from the fact that MIMIC patients on average have significantly shorter visit history than Sutter patients, with much more codes received per visit. Such short sequences make it difficult for the RNN to learn and predict diagnoses sequence. The performance difference between GRAM+ and GRAM suggests that our proposed initialization scheme of the basic embeddings $\mathbf{e}_i$ is important for sequential diagnosis prediction.

Table 5.2c shows the HF prediction performance on Sutter HF cohort. GRAM and GRAM+ consistently outperforms other baselines (except RNN+) by 3$\sim$4% AUC, and RNN+ by maximum 1.8% AUC. These differences are quite significant given that the AUC is already in the mid-80s, a high value for HF prediction, cf. [9]. Note that, for GRAM+ and RNN+, we used the downsampled training data to initialize the basic embeddings $\mathbf{e}_i$'s and the embedding matrix $\mathbf{W}_{emb}$ with GloVe, respectively. The result shows that the initialization

scheme of the basic embeddings in GRAM+ gives limited improvement over GRAM. This stems from the different natures of the two prediction tasks. While the goal of HF prediction is to predict a binary label for the entire visit sequence, the goal of sequential diagnosis prediction is to predict the co-occurring diagnosis codes at every visit. Therefore the co-occurrence information infused by the initialized embedding scheme is more beneficial to sequential diagnosis prediction. Additionally, this benefit is associated with the natures of the two prediction tasks than the datasets used for the prediction tasks. Because the initialized embedding shows different degrees of improvement as shown by Tables 5.2a and 5.2c, when Sutter HF cohort is a subset of Sutter PAMF, thus having similar characteristics.

Overall, GRAM showed superior predictive performance under data insufficiency in three different experiments, demonstrating its general applicability in clinical predictive modeling.

5.3.3 Scalability

We briefly discuss the scalability of GRAM by comparing its training time to RNN's. Table 5.3 shows the number of seconds taken for the two models to train for a single epoch for each predictive modeling task. GRAM+ and RNN+ showed the similar behavior as GRAM and RNN. GRAM takes approximately 50% more time to train for a single epoch for all prediction tasks. This stems from calculating attention weights and the final representations $\mathbf{g}_i$ for all medical codes. GRAM also generally takes about 50% more epochs to reach to the model with the lowest validation loss. This is due to optimizing an extra MLP model that generates the attention weights. Overall, use of GRAM adds a manageable amount of overhead in training time to the plain RNN.

5.3.4 Qualitative evaluation of interpretable representations

To qualitatively assess the interpretability, we generates the t-SNE plots [18] using the final representations $\mathbf{g}_i$ of 2,000 randomly chosen diseases learned by GRAM+ for sequential

diagnoses prediction on Sutter data[4] (Figure 5.3a). The color of the dots represents the highest disease categories and the text annotations represent the detailed disease categories in CCS multi-level hierarchy. For comparison, we also show the t-SNE plots on the strongest results from GRAM (Figure 5.3b), RNN+ (Figure 5.3c), RNN (Figure 5.3d) and RandomDAG (Figure 5.3e). GloVe (Figure 5.3f) and Skip-gram (Figure 5.3g) were trained on the Sutter data, where a single visit V_t was used as the context window to calculate the co-occurrence of codes.

Figures 5.3c and 5.3f confirm that interpretable representations cannot simply be learned only by co-occurrence or supervised prediction without medical knowledge. GRAM+ and GRAM learn interpretable disease representations that are significantly more consistent with the given knowledge DAG $\mathcal{G}$. Based on the prediction performance shown by Table 5.2, and the fact that the representations $\mathbf{g}_i$'s are the final product of GRAM, we can infer that such medically meaningful representations are necessary for predictive models to cope with data insufficiency and make more accurate predictions. Figure 5.3b shows that the quality of the final representations $\mathbf{g}_i$ of GRAM is quite similar to GRAM+. Compared to other baselines, GRAM demonstrates significantly more structured representations that align well with the given knowledge DAG. It is interesting that Skip-gram shows the most structured representation among all baselines. We used GloVe to initialize the basic embeddings $\mathbf{e}_i$ in this chapter because it uses global co-occurrence information and its training time is fast as it only depends on the total number of unique concepts $|\mathcal{C}|$. Skip-gram's training time, on the other hand, depends on both the number of patients and the number of visits each patient made, which makes the algorithm generally slower than GloVe. An interactive visualization tool can be accessed at `http://www.sunlab.org/research/gram-graph-based-attention-model/`.

[4]The scatterplots of models trained for sequential diagnoses prediction on MIMIC-III and HF prediction for Sutter HF cohort were similar but less structured due to smaller data size.

5.3.5 Analysis of the attention behavior

Next we show that GRAM's attention can be explained intuitively based on the data availability and knowledge DAG's structure when performing a prediction task. Using Eq. (5.1), we can calculate the attention weights of individual disease. Figure 5.4 shows the attention behaviors of four representative diseases when performing HF prediction on Sutter HF cohort.

Other pneumothorax (ICD9 512.89) in Figure 5.4a is rarely observed in the data and has only five siblings. In this case, most information is derived from the highest ancestor. *Temporomandibular joint disorders & articular disc disorder* (ICD9 524.63) in Figure 5.4b is rarely observed but has 139 siblings. In this case, its parent receives a stronger attention because it aggregates sufficient samples from all of its children to learn a more accurate representation. Note that the disease itself also receives a stronger attention to facilitate easier distinction from its large number of siblings.

Unspecified essential hypertension (ICD9 401.9) in Figure 5.4c is very frequently observed but has only two siblings. In this case, GRAM assigns a very strong attention to the leaf, which is logical because the more you observe a disease, the stronger your confidence becomes. *Need for prophylactic vaccination and inoculation against influenza* (ICD9 V04.81) in Figure 5.4d is quite frequently observed and also has 103 siblings. The attention behavior in this case is quite similar to the case with fewer siblings (Figure 5.4b) with a slight attention shift towards the leaf concept as more observations lead to higher confidence.

5.4 Related Work

The attention mechanism is a general framework for neural network learning [3], and has been since used in many areas such as speech recognition [140], computer vision [121, 124] and healthcare [13]. However, no one has designed attention model based on knowledge

ontology, which is the focus of this chapter.

There are related works in learning the representations of graphs. Several studies focused on learning the representations of graph vertices by using the neighbor information. DeepWalk [141] and node2vec [142] use random walk while LINE [143] uses breadth-first search to find the neighbors of a vertex and learn its representation based on the neighbor information. Graph convolutional approaches [144, 145] also focus on learning the vertex representations to mainly perform vertex classification. All those works focus on solving the graph data problems whereas `GRAM` focuses on solving clinical predictive modeling problems using the knowledge DAG as supplementary information.

Several researchers tried to model the knowledge DAG such as WordNet [146] or Freebase [147] where two entities are connected with various types of relation, forming a set of triples. They aim to project entities and relations [148, 149, 150, 151] to the latent space based on the triples or additional information such as hierarchy of entities [152]. These works demonstrated tasks such as link prediction, triple classification or entity classification using the learned representations. More recently, [153] learned the representations of words and Wikipedia categories by utilizing the hierarchy of Wikipedia categories. `GRAM` is fundamentally different from the above studies in that it aims to design intuitive attention mechanism on the knowledge DAG as a knowledge prior to cope with data insufficiency and learn medically interpretable representations to make accurate predictions.

A classical approach for incorporating side information in the predictive models is to use graph Laplacian regularization [154, 53]. However, using this approach is not straightforward as it relies on the appropriate definition of distance on graphs which is often unavailable.

5.5 Conclusion

Data insufficiency, either due to less common diseases or small datasets, is one of the key hurdles in healthcare analytics, especially when we apply deep neural networks models.

To overcome this challenge, we leveraged the knowledge DAG, which provides a multi-resolution view of medical concepts. We proposed GRAM, a graph-based attention model using both a knowledge DAG and EHR to learn an accurate and interpretable representations for medical concepts. GRAM chooses a weighted average of ancestors of a medical concept and train the entire process with a predictive model in an end-to-end fashion. We conducted three predictive modeling experiments on real EHR datasets and showed significant improvement in the prediction performance, especially on low-frequency diseases and small datasets. Analysis of the attention behavior provided intuitive insight of GRAM. Although GRAM showed good performance, there is room for improving the way we incorporate knowledge DAG into neural networks. For future work, we plan to devise a method to systematically leverage knowledge DAG in addition to using attention-weighted embeddings.

Model	0-20	20-40	40-60	60-80	80-100
GRAM+	**0.0150**	**0.3242**	0.4325	0.4238	0.4903
GRAM	0.0042	0.2987	0.4224	0.4193	0.4895
RandomDAG	0.0050	0.2700	0.4010	0.4059	0.4853
RNN+	0.0069	0.2742	0.4140	0.4212	**0.4959**
RNN	0.0080	0.2691	0.4134	0.4227	0.4951
SimpleRollUp	0.0085	0.3078	**0.4369**	**0.4330**	0.4924
RollUpRare	0.0062	0.2768	0.4176	0.4226	0.4956

(a) *Accuracy@5* of sequential diagnoses prediction on Sutter data

Model	0-20	20-40	40-60	60-80	80-100
GRAM+	**0.0672**	**0.1787**	**0.2644**	0.2490	0.6267
GRAM	0.0556	0.1016	0.1935	0.2296	0.6363
RandomDAG	0.0329	0.0708	0.1346	0.1512	0.4494
RNN+	0.0454	0.0843	0.2080	0.2494	0.6239
RNN	0.0454	0.0731	0.1804	0.2371	0.6243
SimpleRollUp	0.0578	0.1328	0.2455	**0.2667**	**0.6387**
RollUpRare	0.0454	0.0653	0.1843	0.2364	0.6277

(b) *Accuracy@20* of sequential diagnoses prediction on MIMIC-III

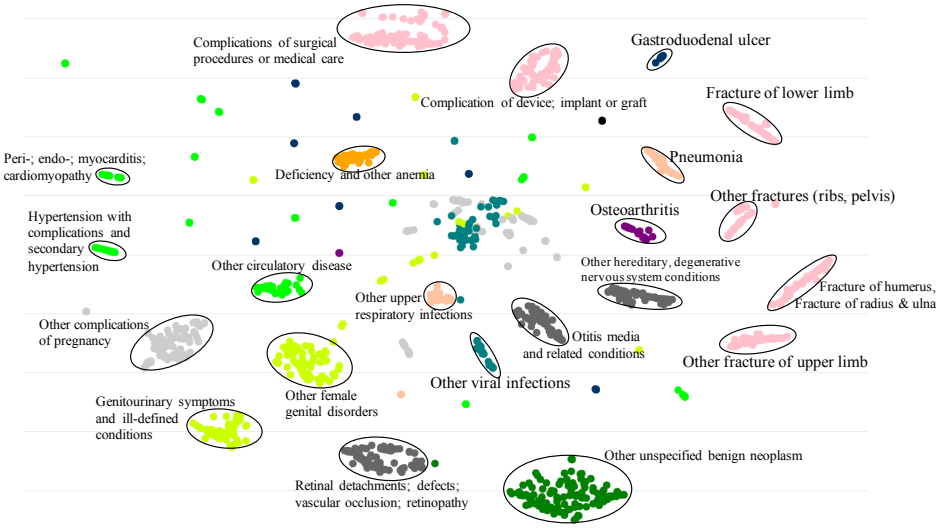
Model	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
GRAM+	0.7970	**0.8223**	0.8307	**0.8332**	**0.8389**	**0.8404**	**0.8452**	**0.8456**	**0.8447**	**0.8448**
GRAM	**0.7981**	0.8217	**0.8340**	**0.8332**	0.8372	0.8377	0.8440	0.8431	0.8430	0.8447
RandomDAG	0.7644	0.7882	0.7986	0.8070	0.8143	0.8185	0.8274	0.8312	0.8254	0.8226
RNN+	0.7930	0.8117	0.8162	0.8215	0.8261	0.8333	0.8343	0.8353	0.8345	0.8335
RNN	0.7811	0.7942	0.8066	0.8111	0.8156	0.8207	0.8258	0.8278	0.8297	0.8314
SimpleRollUp	0.7799	0.8022	0.8108	0.8133	0.8177	0.8207	0.8223	0.8272	0.8269	0.8258
RollUpRare	0.7830	0.8067	0.8064	0.8119	0.8211	0.8202	0.8262	0.8296	0.8307	0.8291

(c) AUC of HF onset prediction on Sutter HF cohort

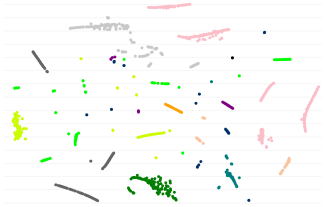
Table 5.2: Performance of three prediction tasks. The x-axis of (a) and (b) represents the labels grouped by the percentile of their frequencies in the training data in non-decreasing order. 0-20 are the most rare diagnosese while 80-100 are the most common ones. (b) uses *Accuracy@20* because MIMIC-III has a large average number of codes per visit (see Table 5.1). For (c), we vary the size of the training data to train the models.

Table 5.3: Scalability result in per epoch training time in second (the number of epochs needed). SDP stands for Sequential Diagnoses Prediction

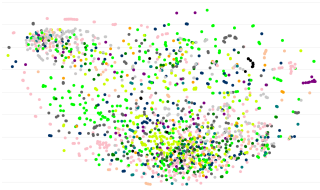
Model	SDP (Sutter data)	SDP (MIMIC-III)	HF prediction (Sutter HF cohort)
GRAM	525s (39 epochs)	2s (11 epochs)	12s (7 epochs)
RNN	352s (24 epochs)	1s (6 epochs)	8s (5 epochs)



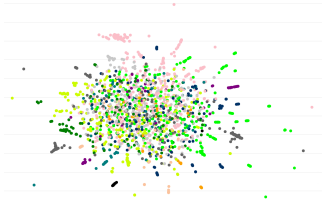
(a) Scatterplot of the final representations $\mathbf{g}_i$'s of GRAM+



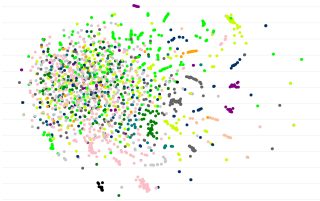
(b) Scatterplot of the final representations $\mathbf{g}_i$'s of GRAM



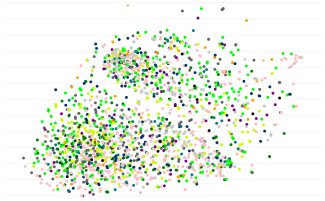
(c) Scatterplot of the trained embedding matrix $\mathbf{W}_{emb}$ of RNN+



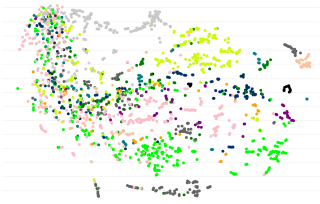
(d) Scatterplot of the trained embedding matrix $\mathbf{W}_{emb}$ of RNN



(e) Scatterplot of the final representations $\mathbf{g}_i$'s of RandomDAG

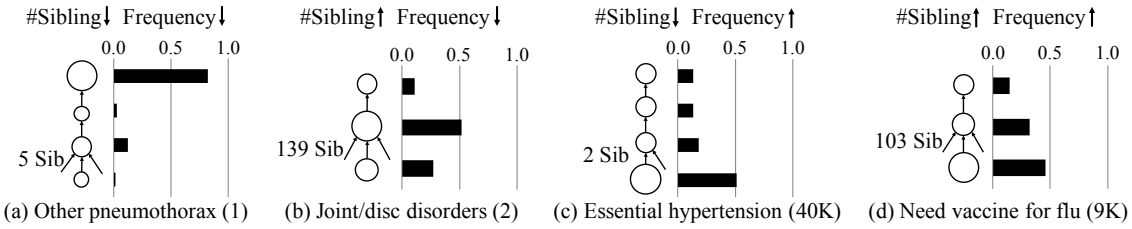


(f) Scatterplot of the disease representations trained by GloVe



(g) Scatterplot of the basic embeddings $\mathbf{e}_i$'s trained by Skip-gram

Figure 5.3: t-SNE scatterplots of medical concepts trained by GRAM+, GRAM, RNN+, RNN, RandomDAG, GloVe and Skip-gram. The color of the dots represents the highest disease categories and the text annotations represent the detailed disease categories in CCS multi-level hierarchy. It is clear that GRAM+ and GRAM exhibit interpretable embedding that are well aligned with the medical ontology.



(a) Other pneumothorax (1) (b) Joint/disc disorders (2) (c) Essential hypertension (40K) (d) Need vaccine for flu (9K)

Figure 5.4: GRAM's attention behavior during HF prediction for four representative diseases (each column). In each figure, the leaf node represents the disease and upper nodes are its ancestors. The size of the node shows the amount of attention it receives, which is also shown by the bar charts. The number in the parenthesis next to the disease is its frequency in the training data. We exclude the root of the knowledge DAG $\mathcal{G}$ from all figures as it did not play a significant role.

CHAPTER 6

MiME: MULTILEVEL MEDICAL EMBEDDING OF ELECTRONIC HEALTH RECORDS FOR PREDICTIVE HEALTHCARE

Medical concept embedding of electronic health record (EHR) data facilitates downstream analytical tasks such as predictive modeling and computational phenotyping, among other applications. Current approaches to embedding do not fully capture the multilevel structure and heterogeneous relations among EHR medical codes that collectively represent the patient's reasons for visits and provider decisions. As a consequence, machine learning models may derive solutions that fail to capture clinically meaningful and actionable qualities. We propose MiME, a general framework to transform the hierarchically structured EHR data into a multilevel embedded format. MiME also jointly trains with multiple auxiliary tasks to inject general purpose prior knowledge into the learning process to achieve robust performance, especially when data volume, density, or diversity are usually deemed insufficient. We evaluated MiME's impact on model performance using three clinical prediction tasks: heart failure prediction, sequential diagnosis prediction, and medication prediction. MiME outperformed the strongest baseline by an absolute increase of 1.9% PR-AUC in heart failure prediction. We also conducted a detailed analysis of the experiment results together with visualization to confirm that capturing the multilevel structure of EHR data indeed improves a model's predictive performance. Lastly, using a much smaller dataset with short visit sequences, we demonstrated that MiME can significantly outperform baselines via joint training with auxiliary tasks, showing 2.5% higher PR-AUC than the best baseline.

6.1 Introduction

Machine learning is increasingly applied to high-dimensional and heterogeneous Electronic health record (EHR) data to devise improved means of prevention (*e.g.* early detection

of disease), population health management, and segmentation (*e.g.* identify treatment phenotypes), among other applications. EHR data volume are doubling approximately every two years with increasing diversity. Patient encounter records are time-stamped and populated by diverse codes (e.g., diagnoses, procedures, medications), which are naturally organized in a multilevel structure. For example, we can imagine a patient receiving multiple diagnoses in an encounter where one of them is *fever*. Then *fever* can give rise to a medication order for *acetaminophen* and a procedure order for *IV fluid*. Such events can happen in all diagnoses, which build up a single encounter. Then a sequence of encounters over time builds up a comprehensive summary of the patient health status and clinical services performed by the provider. Effective medical concept embedding can be a key enabler to decipher these important information for facilitating downstream analytical tasks, such as diagnostic classification [57, 13, 14, 155], disease detection [9, 10, 115], risk prediction [156, 157], and patient subtyping [158, 159].

To that extent, research efforts are often focused on properly deriving medical concept embeddings with advanced modeling techniques. Recently, deep learning models have shown state-of-the-art performance in deriving effective embeddings from high-dimensional raw data for various types of tasks such as image classification [2, 160, 161, 162], machine translation [119, 3], audio processing [5, 163], image captioning [4, 164] and visual question answering [165, 166]. Therefore deep learning approaches have become the preferred technique for processing high-dimensional EHR data to effectively learn medical concept embeddings. Attempts have been made either by summarizing high-dimensional medical codes into compressed vectors or using latent layers of deep models to represent medical concepts [**Tran2015ec**, 167, 12, 14, 9, 10, 13, 155, 168].

Although these approaches have successfully demonstrated the modeling capacity of deep learning for medical concept embedding, there are still gaps between existing approaches and the real-world settings. 1) Most of these approaches do not leverage the explicit multilevel structure in the EHR data, but they rather flatten EHR data as a set of

codes, which destroys the heterogeneous relations between codes in different levels of EHR data hierarchy. Borrowing from the above example, the diagnosis *fever* has a direct relationship with the medication *acetaminophen* and the procedure *IV fluid*. But *acetaminophen* and *IV fluid* are also correlated since they both occur due to the same diagnosis. As these relations encode patient's reasons for the visit and provider decisions, flattening the data structure would miss fine-grained details and reduce the effectiveness of the learned embeddings. 2) Second, in real clinical practice, all medical events (*e.g.* encounter, diagnosis, prescription) are either strongly or implicitly related to diverse prediction tasks researchers and clinicians are interested in. And effectively leveraging this rich information can lead to better model performance in unfavorable situations such as when there are insufficient data. However, many existing approaches either learn embeddings in a purely unsupervised fashion or learn from a single prediction task, both of which have their limitations such as producing too generic embeddings, or not being able to robustly handle insufficient data. 3) Due to the aforementioned gaps, the learned embeddings cannot often robustly handle records with complex multilevel relations, or small EHR data with short longitudinal records often seen in new EHR systems or critical care settings, both of which can be seen in sections 6.3.6 and 6.3.7.

To address these challenges, we propose `MiME`:**M**ult**i**level **M**edical **E**mbedding, to hierarchically transform the multilevel structure and heterogeneous relations (e.g., parent-child, siblings) among medical codes of EHR data into multilevel embeddings. Augmenting `MiME` with multiple auxiliary tasks allows us to inject prior knowledge shared across multiple prediction tasks into the embedding learning process to achieve robust performance under insufficient data settings. The contribution of this work is summarized as follows.

- We propose `MiME`, a general framework for accurately learning the multilevel (e.g., patient-level, visit-level, and diagnosis-level) structure of EHR data. Modeling the hierarchical relations among medical codes enables us to accurately capture the distinguishing patterns of different patient states.

- Against three benchmark tasks (heart failure (HF) prediction, sequential disease prediction, medication prediction) and a diverse set of baseline models, MiME shows good performance across all tasks in all metrics, especially demonstrating the best PR-AUC for HF prediction, outperforming the strongest baseline by 1.9%.
- We conduct detailed analysis to study the benefit coming from MiME's effort to properly capture the multilevel structure of EHR. Specifically, we compare model performance in terms of true positive rates and false positive rates for HF prediction together with visualization. We provide insight as to why MiME succeeds in cases where baselines fail.
- We test model performance on a smaller dataset that imitates the real-world setting where enough data has not been collected. MiME, jointly trained with auxiliary tasks demonstrated a significant generalization power compared to other models, outperforming the strongest baseline by 2.5% PR-AUC for HF prediction.

The rest of the paper is structured as follows: In section **??** we first describe EHR data structure, followed by the mathematical notations, the description of MiME, and the joint training strategy. In section **??**, we demonstrate the advantage of MiME through extensive empirical evaluations and detailed analysis. After we discuss related works in section **??**, we summarize our work and conclude this paper with future work discussion in section **??**.

6.2 Methodology

In this section, we first describe the structure of EHR data, then introduce the notations we will use in the MiME embedding. Next we describe the MiME framework and its multi-task extension MiME_{aux}.

6.2.1 EHR Data Modeling

In most EHR systems, data are stored in relational databases, where tables containing information about patient demographics, encounters, medication orders and procedure

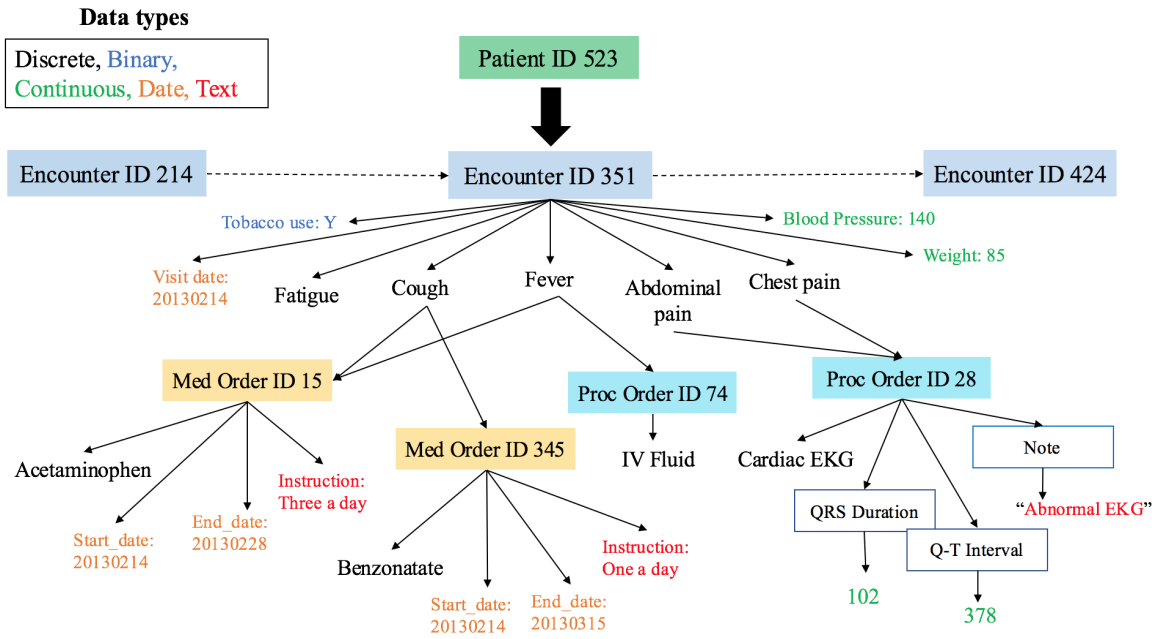


Figure 6.1: In EHR data, medical codes are structured hierarchically with heterogeneous relations, e.g., medication *Acetaminophen* and procedure *IV fluid* are correlated, while both of them occur due to the diagnosis *Fever*.

orders are linked using keys such as *patient ID*, *encounter ID* or *medication order ID*. Although there is more than one way to view EHR data, when performing computational tasks such as disease prediction, we typically group all information by individual patients as depicted by Figure 6.1.

Therefore EHR data can be seen as a collection of individual patient records, where each patient has a sequence of encounters over time. Within an encounter, there are multiple clinical event types and each type includes multiple possible events. Different types of events are structured along the clinical decision making process. For example, given a set of diagnosis codes, each are associated with a set of medication orders and procedure orders. Each medication order contains information such as the medication code, start date, end date and instructions. Procedure orders contain the procedure code and possible lab results. As shown by Figure 6.1, procedures such as *Cardiac EKG* come with several results (QRS duration, Q-T interval, notes), but *IV Fluid* does not. Note that some diagnosis might not be associated with any medication or procedure orders, as *Fatigue* in Figure 6.1.

In order to leverage deep learning's ability to learn complex features from input data, the EHR data must be transformed into a vector. A patient can be seen as a sequence of visits (*i.e.* encounters), and the visit sequence can be transformed into a vector using recurrent neural networks (RNN) [169] or convolutional neural networks (CNN) [170, 2]. However, transforming a visit to a vector leaves some room for imagination as it has a multilevel structure with inter-code relationships. A typical approach adopted by a number of previous works [9, 10, 13, 14, 155, 168] flattens all events within a visit and represents it as a vector of individual features, losing the structural information embedded in the visit. In this work, `MiME` especially focuses on the three types of clinical events, namely diagnosis, medication and procedures, and explicitly captures the heterogeneous relationships between the diagnosis codes and the medication/procedure codes, However, the proposed methodology of `MiME` can be applied to model other relations in EHR data as well.

6.2.2 Notations of `MiME`

Consider patient $\mathcal{P}$ makes a sequence of visits $\mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_t$ over time. Each visit $\mathcal{V}_t$ contains a varying number of diagnosis codes $d_{t,0}, d_{t,1}, \ldots, d_{t,|d_t|} \in \mathcal{D}$, where $\mathcal{D}$ denotes the set of unique diagnosis codes in the data, and $|d_t|$ the total number of diagnosis codes in $\mathcal{V}_t$. Each diagnosis code $d_{t,i}$ can be associated with one or more medication codes $m_{t,i,0}, m_{t,i,1} \ldots, m_{t,i,|m_{t,i}|} \in \mathcal{M}$ and one or more procedure codes $p_{t,i,0}, p_{t,i,1} \ldots, p_{t,i,|p_{t,i}|} \in \mathcal{P}$ where $\mathcal{M}$ and $\mathcal{P}$ respectively denote the set of unique medication codes and the set of unique procedure codes in the data. $|m_{t,i}|$ and $|p_{t,i}|$ respectively denote the total number of medication codes and procedure codes associated with $d_{t,i}$. As shown by Figure 6.1, some medication codes (*e.g. Acetaminophen*) can be shared by two or more diagnosis codes (*e.g. Cough, Fever*), if the doctor ordered a single medication for more than one diagnosis. In that case, each diagnosis code will have its own copy of the medication code attached to it. To reduce clutter, we omit the index t indicating t-th visit in all diagnosis, medication and

procedure codes, and express them $d_i, m_{i,j}$ and $p_{i,k}$ when we are discussing a single visit.

6.2.3 Description of MiME

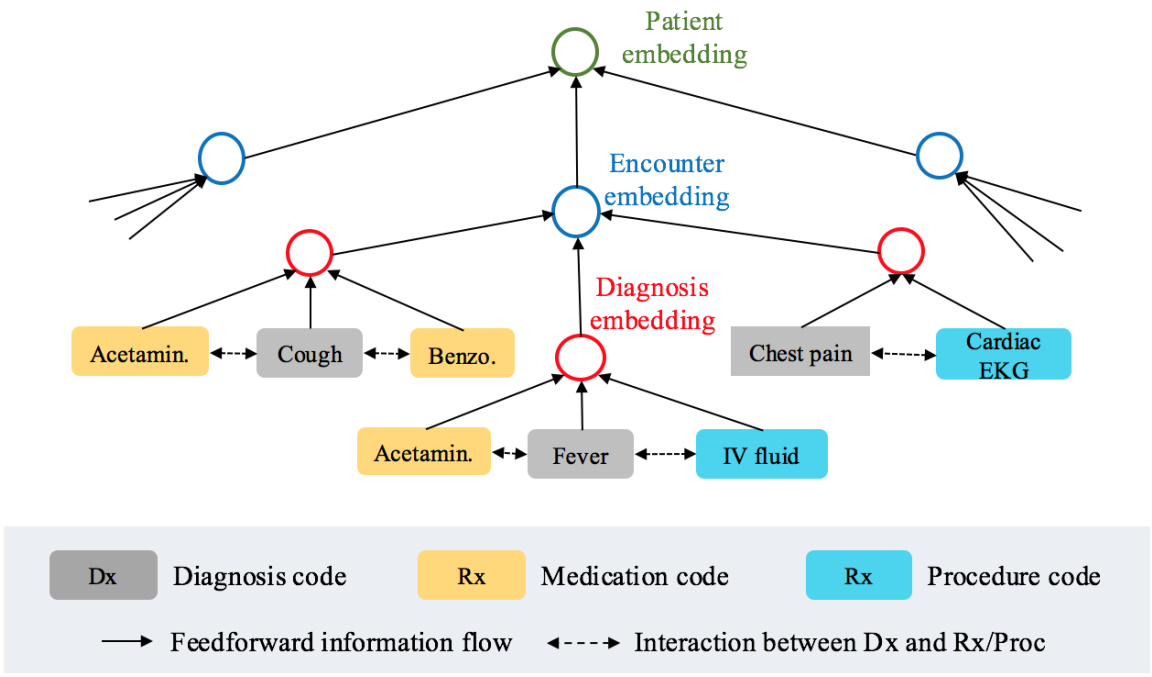


Figure 6.2: Model architecture of MiME, where medical concepts are embedded into multiple levels: diagnosis-level, encounter-level, and patient-level.

As discussed in section 6.2.1, previous approaches often flatten the EHR structure such that diagnosis codes, medication codes and the procedure codes are packed in the same vector. Then a single visit $\mathcal{V}_t$ can be expressed as a binary vector $\mathbf{x}_t \in \{0,1\}^{|\mathcal{D}|+|\mathcal{M}|+|\mathcal{P}|}$ where each dimension corresponds to a specific diagnosis, medication, and procedure code. Therefore encoding a sequence of visits made by a patient was achieved as follows:

$$\mathbf{v}_t = \sigma(\mathbf{W}_x \mathbf{x}_t)$$

$$\mathbf{h} = f(\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_t)$$

where $\mathbf{W}_x$ is the embedding matrix that converts the binary vector $\mathbf{x}$ to a lower-dimensional visit representation[1], σ a non-linear activation function such as sigmoid or rectified linear

[1]We omit bias variables throughout the paper to reduce clutter.

unit (ReLU), $f(\cdot)$ a function that maps a sequence of visit representations $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_t$ to a patient representation $\mathbf{h}$. $f(\cdot)$ can be simply an RNN or a complex combination of RNNs and CNN and attention mechanisms [3].

MiME treats the diagnosis codes $\mathcal{D}$, medication codes $\mathcal{M}$ and the procedure codes $\mathcal{P}$ separately, in order to capture the relationship between $\mathcal{D}$ and $\mathcal{M}$, and $\mathcal{D}$ and $\mathcal{P}$ depicted by Figure 6.1. Then a single visit $\mathcal{V}$ (omitting the index t of the visit) consists of two levels: the diagnosis level, and the medication/procedure level. Figure 6.2 illustrates how the proposed model MiME builds the representation of $\mathcal{V}$ in a bottom-up fashion and how we capture both co-occurrence and parent-child relationship among medical codes in Figure 6.1 via multilevel embedding. A single diagnosis code d_i, with its associated medication codes $m_{i,0}, m_{i,1}, \ldots$ and procedure codes $p_{i,0}, p_{i,1}, \ldots$, forms a single diagnosis embedding $\mathbf{d}_i$ [2]. Then multiple diagnosis embeddings $\mathbf{d}_0, \ldots, \mathbf{d}_{|d|}$ in a single encounter forms a visit embedding $\mathbf{v}$, which in turn forms a patient embedding $\mathbf{h}$ with other visit embeddings. As shown in Figure 6.2, we also aim to capture the interactions between a diagnosis code and its associated medication codes and procedure codes, in order to obtain a more accurate diagnosis embedding. The skeletal form of MiME is as follows,

$$\mathbf{v} = \sum_{i}^{|d|} \underbrace{f_d\Big(d_i, \sum_{j}^{|m_i|} f_{d,m}(d_i, m_{i,j}), \sum_{k}^{|p_i|} f_{d,p}(d_i, p_{i,k})\Big)}_{\text{Diagnosis embedding } \mathbf{d}_i} \tag{6.1}$$

$f_{d,m}(\cdot,\cdot)$: Function that captures the relationship between a diagnosis and a medication order.

$f_{d,p}(\cdot,\cdot)$: Function that captures the relationship between a diagnosis and a procedure order.

$f_d(\cdot,\cdot,\cdot)$: Function that generates a single diagnosis embedding $\mathbf{d}_i$.

[2] Diagnosis embedding $\mathbf{d}_i$, which is vector representation of a diagnosis code and its associated medication/procedure codes, is not to be confused with a diagnosis code embedding which is vector representation of a single diagnosis code d_i

In this work we propose two versions of MiME and conduct experiments with both models. The first version, $\texttt{MiME}_{sum}$ implements the functions f_d, $f_{d,m}$ and $f_{d,p}$ as follows:

$$
\begin{aligned}
f_{d,m}(d_i, m_{i,j}) &= f_{rx}(m_{i,j}) \\
f_{d,p}(d_i, p_{i,k}) &= f_{pr}(p_{i,k}) \\
f_d(\cdot,\cdot,\cdot) = \mathbf{d}_i &= \sigma\Big(\mathbf{W}_d\Big(f_{dx}(d_i) + \sum_{j}^{|m_i|} f_{d,m}(d_i, m_j) + \sum_{k}^{|p_i|} f_{d,p}(d_i, p_j)\Big)\Big)
\end{aligned}
$$

where $f_{dx}(\cdot)$, $f_{rx}(\cdot)$ and $f_{pr}(\cdot)$ respectively denote the embedding functions that map given diagnosis, medication, and procedure codes to their corresponding vector representations. In MiME $_{sum}$, the size of all code embeddings are the same as they are summed to obtain the diagnosis embedding $\mathbf{d}_i$. $\mathbf{W}_d$ is the weight matrix used to obtain the representation of a single diagnosis object, which is applied to the sum of diagnosis code embedding $f_{dx}(d_i)$ and its associated medication/procedure code embeddings $f_{rx}(m_{i,j})$ and $f_{pr}(p_{i,k})$. σ is the non-linear activation function, for which we used ReLU in this work. In $\texttt{MiME}_{sum}$, the functions $f_{d,m}(\cdot,\cdot)$ and $f_{d,m}(\cdot,\cdot)$, which aim to capture the interactions between a diagnosis code and medication/procedure codes are simply implemented as a medication code embedding $f_{rx}(m_{i,j})$ and a procedure code embedding $f_{pr}(p_{i,k})$. In $\texttt{MiME}_{sum}$, it is assumed that the code interactions are implicitly captured by the summation inside the diagnosis embedding generator $f_d(\cdot,\cdot,\cdot)$. Although this is a rather straightforward implementation of Eq. 6.1, it shows robust performance in various prediction tasks as we will demonstrate in the next section.

The second implementation of Eq. 6.1, which we name $\texttt{MiME}_{bp}$, captures the relationship between the diagnosis code and medication/procedure orders more explicitly as follows:

$$
\begin{aligned}
f_{d,m}(d_i, m_{i,j}) &= \sigma(\mathbf{W}_{dm} f_{dx}(d_i) \odot f_{rx}(m_{i,j})) \\
f_{d,p}(d_i, p_{i,k}) &= \sigma(\mathbf{W}_{dp} f_{dx}(d_i) \odot f_{pr}(p_{i,k})) \\
f_d(\cdot,\cdot,\cdot) = \mathbf{d}_i &= \sigma\Big(\mathbf{W}_d\Big(f_{dx}(d_i) + \sum_{j}^{|m_i|} f_{d,m}(d_i, m_j) + \sum_{k}^{|p_i|} f_{d,p}(d_i, p_j)\Big)\Big)
\end{aligned}
$$

Both $f_{d,m}$ and $f_{d,p}$ use a form of bilinear pooling to explicitly capture the interactions between the diagnosis code and medication/procedure codes. Bilinear pooling [171] derives a scalar feature f_i between two embeddings $\mathbf{x}, \mathbf{y}$ such that $f_i = \mathbf{x}^T \mathbf{W}_i \mathbf{y}$ where $\mathbf{W}_i$ is a trainable weight matrix. Since we typically extract many features $f_0, \ldots, f_i$, to capture the interaction between two embeddings, bilinear pooling requires us to train a number of weight matrices (*i.e.* weight tensor). Due to the large number of parameters required, researchers developed more efficient methods such as compact bilinear pooling [172, 173] and low-rank bilinear pooling [174]. In this work, we used a simplified version of low-rank bilinear pooling with Hadamard product [174], which showed impressive performance in Visual Question Answering (VQA), where it is important to effectively capture the interaction between two different domains:text-based question and image. Weight matrices $\mathbf{W}_{dm}$ and $\mathbf{W}_{dp}$ are used to send the diagnosis code embedding $f_{dx}(d_i)$ into another latent space, where the interaction between diagnosis d_i and the corresponding $m_{i,j}$ (or procedure $p_{i,k}$) can be captured effectively. In $\texttt{MiME}_{bp}$ the interactions captured by $f_{d,m}$ and $f_{d,p}$ are added to the diagnosis code embedding $f_{dx}(d_i)$, which can be interpreted as adjusting the diagnosis representation according to its associated medication and procedure orders [3].

6.2.4 Joint Training with Auxiliary Tasks

Since the expressive power of deep learning models comes from their large number of parameters in multiple layers, they typically require a large amount of training data. This poses a challenge in deep embedding tasks on smaller datasets, which are often the case in new EHR systems, with a small number of patients each with a relatively fewer number of visits. In such cases, to prevent the deep models from over-fitting, we adopt a multi-task learning (MTL) approach that can inductively transfer the knowledge contained in multiple auxiliary prediction tasks to improve the model's generalization performance on the original prediction task. MTL was shown to improve model robustness in medical

[3] For both $\texttt{MiME}_{sum}$ and $\texttt{MiME}_{bp}$, we also implemented f_d as a concatenation of $f_{dx}(d_i), \sum f_{d,m}$ and $\sum f_{d,p}$ instead of a summation, but it showed slightly weaker performance.

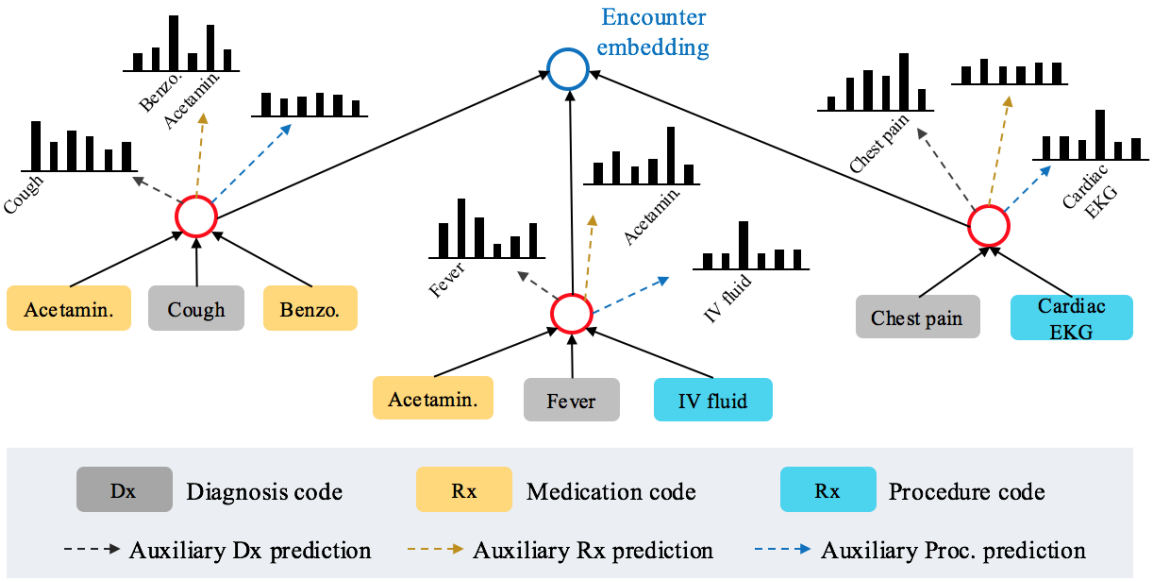


Figure 6.3: Diagnosis embedding and its associated auxiliary prediction tasks, where Dx prediction, Rx prediction, and Procedure prediction serve as auxiliary tasks for improving the performance of some specific target task.

concept embedding. For example, in [175], multiple severity status of a disease were seen as multiple tasks for improving accuracy in disease prediction.

For deep learning models which are typically composed of layers of neurons, an MTL strategy can require certain neurons to be shared among all tasks, and certain neurons to be specialized for specific tasks [176]. Specifically, in longitudinal EHR data based health analytics, the patient embedding $\mathbf{h}$ is typically used for specific prediction tasks, such as heart failure prediction or mortality prediction. And the representation power of $\mathbf{h}$ comes from properly capturing each visit $\mathcal{V}_t$, and modeling the longitudinal aspect with the function $f(\mathbf{v}_0, \ldots, \mathbf{v}_t)$. Since the focus of this work is on modeling a single visit $\mathcal{V}_t$, our MTL strategy performs auxiliary tasks while obtaining the t-th visit embedding $\mathbf{v}_t$ as

follows.

$$\mathbf{v}_t = \sum_{i}^{|d_t|} \mathbf{d}_{t,i}$$

$$\hat{d}_{t,i} = p(d_{t,i}|\mathbf{d}_{t,i}) = \text{softmax}(\mathbf{W}_{ad}\mathbf{d}_{t,i})$$

$$\hat{\mathbf{m}}_{t,i} = p(\mathbf{m}_{t,i}|\mathbf{d}_{t,i}) = \sigma(\mathbf{W}_{am}\mathbf{d}_{t,i})$$

$$\hat{\mathbf{p}}_{t,i} = p(\mathbf{p}_{t,i}|\mathbf{d}_{t,i}) = \sigma(\mathbf{W}_{ap}\mathbf{d}_{t,i})$$

$$L_{aux} = -\lambda_{aux} \sum_{t}^{T} \Big(CE(d_{t,i}, \hat{d}_{t,i}) + CE(\mathbf{m}_{t,i}, \hat{\mathbf{m}}_{t,i}) + CE(\mathbf{p}_{t,i}, \hat{\mathbf{p}}_{t,i}) \Big)$$

Given diagnosis embeddings $\mathbf{d}_{t,0}, \ldots, \mathbf{d}_{t,|d_t|}$, before aggregating them to obtain $\mathbf{v}_t$, the model predicts the diagnosis code $d_{t,i}$, and the associated medication codes $\mathbf{m}_{t,i} = m_{t,i,0}, \ldots, m_{t,i,|m_{t,i}|}$ and procedure codes $\mathbf{p}_{t,i} = p_{t,i,0}, \ldots, p_{t,i,|p_{t,i}|}$, involved with $\mathbf{d}_{t,i}$. $\mathbf{W}_{ad}$, $\mathbf{W}_{am}$ and $\mathbf{W}_{ap}$ are weight matrices used for predicting the diagnosis, medication and procedure code, respectively. $CE(\cdot, \cdot)$ denotes the cross-entropy function, and λ_{aux} is the coefficient for the auxiliary loss term. We used the softmax function for predicting $d_{t,i}$ since in a single diagnosis embedding, there is only one diagnosis code involved. However, there could be no medication/procedure codes associated with $d_{t,i}$, and therefore we used the sigmoid function for predicting medication and procedure codes.

These auxiliary tasks guide the model to learn diagnosis embeddings $\mathbf{d}_{t,i}$ that are representative of the specific codes involved with it. Correctly capturing the events within an encounter is a foundation of all downstream prediction tasks, and these general-purpose auxiliary tasks, combined with the specific target task, encourage the model to learn visit embeddings $\mathbf{v}_t$ that are not only tuned for the target prediction task, but also grounded in general-purpose foundational knowledge.

6.3 Experiments

In this section, we first talk about the dataset used for the experiments, followed by the description of the baseline models. Then we introduce three prediction tasks: heart failure

Table 6.1: Statistics of the Sutter PAMF dataset

# of patients	30,764
# of visits	616,073
Avg. # of visits per patient	20.0
# of unique codes	2,311 (Dx:388, Rx:99, Proc:1,824)
Avg. # of Dx per visit	1.93 (Max: 29)
Avg. # of Rx per diagnosis	0.31 (Max: 17)
Avg. # of Proc. per diagnosis	0.36 (Max: 10)

prediction, sequential disease prediction and medication prediction. After describing the training details, we analyze the experiment results in detail. We make the source code of `MiME` publicly available at `https://github.com/mp2893/mime`.

6.3.1 Source of Data

We conducted all our experiments with the EHR data provided by Sutter Palo Alto Medical Foundation (PAMF). The dataset was constructed for a heart failure study, which consisted of 30,764 senior patients (age 40-85), observed for a 18 months period. We extracted the diagnosis codes, medication codes and the procedure codes from encounter records, medication orders and procedure orders in the data. We used Clinical Classification Software for ICD9-CM[4] to group the ICD9 diagnosis codes into 388 categories. Generic Product Identifier Drug Group[5] was used to group the medication codes into 99 categories. Clinical Classifications Software for Services and Procedures[6] was used to group the CPT procedure codes into 1,824 categories. When grouping the codes, any code that did not fit into the grouper formed its own category. The summary statistics of our dataset is described in Table 6.1.

6.3.2 Baseline Models

Since our prediction tasks, which will be described in the following section, are based on processing sequences of visits (*i.e.* encounters), we use various approaches to model

[4]https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp
[5]http://www.wolterskluwercdi.com/drug-data/medi-span-electronic-drug-file/
[6]https://www.hcup-us.ahrq.gov/toolssoftware /ccs_svcsproc/ccssvcproc.jsp

encounters as baselines.

- **raw**: A single visit $\mathcal{V}_t$ is represented by a binary vector $\mathbf{x}_t \in \{0,1\}^{|\mathcal{D}|+|\mathcal{M}|+|\mathcal{P}|}$. Only the dimensions corresponding to the codes occurring in that visit is set to 1, and the rest are 0.

- **linear**: The binary vector $\mathbf{x}_t$ is linearly transformed to a lower- dimensional vector $\mathbf{v}_t = \mathbf{W}_x\mathbf{x}_t$ where $\mathbf{W}_x \in \mathbb{R}^{d\times(|\mathcal{D}|+|\mathcal{M}|+|\mathcal{P}|)}$ is the embeding matrix. This is equivalent to taking the vector representations of the codes (*i.e.* columns of the embedding matrix $\mathbf{W}_x$) occurring in the visit $\mathcal{V}_t$, and summing them all to derive a single vector $\mathbf{v}_t \in \mathbb{R}^d$.

- **sigmoid**: The binary vector $\mathbf{x}_t$ is transformed to a lower- dimensional vector $\mathbf{v}_t = \sigma(\mathbf{W}_x\mathbf{x}_t)$ where $\sigma(\cdot)$ is the sigmoid function. This is simply adding non-linearity to **linear**.

- **relu**: This is equivalent to **sigma**, except that we use the rectified linear unit (ReLU) instead of sigmoid for non-linearity.

- **linear**$_m$**, sigmoid**$_m$**, relu**$_m$: The visit representation vector $\mathbf{v}_t$ obtained in previous baselines is divided by the number of codes occurring in the visit.

- **sigmoid**$_{mlp}$**, relu**$_{mlp}$: We add one more layer to **sigmoid** and **relu** to increase their expressivity. The visit embedding is now $\mathbf{v}_t = \sigma(\mathbf{W}_{x_2}\sigma(\mathbf{W}_{x_1}\mathbf{x}_t))$ where σ is either the sigmoid function or ReLU. We do not test **linear**$_{mlp}$ since two consecutive linear layers can be collapsed to a single linear layer.

- **Med2Vec**: We train Med2Vec [12] on the training data for 100 epochs, then obtain visit representation vectors for every visit of all patients. We then use these visit vectors when training for prediction tasks. We test this model as a representative case of unsupervised embedding approach using EHR data.

6.3.3 Prediction Tasks

Heart failure prediction The objective is to predict the onset of heart failure (HF), given the 18-months observation records discussed in section 6.3.1. Among 30,764 patients, 3,414 were case patients who were diagnosed with HF within a 1-year window after the 18-months observation. The remaining 27,350 patients were control patients. The case-control selection criteria were from [137]. where various aspects were considered such as the age, diagnosis codes, and frequency of hospital visits. While an accurate prediction of HF can save a large amount of costs and lives [177], this task is also suitable for assessing how well a model can learn the relationship between the external label (*i.e.* the label information is not inherent in the EHR data) and the features (*i.e.* codes) . For HF prediction, after the sequence of visit representation vectors $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_T$ were obtained, we fed them into a function $f(\cdot)$ that maps each sequence to a fixed size vector $\mathbf{h}$, as described at the beginning of section 6.2.3. Specifically in this experiment, we used Gated Recurrent Units (GRU) [119] for $f(\cdot)$, and its hidden vector at the last timestep $\mathbf{h}_T$ for $\mathbf{h}$. Then we applied logistic regression to $\mathbf{h}$ to obtain a value between 0 (no HF onset) and 1 (HF onset). The performance was measured by the Area under the Receiver Operating Characteristic (ROC-AUC) and Area under the Precision-Recall Curve (PR-AUC).

Sequential disease prediction The objective is to predict the diagnosis codes occurring in visit $\mathcal{V}_{t+1}$, given all past visits $\mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_t$. The input features are diagnosis codes $\mathcal{D}$, medication codes $\mathcal{M}$ and procedure codes $\mathcal{P}$, while the output space only has diagnosis codes $\mathcal{D}$. This task is useful for preemptively assessing the patient's potential future risk [10], but is also appropriate for assessing how well a model captures the progression of the patient status over time. We used GRU as the mapping function $f(\cdot)$, and hidden vectors from all timesteps were fed to the softmax function with $|\mathcal{D}|$ output classes to perform sequential prediction. The performance was measured by sorting the predicted diagnosis codes for $\mathcal{V}_{t+1}$ by their prediction value, and calculating $Recall@k$ using the true diagnosis codes of $\mathcal{V}_{t+1}$.

Table 6.2: Prediction Performance on Benchmark Tasks. The two strongest performances are marked in bold.

	HF prediction			**Seq. Dx prediction**		**Med. prediction**	
	test loss	test ROC-AUC	test PR-AUC	test loss	test recall@5	test loss	test recall@5
raw	0.2665 (0.0094)	0.8327 (0.0111)	0.4843 (0.0220)	7.2121 (0.0319)	0.5329 (0.0016)	3.1842 (0.0575)	**0.8556** (0.0056)
linear	0.2636 (0.0098)	0.8372 (0.0108)	0.4927 (0.0147)	7.1458 (0.0311)	0.5450 (0.0008)	3.1644 (0.0504)	**0.8547** (0.0054)
sigmoid	0.2649 (0.0100)	0.8351 (0.0137)	0.4794 (0.0236)	7.3494 (0.0438)	0.5110 (0.0054)	3.1551 (0.0658)	0.8506 (0.0059)
relu	0.2642 (0.0098)	0.8358 (0.0096)	0.4939 (0.0313)	7.1597 (0.0349)	0.5439 (0.0005)	**3.1314** (0.0503)	0.8541 (0.0055)
linear$_m$	0.2688 (0.0084)	0.8325 (0.0104)	0.4552 (0.0150)	7.2548 (0.0354)	0.5304 (0.0020)	3.1541 (0.0558)	0.8494 (0.0055)
sigmoid$_m$	0.2698 (0.0079)	0.8281 (0.0117)	0.4525 (0.0259)	7.5575 (0.0420)	0.4784 (0.0053)	3.1919 (0.0393)	0.8345 (0.0043)
relu$_m$	0.2651 (0.0092)	0.8374 (0.0092)	0.4762 (0.0270)	7.2714 (0.0318)	0.5281 (0.0014)	3.1373 (0.0536)	0.8509 (0.0034)
sigmoid$_{mlp}$	0.3334 (0.0065)	0.6630 (0.0176)	0.2222 (0.0146)	8.7886 (0.0257)	0.2132 (0.0037)	3.5345 (0.0518)	0.7975 (0.0066)
relu$_{mlp}$	0.2656 (0.0083)	0.8340 (0.0102)	0.4825 (0.0279)	7.1487 (0.0346)	**0.5464** (0.0012)	3.1933 (0.0479)	0.8457 (0.0039)
med2vec	0.2677 (0.0070)	0.8358 (0.0069)	0.4668 (0.0069)	7.2429 (0.0283)	0.5317 (0.0011)	3.1976 (0.0503)	0.8455 (0.0067)
MiME$_{sum}$	**0.2562** (0.0091)	**0.8503** (0.0100)	**0.5040** (0.0181)	**7.1372** (0.0345)	**0.5463** (0.0017)	**3.1007** (0.0429)	0.8529 (0.0030)
MiME$_{bp}$	**0.2546** (0.0103)	**0.8505** (0.0118)	**0.5130** (0.0196)	**7.1326** (0.0314)	0.5460 (0.0012)	3.1351 (0.0437)	0.8533 (0.0047)

Medication prediction The task is to predict medication codes occurring in $\mathcal{V}_t$, given all past and current visits $\mathcal{V}_0, \ldots, \mathcal{V}_t$. However, in the current visit $\mathcal{V}_t$, we are only given the diagnosis codes $\mathcal{D}$ and procedure codes $\mathcal{P}$, while all three codes are given in the past visits. Unlike sequential disease prediction where there are at least one diagnosis code per visit, medication codes might not exist in many visits. Therefore, for each patient, we looked for the last visit that had at least one medication code, and removed all visits that came after that. Medication prediction is important for ensuring the correctness of a patient's medication list [168], but it also helps us assess how well a model can capture the co-occurrence pattern between different code domains, $\mathcal{D}, \mathcal{M}$ and $\mathcal{P}$. GRU was used as the mapping function $f(\cdot)$, and its last hidden vector was fed to the softmax function with $|\mathcal{M}|$ output classes. The performance was measured by calculating $Recall@k$ between the predicted medication codes and the true medication codes of $\mathcal{V}_t$.

6.3.4 Training Details

All models were implemented in TensorFlow 1.4 [178], and trained with a system equipped with Intel Xeon E5-2620, 512TB memories and 8 Nvidia Pascal Titan X's. We used Adam [179] for optimization, with the learning rate set to $1e-3$.

In all tasks, we divided the data into a training set (70%), a validation set (10%) and a test set (20%). In all tasks, we trained all models with the minibatch of 20 patients for 20,000 iterations, which was a sufficient number of iterations for achieving the best performance for

all three benchmark tasks. At every 100 iterations, we evaluated the performance against the validation set, and only if the validation performance was better than before, we evaluated the performance against the test set. In all three tasks, we tested each model for 5 times using different random seeds for splitting the training set and the test set. In each run, we obtained the best validation performance and the corresponding test performance. We report their mean values and standard deviation in the next section.

The size of the visit vector $\mathbf{v}_t$ was 256 in all baselines except **raw**. We ran a number of preliminary experiments with values 64, 128 and 512, and we concluded that 256 was sufficient for all models to obtain optimal performance. For $\texttt{MiME}_{sum}$ and $\texttt{MiME}_{bp}$, we adjusted the size of the code embeddings $f_{dx}(\cdot), f_{rx}(\cdot)$ and $f_{pr}(\cdot)$ and the diagnosis embeddings $\mathbf{d}_i$ to match the number of parameters to that of **linear, sigmoid, relu, linear**$_m$**,** **sigmoid**$_m$ and **relu**$_m$. **Med2Vec** was also trained to obtain 256 dimensional visit vectors. Note that **sigmoid**$_{mlp}$ and **relu**$_{mlp}$ used 256×256 more parameters than other models. All models were connected to a GRU as described in section 6.3.3, for the cell size of which we used 256 in all prediction tasks. We used L_2 regularization with the coefficient $1e-4$ for all models. We did not use any dropout technique.

6.3.5 Experiment Results

We first evaluate the prediction performance of all models across the three prediction tasks. As shown in Table 6.2, the proposed models consistently yielded the lowest test loss for all benchmark tasks, with one exception of medication prediction, demonstrating MiME's strong generalization performance.

For HF prediction, both $\texttt{MiME}_{sum}$ and $\texttt{MiME}_{bp}$ demonstrated stronger prediction performance than all baselines in all metrics, especially the PR-AUC. This is especially meaningful in HF prediction since we want the model to make as many correct positive predictions as possible and miss as few case patients as possible, and PR-AUC is a better metric for our goal given that the dataset is class-imbalanced [180]. Both the proposed models, however, showed the highest ROC-AUC as well, showing that they also are capable of correctly clas-

sifying control patients, which is important in real-world applications where frequent false alarms lead to alert fatigue. It is also notable that $\texttt{MiME}_{bp}$ outperforms $\texttt{MiME}_{sum}$ in terms of PR-AUC, suggesting that explicitly capturing the interactions between the diagnosis codes and the medication/procedure codes improves the model's ability to correctly recognize case patients.

For sequential disease prediction, both MiME models demonstrated the lowest test loss. But in terms of $Recall@5$, **relu**$_{mlp}$ showed the strongest performance. However, the difference of $Recall@5$ between **relu**$_{mlp}$ and both MiME models were marginal, showing that MiME can properly capture the temporal progression of the patient status. It is noteworthy that **linear** displayed very competitive performance compared to the best performing models. This is due to the fact that chronic conditions such as hypertension or diabetes persist over a long period of time, and sequentially predicting them becomes an easy task that does not require an expressive model. This was also reported in [10] where a strategy to choose the most frequent diagnosis code as the prediction showed competitive performance in a similar task.

In medication prediction, both MiME models showed better performance than most baseline models in terms of both test loss and $Recall@5$. Interestingly, however, the best $Recall@5$ was achieved by **raw** which does not even learn visit representations at all. This suggests that medication prediction is an easy enough task that a single GRU can memorize the code co-occurrence patterns in the past visits $\mathcal{V}, \ldots, \mathcal{V}_{t-1}$ and accurately predict the medication codes most likely to occur in the current visit $\mathcal{V}_t$.

Overall, MiME models, especially $\texttt{MiME}_{bp}$ demonstrated good performance in all benchmark tasks, and it is notable that they significantly outperformed the baseline models in the most complex task, namely HF prediction, where the relationship between the label and the features (*i.e.* codes) from the data was more than straightforward. For baseline models, using ReLU as an activation function consistently yielded better results than using the sigmoid function, especially when we compare **relu**$_{mlp}$ and **sigmoid**$_{mlp}$. This is a

Table 6.3: Heart failure (HF) prediction performance in terms of false positive rate (FPR) and true positive rate (TPR). Models with significantly lower TPR values are grayed out as they have minimal value for HF prediction.

	False positive rate	True positive rate
raw	0.0176 (0.0067)	0.0304 (0.0042)
linear	0.0186 (0.0066)	0.0331 (0.0055)
sigmoid	0.0123 (0.0036)	0.0246 (0.0037)
relu	0.0199 (0.0075)	0.0334 (0.0064)
linear$_m$	0.0215 (0.0086)	0.0294 (0.0083)
sigmoid$_m$	0.0147 (0.0033)	0.0249 (0.0035)
relu$_m$	0.0214 (0.0034)	0.0330(0.0051)
sigmoid$_{mlp}$	0 (0)	0 (0)
relu$_{mlp}$	0.0171 (0.0042)	0.0300 (0.0047)
med2vec	0.0214 (0.0037)	0.0305 (0.0032)
MiME$_{sum}$	0.0178 (0.0041)	0.0325 (0.0063)
MiME$_{bp}$	**0.0170** (0.0033)	**0.0341** (0.0030)

rather natural outcome considering that, given large activation signals, the sigmoid function produces near-zero gradient values for updating the parameters. It could be surprising that **raw** demonstrated decent performance in all three benchmark tasks. However, GRU, being a very expressive model with many non-linear operations inside, can possibly undertake a considerable portion of the prediction task, or even the entire task if the task is simple enough, which was the case with **raw** in medication prediction. **Med2Vec** showed better performance than some baseline models, but it was consistently outperformed by **linear**, suggesting that features learned by unsupervised approaches alone are less effective than features learned in an end-to-end fashion when it comes to performing a specific prediction task.

6.3.6 Performance Analysis and Visualization

In order to further evaluate the strength of MiME, we set HF prediction as a target task and study all models' performance in terms of true positive (TP) and false positive (FP) rates, using 0.5 as the cut-off threshold for binary prediction. Results in Table 6.3 show MiME$_{bp}$ clearly outperforms other models including MiME$_{sum}$ in terms of high TP rate, which is expected from its high PR-AUC score. This indicates MiME$_{bp}$ has the highest chance of

correctly classifying HF cases, which is the primary objective of HF prediction.

Another important aspect of HF prediction is to reduce the FP rate to prevent alert fatigue as mentioned in section 6.3.5. Models that use the sigmoid non-linearity showed very low FP rates, at the cost of significantly sacrificing TP rate. **sigmoid**$_{mlp}$ even went to the very extreme by classifying all test samples as HF controls to achieve 0.0 FP rates, as well as 0.0 TP rate. As a contrast, $\texttt{MiME}_{bp}$ demonstrated the lowest FP rate, while achieving the highest TP rate. The fact that $\texttt{MiME}_{bp}$ clearly outperformed all models including $\texttt{MiME}_{sum}$ suggests that explicitly capturing the relationship between the diagnosis code and medication/procedure codes produces better visit representations.

To better understand how $\texttt{MiME}_{bp}$'s effort in capturing such relationships leads to a high TP rate and a low FP rate, we conduct two analyses. For TP rate, we select 125 HF cases from the test set that both **linear** and **relu** missed but $\texttt{MiME}_{bp}$ caught. Then we use t-SNE [18] to plot these cases together with HF controls from the test set on a 2D plane to study how **linear** and **relu** mistook those cases as controls but $\texttt{MiME}_{bp}$ did not. For FP rate, we select 108 HF controls from the test set that **linear** and **relu** misclassified as cases but $\texttt{MiME}_{bp}$ did not. Then we plot them on a 2D plane with HF cases from the test set to perform similar analysis as before. We chose **linear** and **relu** for comparison with $\texttt{MiME}_{bp}$ since they both showed good HF prediction performance in terms of TP rate and FP rate. For both analyses we used the last hidden layer of the GRU as the patient representation $\mathbf{h}$. Figure 6.4 shows scatterplots for **linear** and **relu** making FN predictions (Fig. 6.4a,b) and FP predictions (Fig. 6.4d,e), and $\texttt{MiME}_{bp}$'s behavior in both cases (Fig. 6.4c,f). 500 controls in the top row were chosen so that their embeddings were closest to the 125 case embeddings of both **linear** and **relu**. 500 cases in the bottom row were chosen in the same manner. Unlike **linear** and **relu** that confuse cases and controls in Figure 6.4a,b,c,d, $\texttt{MiME}_{bp}$ can clearly distinguish the same cases and controls.

Additionally, we wanted to check if $\texttt{MiME}_{bp}$ were able to correctly identify cases and controls that confused **linear** and **relu**, because of its ability to capture the complex code

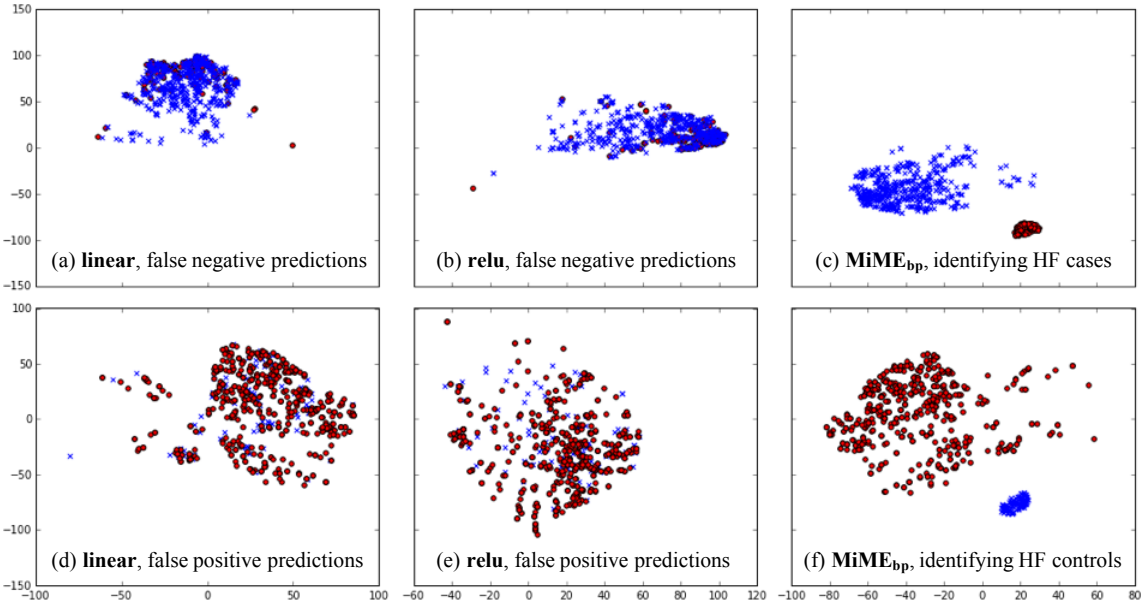


Figure 6.4: Scatterplots visualize how **linear** and **relu** recognize cases (red circles) and controls (blue crosses) in false negative and false positve predictions. Plots in the same row use the same cases and controls. MiME $_{bp}$ clearly distinguishes cases and controls in both cases.

interactions. We took cases from the top row of Figure 6.4 and controls from the bottom row to see if they had complex diagnosis-medication/procedure interactions in their records. We defined *interaction ratio* as below to calculate for a patient how many visits have at least 2 diagnosis codes and at least one medication/procedure code,

$$\text{interaction ratio} = \frac{\#\mathcal{V}_t \text{ where } |d_t| \geq 2,\ \sum_i^{|d_t|} \left(|m_{t,i}| + |p_{t,i}|\right) \geq 1}{T}$$

where T denotes the total number of visits. The mean interaction ratio of the top row cases was 32.2%, and the bottom row controls was 32.3%. The mean interaction ratio of the entire test set was 26.1%, supporting our assumption that MiME $_{bp}$'s advantage over **linear** and **relu** comes from its ability to capture the code interactions.

For further analysis, we aimed to see if **linear** and **relu** were being confused in Figure 6.4 because of their inability to capture the code interactions. We defined *partial overlap score*

that takes two patient records as follows,

$$\text{condition} = |D_t^1 \cup D_u^2| \geq 3, |M_t^1 \cup M_u^2 \cup P_t^1 \cup P_u^2| \geq 1, |D_t^1 \cap D_u^2| \geq 1$$

$$\text{partial overlap score} = \frac{\#\text{satisfied condition for all } \mathcal{V}_t^1, \mathcal{V}_t^2}{T^1 + T^2}$$

where D_t^1, M_t^1 and P_t^1 respectively denote all diagnosis, medication and procedure codes in the t-th visit of the first patient. T^1 denotes the total number of visits made by the first patient. Given two visits, if one visit has at least two diagnosis codes, at least one of which is shared with another visit, and there are associated medication/procedure codes, then there is a high chance that **linear** and **relu** will be confused since they model a single visit as a flattened set of codes. From the top row of Figure 6.4, we took 10 nearest controls and 10 farthest controls for each of the 125 cases, for both **linear** and **relu**. We also took from the bottom row 10 nearest and farthest cases for each of the 108 controls. The mean partial overlap score between the top row cases and their nearest/farthest controls were 0.58/0.50. The score between the bottom row controls and their nearest/farthest cases were 1.10/0.67. From both results, we could see that **linear** and **relu** were being confused by samples that have higher partial overlap scores than ones with lower scores, confirming that flattening the multi-level structure of EHR data is a suboptimal approach. However, the significant score difference in false positive predictions suggests correctly recognizing controls come in varying degrees of difficulty. And some controls share a large amount of codes with some cases on the surface, making it extra confusing for flat models such as **linear** and **relu**. Further analysis into these shared codes could be an interesting future work.

6.3.7 Analysis on Smaller Data with Short Records

As discussed in section 6.2.4, it is not always the case that researchers or clinicians have a large dataset to conduct experiments on. In this section, we demonstrate the benefit of joint training of `MiME` with auxiliary task when we are given a smaller dataset. Specifically, we use HF prediction as the target task, and $\texttt{MiME}_{bp}$ will be trained with auxiliary tasks. For

Table 6.4: Statistics of the smaller dataset

# of patients	3,973 (272 HF cases)
# of visits	22,415
Avg. # of visits per patient	5.6
Avg. observation period	10 months
# of unique codes	957 (Dx:239, Rx:86, Proc:632)
Avg. # of Dx per visit	2.31 (Max:20)
Avg. # of Rx per diagnosis	0.37 (Max:9)
Avg. # of Proc. per diagnosis	0.43 (Max:8)

Table 6.5: HF prediction performance on a smaller datset

	test loss	test ROC-AUC	test PR-AUC
raw	**0.2225** (0.0251)	0.7572 (0.0380)	0.2920 (0.0825)
linear	0.2227 (0.0195)	0.7436 (0.0270)	0.2986 (0.0569)
relu	0.2400 (0.0261)	0.7403 (0.0254)	0.2766 (0.0550)
relu$_{mlp}$	0.2381 (0.0241)	0.7211 (0.0216)	0.2539 (0.0643)
MiME$_{bp,aux}$	0.2243 (0.0256)	**0.7640** (0.0222)	**0.3236** (0.0683)

comparison, we choose models that showed competitive performance in Table 6.2: **linear**, and all models that use ReLU non-linearity.

From the original dataset, we choose patients with a short sequence (less than 10 visits) for two reasons: 1) shorter sequences make it less likely for GRU to have too big a role in prediction tasks, 2) we want to mimic a real-world setting where a hospital is newly equipped with a EHR system and there aren't much data collected yet. Among the patients with less than 10 visits, we select ones that received many diagnosis codes and the associated medication/procedure codes per visit to emphasize MiME$_{bp}$'s ability to capture the interaction between them. Specifically, we chose patients whose *interaction ratio* was higher than 0.2. The basic statistics of the dataset created as such are described in Table 6.4.

For training the baselines, we used the same hyperparameters as we used in previous experiments. For training MiME$_{bp}$ with auxiliary tasks, we explored various coefficient values for λ_{aux} such as $0.001, 0.01, 0.05, 0.1, 0.5$ and 1, and found 0.05 to provide the best performance, although some of the other values also improved the prediction performance in varying degrees. The results summarized in Table 6.5 show that MiME$_{bp}$ clearly outperforms

baselines in terms of both ROC-AUC and PR-AUC. Although the lowest test loss was achieved by **raw**, its low PR-AUC tells us that it is missing many HF cases and focusing on correctly classifying HF controls, which is secondary compared to correctly identifying the cases. This is understandable given that the smaller dataset has even more severe class imbalance than before, which is corroborated by other baselines also showing significantly lower PR-AUC than $\texttt{MiME}_{bp,aux}$. This experiment demonstrated that `MiME` can robustly handle a very small dataset when jointly trained with the general-purpose auxiliary tasks, which grants an optimistic outlook for `MiME` to be actively adopted by many researchers and clinicians who cannot easily access large EHR datasets.

6.4 Related Work

Over the years, medical concept embedding has been an active research area. Some attempts have been made to develop medical concept embeddings through distributed embedding that summarizes sparse and high-dimensional medical concepts into compressed vector forms. In [55] and [167], medical concepts were taken as tokens and clinical events were organized as temporal sequences, from which medical concept embeddings were derived. On the other hand, some works used latent layers of deep models for representing more abstract medical concepts [9, 10, 13, 14, 155, 168]. For example, [181] formulated a modified restricted RBM to increase interpretability of representation. [182] directly generated patient vectors from raw clinical codes, where raw features were vectorized via a three-layer stacked auto-encoders network, with final hidden layer's weights yielding the patient's corresponding representation. [12] used a multi-layer neural network to incorporate demographics for joint learning of code level and visit level representations. Later, [14] developed a graph-based predictive attention model where weights from a specific layer could be seen as the code embeddings. Although they were all able to successfully learn embeddings for some task in varying degrees, the aforementioned methods did not fully utilize the multi-level structure or heterogeneous inter-code relations in EHR.

Recently, the hierarchical structures and multiple typed codes in EHR aroused more interests. In [183], authors viewed different code types differently, and tried to capture complex relationships across these disparate data types using long-short term memory units, but their model did not explicitly address the hierarchical inter-code relations. More recently in [184], the authors tried to explicitly capture the interaction between a set of all diagnosis codes and a set of all medication codes occurring in a visit. However, in their experiment, simply concatenating both sets to obtain a visit vector outperformed other methods in many tasks. This suggests that disregarding the diagnosis-specific Dx-Rx interaction and flattening all codes as sets is a suboptimal approach to model EHR data.

6.5 Conclusion

In this work, we presented `MiME`, an integrated approach that simultaneously models hierarchical inter-code relations into medical concept embedding, and jointly learns multilevel embeddings using multiple general-purpose prediction tasks. Through extensive empirical evaluation, `MiME` demonstrated impressive performance across all benchmark tasks and its generalization ability to smaller datasets, especially outperforming baselines in terms of PR-AUC in heart failure prediction. We also studied in depth to understand the benefit of `MiME`'s effort to explicitly model the inter-code relations, through visualization and carefully studying the data characteristic. As we have established in this work that `MiME` can be a good choice for modeling patient encounters, in the future, we plan to extend `MiME` to include more fine-grained medical events such as procedure outcomes, demographic information, and medication instructions.

CHAPTER 7
CONCLUSION AND FUTURE WORK

In this thesis, we proposed to develop interpretable deep learning methods for temporal modeling and representation learning on longitudinal electronic health records. We motivated our work by describing the importance of large data and powerful compute for the success of deep learning, and how computational healthcare could benefit from deep learning techniques now that there are sufficient data in healthcare domain as well. We also emphasized the importance of the interpretability for the deep learning models to be readily adopted by real-world clinical practice. We described the past algorithms we developed so far, beginning from a direct application of the blackbox RNN (Dr.AI), then three follow-up works that addressed important aspects of computational healthcare: medical concept representation learning (Med2Vec), code-level interpretability for sequence predictions (RETAIN), and domain knowledge incorporation (GRAM). We also described in the last chapter, the new framework (MiME) to establish a foundation on which we can incorporate more data sources in the future. Our research has consistently shown impressive prediction performance while providing various forms of interpretation, and we believe deep learning can make significant contribution to computational healthcare in general. For future works, we specify the following three potential directions.

7.1 Utilizing heterogeneous data sources

So far our research effort has focused on capturing the hidden relationships among structured medical codes (*e.g.* diagnosis codes, medication codes, procedure codes). However, EHR consists of multiple data modalities such as various lab measures, clinical notes, spectrograms (*e.g.* EEG, brain waves), and demographics. Different data sources can contribute to more accurate modeling of patient status as they can provide more detailed, or

complementary information about the patient.

Therefore as the primary future work, we propose to extend MiME, which accurately models the structure of longitudinal EHR, to handle multiple data modalities. Our primary concern is to cope with varying number of data modalities due to the missing values, and how to derive a reliable patient state from the available data modalities and make accurate predictions. In addition, MiME in its current form does not provide detailed interpretation of its prediction. Therefore addressing the interpretability aspect can be another natural extension of MiME.

7.2 Making predictions with a reinforcement learning agent

There are certain prediction tasks in healthcare for which constructing labeled data is difficult. For example, if we want to predict heart failure onset for a patient as early as possible, typical supervised learning methods will require the training dataset to have labels for both the prediction outcome and the optimal time point to make the prediction. However, we are typically given a training dataset where only the former label is available. In fact, it is not an easy task for even medical experts to determine the optimal time to make the decision.

Reinforcement learning (RL) technique could be a potential solution to this problem. RL allows us to delegate certain decision making processes to the machine, and we only need to define the reward coming from making the decisions properly. For example, if we want to train a neural network agent to make an early HF onset detection, we can set the reward such that, the earlier the correct prediction is made, the bigger the reward. Of course, for an incorrect prediction, the reward should be negative. We propose to combine this approach with RETAIN so that we can make early predictions as well as provide interpretation to the user as to why the agent made such decisions.

7.3 Incorporating additional domain knowledge into GRAM

A rather straightforward, but nonetheless promising future work is to leverage more domain knowledge into deep learning models in the GRAM framework. Injecting a well-known diagnosis hierarchy turned out to improve prediction performance of the deep learning models, and we can imagine that leveraging diverse, high-quality domain knowledge will show dramatic increase in deep learning models' performance in various prediction tasks.

Specifically, we propose to use the code hierarchy for procedure codes such as the Current Procedural Terminology[1], clinical terminology network such as SNOMED-CT[2] or drug interaction networks such as DrugBank [185].

[1] https://www.ama-assn.org/practice-management/cpt-current-procedural-terminology
[2] https://www.snomed.org/snomed-ct

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[4] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3128–3137.

[5] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[6] Y. Wang, K. Ng, R. J. Byrd, J. Hu, S. Ebadollahi, Z. Daar, C. deFilippi, S. R. Steinhubl, and W. F. Stewart, "Early detection of heart failure with varying prediction windows by structured and unstructured data in electronic health records," in *EMBC*, 2015.

[7] J. Sun, J. Hu, D. Luo, M. Markatou, F. Wang, S. Edabollahi, S. E. Steinhubl, Z. Daar, and W. F. Stewart, "Combining knowledge and data driven insights for identifying risk factors using electronic health records," in *AMIA Annual Symposium Proceedings*, American Medical Informatics Association, vol. 2012, 2012, p. 901.

[8] J. Wu, J. Roy, and W. F. Stewart, "Prediction modeling using ehr data: Challenges, strategies, and a comparison of machine learning approaches," *Medical care*, vol. 48, no. 6, S106–S113, 2010.

[9] E. Choi, A. Schuetz, W. F. Stewart, and J. Sun, "Using recurrent neural network models for early detection of heart failure onset," *Journal of the American Medical Informatics Association*, vol. 24, no. 2, pp. 361–370, 2016.

[10] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun, "Doctor ai: Predicting clinical events via recurrent neural networks," in *Machine Learning for Healthcare Conference*, 2016, pp. 301–318.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[12] E. Choi, M. T. Bahadori, E. Searles, C. Coffey, M. Thompson, J. Bost, J. Tejedor-Sojo, and J. Sun, "Multi-layer representation learning for medical concepts," in *SIGKDD*, 2016.

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

[14] E. Choi, M. T. Bahadori, L. Song, W. F. Stewart, and J. Sun, "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, 2017, pp. 787–795.

[15] Z. C. Lipton, "The mythos of model interpretability," *ArXiv preprint arXiv:1606.03490*, 2016.

[16] B. Kim and F. Doshi-Velez, *Interpretable machine learning: The fuss, the concrete and the questions*, Tutorial at ICML 2017, 2017.

[17] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *ArXiv preprint arXiv:1312.6034*, 2013.

[18] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *JMLR*, vol. 9, no. Nov, 2008.

[19] C. Molnar, *Interpretable machine learning: A guide for making black box models explainable*, https://christophm.github.io/interpretable-ml-book, 2018.

[20] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[21] L. S. Shapley, "A value for n-person games," *Contributions to the Theory of Games*, vol. 2, no. 28, pp. 307–317, 1953.

[22] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 1135–1144.

[23] U. Nodelman, C. R. Shelton, and D. Koller, "Continuous time bayesian networks," in *UAI*, Morgan Kaufmann Publishers Inc., 2002, pp. 378–387.

[24] J. M. Lange, R. A. Hubbard, L. Y. Inoue, and V. N. Minin, "A joint model for multi-state disease processes and random informative observation times, with applications to electronic medical records data," *Biometrics*, vol. 71, no. 1, pp. 90–101, 2015.

[25] M. J. Johnson and A. S. Willsky, "Bayesian nonparametric hidden semi-markov models," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 673–701, 2013.

[26] T. J. Liniger, "Multivariate hawkes processes," 2009.

[27] L. Zhu, "Nonlinear hawkes processes," PhD thesis, New York University, 2013.

[28] E. Choi, N. Du, R. Chen, L. Song, and J. Sun, "Constructing disease network and temporal progression model via context-sensitive hawkes process," in *ICDM*, 2015.

[29] A. Graves, "Generating sequences with recurrent neural networks," *ArXiv preprint arXiv:1308.0850*, 2013.

[30] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *ICML*, 2014, pp. 1764–1772.

[31] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014, pp. 3104–3112.

[32] R. Kiros, R. Salakhutdinov, and R. S. Zemel, "Unifying visual-semantic embeddings with multimodal neural language models," *ArXiv preprint arXiv:1411.2539*, 2014.

[33] W. Zaremba and I. Sutskever, "Learning to execute," *ArXiv preprint arXiv:1410.4615*, 2014.

[34] W. Truccolo, U. T. Eden, M. R. Fellows, J. P. Donoghue, and E. N. Brown, "A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects," *Journal of neurophysiology*, vol. 93, no. 2, pp. 1074–1089, 2005.

[35] M. T. Bahadori, Y. Liu, and E. P. Xing, "Fast structure learning in generalized stochastic processes with latent factors," in *KDD*, 2013, pp. 284–292.

[36] R. Ranganath, A. Perotte, N. Elhadad, and D. M. Blei, "The survival filter: Joint survival analysis with a latent time series," in *UAI*, 2015.

[37] Y. Foucher, M. Giral, J.-P. Soulillou, and J.-P. Daures, "A semi-markov model for multistate and interval-censored data with multiple terminal events. application in renal transplantation," *Statistics in medicine*, vol. 26, no. 30, pp. 5381–5393, 2007.

[38] J. Lange, "Latent continuous time markov chains for partially-observed multistate disease processes," PhD thesis, 2014.

[39] Y.-Y. Liu, H. Ishikawa, M. Chen, G. Wollstein, J. S. Schuman, and J. M. Rehg, "Longitudinal modeling of glaucoma progression using 2-dimensional continuous-time hidden markov model," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2013*, 2013, pp. 444–451.

[40] J. Weiss, S. Natarajan, and D. Page, "Multiplicative forests for continuous-time processes," in *Advances in neural information processing systems*, 2012, pp. 458–466.

[41] K. Zhou, H. Zha, and L. Song, "Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes," in *AISTATS*, 2013, pp. 641–649.

[42] S. Linderman and R. Adams, "Discovering latent network structure in point process data," in *ICML*, 2014, pp. 1413–1421.

[43] A. Veen and F. P. Schoenberg, "Estimation of space–time branching process models in seismology using an em–type algorithm," *JASA*, vol. 103, no. 482, pp. 614–624, 2008.

[44] D. Mould, "Models for disease progression: New approaches and uses," *Clinical Pharmacology & Therapeutics*, vol. 92, no. 1, pp. 125–131, 2012.

[45] W. De Winter, J. DeJongh, T. Post, B. Ploeger, R. Urquhart, I. Moules, D. Eckland, and M. Danhof, "A mechanism-based disease progression model for comparison of long-term effects of pioglitazone, metformin and gliclazide on disease processes underlying type 2 diabetes mellitus," *Journal of pharmacokinetics and pharmacodynamics*, vol. 33, no. 3, pp. 313–343, 2006.

[46] K. Ito, S. Ahadieh, B. Corrigan, J. French, T. Fullerton, T. Tensfeldt, A. D. W. Group, *et al.*, "Disease progression meta-analysis model in alzheimer's disease," *Alzheimer's & Dementia*, vol. 6, no. 1, pp. 39–53, 2010.

[47] N. Tangri, L. A. Stevens, J. Griffith, H. Tighiouart, O. Djurdjev, D. Naimark, A. Levin, and A. S. Levey, "A predictive model for progression of chronic kidney disease to kidney failure," *Jama*, vol. 305, no. 15, pp. 1553–1559, 2011.

[48] C. H. Jackson, L. D. Sharples, S. G. Thompson, S. W. Duffy, and E. Couto, "Multistate markov models for disease progression with classification error," *JRSS-D*, 2003.

[49] R. Sukkar, E. Katz, Y. Zhang, D. Raunig, and B. T. Wyman, "Disease progression modeling using hidden markov models," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, 2012, pp. 2845–2848.

[50] J. Zhou, J. Liu, V. A. Narayan, and J. Ye, "Modeling disease progression via fused sparse group lasso," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 1095–1103.

[51] X. Wang, D. Sontag, and F. Wang, "Unsupervised learning of disease progression models," in *KDD*, 2014.

[52] T. A. Lasko, J. C. Denny, and M. A. Levy, "Computational phenotype discovery using unsupervised feature learning over noisy, sparse, and irregular clinical data," *PloS one*, vol. 8, no. 6, e66341, 2013.

[53] Z. Che, D. Kale, W. Li, M. T. Bahadori, and Y. Liu, "Deep computational phenotyping," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2015, pp. 507–516.

[54] N. Y. Hammerla, J. Fisher, P. Andras, L. Rochester, R. Walker, and T. Plötz, "Pd disease state assessment in naturalistic environments using deep learning.," in *AAAI*, 2015, pp. 1742–1748.

[55] Y. Choi, C. I. Chiu, and D. Sontag, "Learning low-dimensional representations of medical concepts," in *AMIA CRI*, 2016.

[56] R. Miotto, L. Li, B. A. Kidd, and J. T. Dudley, "Deep patient: An unsupervised representation to predict the future of patients from the electronic health records," *Scientific Reports*, vol. 6, no. 26094, 2016.

[57] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell, "Learning to diagnose with lstm recurrent neural networks," in *ICLR*, 2016.

[58] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[59] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *PAMI*, 2009.

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

[61] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *ArXiv preprint arXiv:1312.6120*, 2013.

[62] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, *Theano: New features and speed improvements*, Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[63] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *PAMI*, 2013.

[64] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[65] G. Mesnil, Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I. J. Goodfellow, E. Lavoie, X. Muller, G. Desjardins, D. Warde-Farley, *et al.*, "Unsupervised and transfer learning challenge: A deep learning approach.," *ICML Unsupervised and Transfer Learning*, vol. 27, pp. 97–110, 2012.

[66] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," *Unsupervised and Transfer Learning Challenges in Machine Learning*, vol. 7, p. 19, 2012.

[67] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" In *Advances in Neural Information Processing Systems*, 2014, pp. 3320–3328.

[68] J. Hoffman, S. Guadarrama, E. S. Tzeng, R. Hu, J. Donahue, R. Girshick, T. Darrell, and K. Saenko, "Lsda: Large scale detection through adaptation," in *Advances in Neural Information Processing Systems*, 2014, pp. 3536–3544.

[69] V. J. Stevens, C. A. Rouzer, V. M. Monnier, and A. Cerami, "Diabetic cataract formation: Potential role of glycosylation of lens crystallins," *PNAS*, vol. 75, no. 6, pp. 2918–2922, 1978.

[70] N. M. Keith, H. P. Wagener, and N. W. Barker, "Some different types of essential hypertension: Their course and prognosis.," *The American Journal of the Medical Sciences*, vol. 197, no. 3, pp. 332–343, 1939.

[71] J. Kuusisto, K. Koivisto, L Mykkänen, E.-L. Helkala, M. Vanhanen, T Hänninen, K Pyörälä, P. Riekkinen, and M. Laakso, "Essential hypertension and cognitive function. the role of hyperinsulinemia.," *Hypertension*, vol. 22, no. 5, pp. 771–779, 1993.

[72] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[73] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, 2006.

[74] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *ICML*, 2008.

[75] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *JMLR*, 2003.

[76] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model.," in *INTERSPEECH*, 2010.

[77] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," *EMNLP*, 2014.

[78] J. Schluter and S. Bock, "Improved musical onset detection with convolutional neural networks," in *ICASSP*, 2014.

[79] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *EMNLP*, 2013.

[80] R. Kiros, R. Zemel, and R. R. Salakhutdinov, "A multiplicative model for learning distributed text-based attribute representations," in *NIPS*, 2014.

[81] K. P. Murphy, *Machine learning: A probabilistic perspective*. MIT press, 2012.

[82] R. Chen, H. Su, Y. Zhen, M. Khalilia, D. Hirsch, M. Thompson, T. Davis, Y. Peng, S. Lin, J. Tejedor-Sojo, E. Searles, and J. Sun, "Cloud-based predictive modeling system and its application to asthma readmission prediction," in *AMIA*, AMIA, 2015.

[83] J. Sun, C. D. McNaughton, P. Zhang, A. Perer, A. Gkoulalas-Divanis, J. C. Denny, J. Kirby, T. Lasko, A. Saip, and B. A. Malin, "Predicting changes in hypertension control using electronic health records from a chronic disease management program," *JAMIA*, vol. 21, 2014.

[84] J. Sun, F. Wang, J. Hu, and S. Edabollahi, "Supervised patient similarity measure of heterogeneous patient records," *KDD Explorations*, 2012.

[85] M. Ghassemi, T. Naumann, F. Doshi-Velez, N. Brimmer, R. Joshi, A. Rumshisky, and P. Szolovits, "Unfolding physiological state: Mortality modelling in intensive care units," in *KDD*, 2014.

[86] E. Choi, A. Schuetz, W. F. Stewart, and J. Sun, "Medical concept representation learning from electronic health records and its application on heart failure prediction," *ArXiv preprint arXiv:1602.03686*, 2016.

[87] Z. Che, S. Purushotham, R. Khemani, and Y. Liu, "Distilling knowledge from deep networks with applications to healthcare domain," *ArXiv preprint arXiv:1512.03542*, 2015.

[88] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *ICML*, 2008.

[89] A. Mnih and G. E. Hinton, "A scalable hierarchical distributed language model," in *NIPS*, 2009.

[90] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: A simple and general method for semi-supervised learning," in *ACL*, 2010.

[91] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *ArXiv preprint arXiv:1301.3781*, 2013.

[92] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML*, 2014.

[93] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler, "Skip-thought vectors," in *NIPS*, 2015.

[94] J. A. Minarro-Giménez, O. Marín-Alonso, and M. Samwald, "Exploring the application of deep learning techniques on medical text corpora.," *Studies in health technology and informatics*, 2013.

[95] L. De Vine, G. Zuccon, B. Koopman, L. Sitbon, and P. Bruza, "Medical semantic similarity with a neural language model," in *KDD*, 2014.

[96] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, 2009.

[97] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *ICASSP*, 2013.

[98] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *JMLR*, 2010.

[99] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, *et al.*, "Greedy layer-wise training of deep networks," *NIPS*, 2007.

[100] M. D. Zeiler, "Adadelta: An adaptive learning rate method," *ArXiv preprint arXiv:1212.5701*, 2012.

[101] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: A CPU and GPU math expression compiler," in *Proceedings of SciPy*, 2010.

[102] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends® in Machine Learning*, 2009.

[103] L. Cocito, E. Favale, and L. Reni, "Epileptic seizures in cerebral arterial occlusive disease.," *Stroke*, 1982.

[104] J. Forner, "Lung volumes and mechanics of breathing in tetraplegics," *Spinal Cord*, vol. 18, no. 4, pp. 258–266, 1980.

[105] B. Chaudhry, J. Wang, S. Wu, M. Maglione, W. Mojica, E. Roth, S. C. Morton, and P. G. Shekelle, "Systematic review: Impact of health information technology on quality, efficiency, and costs of medical care," *Annals of internal medicine*, vol. 144, no. 10, pp. 742–752, 2006.

[106] A. K. Jha, C. M. DesRoches, E. G. Campbell, K. Donelan, S. R. Rao, T. G. Ferris, A. Shields, S. Rosenbaum, and D. Blumenthal, "Use of electronic health records in us hospitals," *N Engl J Med*, 2009.

[107] A. D. Black, J. Car, C. Pagliari, C. Anandan, K. Cresswell, T. Bokun, B. McKinstry, R. Procter, A. Majeed, and A. Sheikh, "The impact of ehealth on the quality and safety of health care: A systematic overview," *PLoS Med*, vol. 8, no. 1, e1000387, 2011.

[108] C. L. Goldzweig, A. Towfigh, M. Maglione, and P. G. Shekelle, "Costs and benefits of health information technology: New trends from the literature," *Health affairs*, vol. 28, no. 2, w282–w293, 2009.

[109] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission," in *KDD*, 2015.

[110] A. N. Kho *et al.*, "Use of diverse electronic medical record systems to identify genetic risk for type 2 diabetes within a genome-wide association study," *JAMIA*, vol. 19, no. 2, pp. 212–218, 2012.

[111] B. Gallego, S. R. Walter, R. O. Day, A. G. Dunn, V. Sivaraman, N. Shah, C. A. Longhurst, and E. Coiera, "Bringing cohort studies to the bedside: Framework for a 'green button' to support clinical decision-making," *Journal of Comparative Effectiveness Research*, pp. 1–7, 2015.

[112] A. S. Fleisher, B. B. Sowell, C. Taylor, A. C. Gamst, R. C. Petersen, L. J. Thal, and f. t.A.D. C. Study, "Clinical predictors of progression to Alzheimer disease in amnestic mild cognitive impairment," *Neurology*, vol. 68, no. 19, pp. 1588–1595, 2007.

[113] S. Saria, D. Koller, and A. Penn, "Learning individual and population level traits from clinical temporal data," in *NIPS, Predictive Models in Personalized Medicine workshop*, 2010.

[114] P. Schulam and S. Saria, "A probabilistic graphical model for individualizing prognosis in chronic, complex diseases," in *AMIA*, vol. 2015, 2015, p. 143.

[115] C. Esteban, O. Staeck, Y. Yang, and V. Tresp, "Predicting clinical events by combining static and dynamic information using recurrent neural networks," *ArXiv preprint arXiv:1602.02685*, 2016.

[116] J. Ghosh and V. Karamcheti, "Sequence learning with recurrent networks: Analysis of internal representations," in *Aerospace Sensing*, 1992, pp. 449–460.

[117] A. Karpathy, J. Johnson, and F.-F. Li, "Visualizing and understanding recurrent networks," *ArXiv preprint arXiv:1506.02078*, 2015.

[118] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.

[119] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *EMNLP*, 2014.

[120] Q. V. Le, N. Jaitly, and G. E. Hinton, "A simple way to initialize recurrent networks of rectified linear units," *ArXiv preprint arXiv:1504.00941*, 2015.

[121] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," in *ICLR*, 2015.

[122] V. Mnih, N. Heess, A. Graves, *et al.*, "Recurrent models of visual attention," in *NIPS*, 2014.

[123] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra, "Draw: A recurrent neural network for image generation," *ArXiv preprint arXiv:1502.04623*, 2015.

[124] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015.

[125] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *NIPS*, 2015, pp. 1684–1692.

[126] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," in *EMNLP*, 2015.

[127] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NIPS*, 2015, pp. 577–585.

[128] A. F. Martins and R. F. Astudillo, "From softmax to sparsemax: A sparse model of attention and multi-label classification," in *ICML*, 2016.

[129] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *ArXiv:1606.01865*, 2016.

[130] Z. C. Lipton, D. C. Kale, and R. Wetzel, "Modeling missing data in clinical time series with rnns," in *MLHC*, 2016.

[131] P. Nguyen, T. Tran, N. Wickramasinghe, and S. Venkatesh, "Deepr: A convolutional net for medical records," *ArXiv:1607.07519*, 2016.

[132] N. Razavian, J. Marcus, and D. Sontag, "Multi-task prediction of disease onsets from longitudinal lab tests," in *MLHC*, 2016.

[133] M. Q. Stearns, C. Price, K. A. Spackman, and A. Y. Wang, "Snomed clinical terms: Overview of the development process and project status," in *AMIA*, 2001.

[134] H. C. U. Project *et al.*, "Clinical classifications software (ccs) for icd-9-cm," *Rockville, MD: Agency for Healthcare Research and Quality*, 2010.

[135] A. Johnson *et al.*, "Mimic-iii, a freely accessible critical care database," *Scientific Data*, vol. 3, 2016.

[136] A. Goldberger *et al.*, "Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals," *Circulation*, 2000.

[137] R. Vijayakrishnan, S. Steinhubl, K. Ng, J. Sun, R. Byrd, Z. Daar, B. Williams, S. Ebadollahi, W. Stewart, *et al.*, "Prevalence of heart failure signs and symptoms in a large primary care population identified through the use of text and data mining of the electronic health record," *Journal of cardiac failure*, vol. 20, no. 7, 2014.

[138] J. Gurwitz, D. Magid, D. Smith, R. Goldberg, D. McManus, L. Allen, J. Saczynski, M. Thorp, G. Hsu, S. H. Sung, *et al.*, "Contemporary prevalence and correlates of incident heart failure with preserved ejection fraction," *The American journal of medicine*, vol. 126, no. 5, 2013.

[139] T. T. D. Team, "Theano: A python framework for fast computation of mathematical expressions," *ArXiv:1605.02688*, 2016.

[140] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, "End-to-end continuous speech recognition using attention-based recurrent nn: First results," *ArXiv:1412.1602*, 2014.

[141] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, 2014.

[142] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016.

[143] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, 2015.

[144] Z. Yang, W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," *ArXiv:1603.08861*, 2016.

[145] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *ArXiv:1609.02907*, 2016.

[146] G. A. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, vol. 38, no. 11, 1995.

[147] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *SIGMOD*, 2008.

[148] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, 2013.

[149] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *NIPS*, 2013.

[150] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *AAAI*, 2014.

[151] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *AAAI*, 2015.

[152] R. Xie, Z. Liu, and M. Sun, "Representation learning of knowledge graphs with hierarchical types," in *IJCAI*, 2016.

[153] Y. Li, R. Zheng, T. Tian, Z. Hu, R. Iyer, and K. Sycara, "Joint embedding of hierarchical categories and entities for concept categorization and dataless classification," 2016.

[154] K. Q. Weinberger, F. Sha, Q. Zhu, and L. K. Saul, "Graph Laplacian Regularization for Large-Scale Semidefinite Programming," in *NIPS*, 2006.

[155] F. Ma, R. Chitta, J. Zhou, Q. You, T. Sun, and J. Gao, "Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks," in *SIGKDD*, 2017.

[156] J. Futoma, J. Morris, and J. Lucas, "A comparison of models for predicting early hospital readmissions," *JBI*, 2015.

[157] T. Pham, T. Tran, D. Phung, and S. Venkatesh, "Predicting healthcare trajectories from medical records: A deep learning approach.," *Journal of Biomedical Informatics*, 2017.

[158] I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou, "Patient subtyping via time-aware lstm networks," in *SIGKDD*, 2017.

[159] C. Che, C. Xiao, J. Liang, B. Jin, J. Zho, and F. Wang, "An rnn architecture with dynamic temporal matching for personalized predictions of parkinson's disease," in *SIAM on Data Mining*, 2017.

[160] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ArXiv:1409.1556*, 2014.

[161] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[162] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[163] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*, 2013.

[164] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *CVPR*, 2015.

[165] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *CVPR*, 2015.

[166] S. Sukhbaatar, J. Weston, R. Fergus, *et al.*, "End-to-end memory networks," in *NIPS*, 2015.

[167] W. Farhan, Z. Wang, Y. Huang, S. Wang, F. Wang, and X. Jiang, "A predictive model for medical events based on contextual embedding of temporal sequences," *JMIR medical informatics*, 2016.

[168] J. M. Bajor and T. A. Lasko, "Predicting medications from diagnostic codes with recurrent neural networks," in *ICLR*, 2017.

[169] J. L. Elman, "Finding structure in time," *Cognitive Science*, 1990.

[170] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *CVPR*, 2004.

[171] J. Tenenbaum and W. Freeman, "Separating style and content with bilinear models," *Neural Computation*, 2000.

[172] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell, "Compact bilinear pooling," in *CVPR*, 2016.

[173] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach, "Multimodal compact bilinear pooling for visual question answering and visual grounding," in *EMNLP*, 2016.

[174] J.-H. Kim, K.-W. On, W. Lim, J. Kim, J.-W. Ha, and B.-T. Zhang, "Hadamard product for low-rank bilinear pooling," in *ICLR*, 2017.

[175] Q. Suo, F. Ma, G. Canino, J. Gao, A. Zhang, P. Veltri, and A. Gnasso, "A multi-task framework for monitoring health conditions via attention-based recurrent neural networks," in *AMIA*, 2017.

[176] R. Caruana, "Multitask learning," in *Learning to learn*, 1998.

[177] V. L. Roger, S. A. Weston, M. M. Redfield, J. P. Hellermann-Homan, J. Killian, B. P. Yawn, and S. J. Jacobsen, "Trends in heart failure incidence and survival in a community-based population," *JAMA*, 2004.

[178] T. Team, "Tensorflow: A system for large-scale machine learning.," in *OSDI*, 2016.

[179] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ArXiv:1412.6980*, 2014.

[180] T. Saito, T. Rehmsmeier Saito, and M. Rehmsmeier., "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets.," *PLoS ONE*, 2015.

[181] T. Tran, T. D. Nguyen, D. Phung, and S. Venkatesh, "Learning vector representation of medical objects via emr-driven nonnegative restricted boltzmann machines (enrbm)," *Journal of Biomedical Informatics*, 2015.

[182] R. Miotto, L. Li, B. A. Kidd, and J. Dudley, "Deep patient: An unsupervised representation to predict the future of patients from the electronic health records," in *Scientific reports*, 2016.

[183] H. Suresh, N. Hunt, A. Johnson, L. A. Celi, P. Szolovits, and M. Ghassemi, "Clinical intervention prediction and understanding using deep networks," in *MLHC*, 2017.

[184] P. Nguyen, T. Tran, and S. Venkatesh, "Resset: A recurrent model for sequence of sets with applications to electronic medical records," *ArXiv:1802.00948*, 2018.

[185] D. S. Wishart, C. Knox, A. C. Guo, S. Shrivastava, M. Hassanali, P. Stothard, Z. Chang, and J. Woolsey, "Drugbank: A comprehensive resource for in silico drug discovery and exploration," *Nucleic acids research*, vol. 34, no. suppl_1, pp. D668–D672, 2006.