IMPROVING THE PERFORMANCE OF CLINICAL PREDICTION TASKS BY USING STRUCTURED AND UNSTRUCTURED DATA COMBINED WITH A PATIENT NETWORK

by

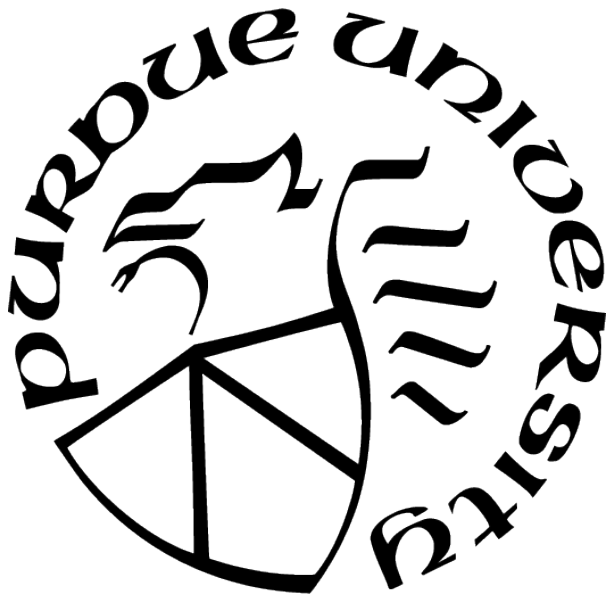
Sara Nouri Golmaei

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science



Department of Electrical and Computer Engineering

Indianapolis, Indiana

August 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Xiao Luo, Co-Chair

Department of Computer and Information Technology

Dr. Brian King, Co-Chair

Department of Electrical and Computer Engineering

Dr. Qingxue Zhang

Department of Electrical and Computer Engineering

Approved by:

Dr. Brian King

Dedicated to my sisters Nadia, Parisa, and Taraneh.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisors, Dr. Luo, Dr. King, and Dr. Zhang, for their help and mentorship. I would also like to thank Sherrie Tucker for keeping me on track since the beginning of the program. Last but not least, I thank my family, who are my main source of motivation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

EHR	Electronic Health Record
NLP	Natural Language Processing
BERT	Bidirectional Encoder Representations from Transformers
GNN	Graph Neural Network
GCN	Graph Convolutional Network
GAT	Graph Attention Network
MIMIC	Medical Information Mart for Intensive Care
LOS	Length of Stay
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
ICU	Intensive Care Unit
CMS	Centers for Medicare and Medicaid Services
ROC	Receiver Operating Characteristic
NER	Named Entity Recognition

ABSTRACT

With the increasing availability of Electronic Health Records (EHRs) and advances in deep learning techniques, developing deep predictive models that use EHR data to solve healthcare problems has gained momentum in recent years. The majority of clinical predictive models benefit from structured data in EHR (e.g., lab measurements and medications). Still, learning clinical outcomes from all possible information sources is one of the main challenges when building predictive models. This work focuses mainly on two sources of information that have been underused by researchers; unstructured data (e.g., clinical notes) and a patient network. We propose a novel hybrid deep learning model, DeepNote-GNN, that integrates clinical notes information and patient network topological structure to improve 30-day hospital readmission prediction. DeepNote-GNN is a robust deep learning framework consisting of two modules: DeepNote and patient network. DeepNote extracts deep representations of clinical notes using a feature aggregation unit on top of a state-of-the-art Natural Language Processing (NLP) technique - BERT. By exploiting these deep representations, a patient network is built, and Graph Neural Network (GNN) is used to train the network for hospital readmission predictions. Performance evaluation on the MIMIC-III dataset demonstrates that DeepNote-GNN achieves superior results compared to the state-of-the-art baselines on the 30-day hospital readmission task. We extensively analyze the DeepNote-GNN model to illustrate the effectiveness and contribution of each component of it. The model analysis shows that patient network has a significant contribution to the overall performance, and DeepNote-GNN is robust and can consistently perform well on the 30-day readmission prediction task. To evaluate the generalization of DeepNote and patient network modules on new prediction tasks, we create a multimodal model and train it on structured and unstructured data of MIMIC-III dataset to predict patient mortality and Length of Stay (LOS). Our proposed multimodal model consists of four components: DeepNote, patient network, DeepTemporal, and score aggregation. While DeepNote keeps its functionality and extracts representations of clinical notes, we build a DeepTemporal module using a fully connected layer stacked on top of a one-layer Gated Recurrent Unit (GRU) to extract the deep representations of temporal signals. Independent to DeepTempo-

ral, we extract feature vectors of temporal signals and use them to build a patient network. Finally, the DeepNote, DeepTemporal, and patient network scores are linearly aggregated to fit the multimodal model on downstream prediction tasks. Our results are very competitive to the baseline model. The multimodal model analysis reveals that unstructured text data better help to estimate predictions than temporal signals. Moreover, there is no limitation in applying a patient network on structured data. In comparison to other modules, the patient network makes a more significant contribution to prediction tasks. We believe that our efforts in this work have opened up a new study area that can be used to enhance the performance of clinical predictive models.

1. INTRODUCTION

The Electronic health record systems (EHRs) hold a patient's data as structured features (e.g., such as lab results and vital sign measurements) and unstructured features (e.g., clinical notes provided by caregivers) [1]. With the EHRs becoming available for researchers, there has been an increasing interest in developing machine learning and, recently, deep learning algorithms that can leverage EHR data to improve clinical outcomes.

In contrast to the structured portions of EHR data, typically used for billing and administrative purposes, healthcare providers primarily use clinical notes for detailed documentation [2]. In other words, clinical notes provide richer information about patients since they describe symptoms, reasons for diagnoses, radiology results, daily activities, and patient illness history [3][4]. Although clinical notes are a valuable source of information, it is challenging to extract patterns from them. They contain highly heterogeneous writing styles, including non-standard terminology or abbreviations [5]. Many studies primarily focused on developing predictive models using structured EHR data features due to clinical notes' sparse and high-dimensional nature [2][6][7].

Despite all these challenges, with the advancement of deep learning algorithms, the interest and necessity in extracting relevant clinical information from clinical notes have increased significantly. 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 [8]. Therefore, it is essential to process available sources, including clinical notes, to feed data-hungry deep learning models.

Bidirectional Encoder Representations from Transformers (BERT) [9] is one of the state-of-the-art Natural Language Processing (NLP) models that can provide contextualized word representations. By pre-training on a large clinical text corpus as a language model, BERT can create context-sensitive word embeddings as features, which will be fed into downstream tasks. Many clinical predictive models incorporate BERT architecture into a downstream task and fine-tune it as an integrated task-specific architecture [3][10][11]. However, since BERT is a deep algorithm with many parameters, it is computationally expensive to fine-tune it on downstream tasks. Especially in clinical problems with a large dataset. As a

solution, we propose a hybrid deep learning model – DeepNote-GNN, which utilizes a pre-trained BERT with a patient network for downstream tasks, such as hospital readmission prediction.

Early hospital readmission prediction facilitates identifying patients at high risk of readmission, which prevents the waste of hospital resources and increases inpatient care quality. To predict hospital readmission, most existing models utilize structured data or require a large amount of manual feature engineering [12][13][14].

DeepNote-GNN uses pre-trained BERT architecture to compute the latent embeddings of clinical notes. Then, a patient network based on note embeddings and the graph topological structure associated with hospital admissions is used to boost 30-day hospital readmission prediction.

In recent years, Graph Neural Networks (GNNs) have been widely used to model a social network as a graph and have shown success when applied to tasks like node classification and social predictions[15][16][17]. Users' common interests (e.g., purchase history) and their neighbors in a social network assist item recommendation and user recommendation or classification. The patient network is modeled similarly as a social network in this research. The similarity and relation between patients' admissions (e.g., medical history, allergies, medication) are valuable sources of information to improve clinical predictions. The patient network also mimics the doctors' clinical decisions on patients based on their experience with similar patients.

In addition to the readmission prediction task, we use DeepNote and patient network modules of DeepNote-GNN to build a novel multimodal model and train it on a combination of structured and unstructured data to make the patient mortality and length of Stay (LOS) predictions. Modifying the architecture of a network as necessary and training it on new prediction tasks or new sets of data is a common strategy in the machine learning community to assess the network's generalization ability.

In addition to DeepNote and patient network, we add two other components, DeepTemporal and score aggregation, to our multimodal model, which helps to boost its performance. The results suggest that structured data can also be utilized to build a patient network. Fur-

thermore, DeepNote-GNN's components have the potential to be installed into other deep learning models.

To the best of our knowledge, this research is the first that leverages clinical notes and temporal signals to create patient networks inspired by social networks. Our contribution in this research can be summarized in five steps:

- DeepNote-GNN takes a computationally efficient feature-based approach and applies it to large clinical datasets without fine-tuning a pre-trained NLP model.
- Our DeepNote representation uses the state-of-the-art language model, BERT [9], pre-trained on medical data [3], with a modification to create richer contextual embeddings associated with admissions.
- DeepNote-GNN builds a patient network using the graph structure to boost the patient outcome predictions by integrating admission note features and topological structure. The patient network takes advantage of similar patient data in the system for the prediction. It improves performance and is computationally efficient.
- We evaluate DeepNote-GNN on a 30-day hospital readmission task using two different subsets of clinical notes in the MIMIC-III dataset. The results show that our model outperforms the baselines by a significant margin.
- We evaluate the generalization ability of DeepNote and patient network on new prediction tasks and new sets of data by installing them into a novel multimodal model. The model is trained on new patient mortality and length of stay prediction tasks using a combination of structured and unstructured data.

2. RELATED WORK

Predicting unplanned hospital readmission has a clinical significance for improving the inpatient care quality while also saving the hospital's resources [18][19]. Traditional machine learning and statistical techniques used to evaluate patient readmission risk utilize various structured data in the EHR, such as patient demographics, medications, and lab tests. Past models generally perform logistic regressions on features obtained by manual feature engineering. A thorough review and comparison of past models can be found in [20][21].

Various deep learning algorithms and data representations have also been used for hospital readmission prediction using structured data. For example, Min et al. [22] used both knowledge-driven and data-driven features extracted from the claims history and the deep and non-deep machine learning models to predict the readmission of Chronic Obstructive Pulmonary Disease (COPD) patients. Wang et al. [23] investigated convolutional neural networks (CNN) for feature extraction and multi-layer perceptron (MLP) to predict hospital readmission using structured data. Barbieri et al. [24] compared several deep learning models with an attention mechanism for readmission prediction using structured data of MIMIC-III. They concluded that data embeddings with neural ordinary differential equations achieved the best result. Ashfaq et al. [25] also applied structured data fusion and utilized LSTM to improve the readmission predictions.

Although clinical notes provide a richer picture of a patient's information collected during their admission, models trained on clinical notes to predict hospital readmission risk have been underdeveloped. Zhang et al. [26] proposed two multi-modal neural network architectures, either Long Short-Term Memory (LSTMs) or Convolutional Neural Networks (CNNs), to enhance patient representation learning by combining sequential unstructured notes with static and structured data. Their model performance on 30-day readmission prediction exceeded the traditional baseline models, such as random forest and logistic regression, that used only structured data. Liu et al. [27] leveraged unstructured clinical notes to generate feature maps using deep learning models based on CNNs and trained them on heart failure 30-day readmission prediction task. Their model showed higher prediction performance than conventional random forest models.

Clinical notes are full of jargon and have an unusual grammatical structure; therefore, it is challenging to build models to learn their representations. Traditional word-level vector representations such as GloVe [28], word2vec [29], and fastText [30], extract each word as a single context-independent vector representation, i.e., they are not able to capture the long-range dependencies in clinical texts. With the advance in natural language processing (NLP), bidirectional language models, such as BERT [9] is able to extract contextual word embeddings. BERT uses the transformer encoder architecture based on a self-attention mechanism and provides contextual word representations based on a masked language model [9].

Although BERT achieves state-of-the-art results on most NLP tasks using almost the same structure across all tasks, it shows poor performance on medical and biomedical corpora [10]. The reason is BERT has been pre-trained on English Wikipedia and BooksCorpus, which vary from clinical and biomedical texts in terms of vocabulary, word distribution, and grammar structure. Several works pre-train BERT on clinical notes [11][3] and biomedical [10] literature to address this problem. The effectiveness of their proposed pre-trained BERT models is evaluated by obtaining good performance on downstream tasks. Huang et al. [3] proposed ClinicalBert by randomly sampling 100,000 notes from the MIMIC-III dataset and pre-training the standard BERT model architecture on them. They follow the BERTbase [9] parameter initialization and hyperparameter setting. ClinicalBERT uses the same pre-training tasks as BERT: masked language modeling and next sentence prediction.

The ClinicalBert [3] original paper takes a fine-tuning approach. It utilizes the pre-trained BERT model with an added fully connected layer on top and fine-tunes it on a 30-day readmission prediction task. Clinical notes associated with a patient's unique admission are split into smaller sub-notes, and the probability on each sub-note is obtained for the 30-day readmission prediction task. To find the prediction outcome at an admission level, Huang et al. [3] combined all probabilities corresponding to the same admission using score aggregation. However, using score aggregation has a computational drawback, especially in clinical datasets with larger data table size, which reduces speed and increases memory requirements. Another issue is degree disparity [31], which has been addressed in ClinicalBert by hand-engineering score aggregation to alleviate the influence of sub-notes that do not contain information about readmission. However, hand-engineered methods reduce

the generalization of the machine learning models. In this research work, we utilize feature aggregation to obtain more powerful representations of clinical notes.

With the rapid growth of EHR data in volume, the interest in applying state-of-the-art deep learning models to EHR data has increased. Graph neural networks (GNNs) are not an exception. GNNs have been widely used in social classification and prediction tasks [15][16][17], but in the clinical setting, they are a young and underdeveloped domain [32].

Research works in this domain primarily focus on utilizing graphs to model medical knowledge or structure the EHR itself (e.g., the relationship between diagnoses and treatments) [33]. However, models that build a patient graph to learn from connections between similar patients are underdeveloped. Malone et al. leveraged patient graph to solve the task of missing data imputation using embedding propagation [34][35]. Rocheteau et al. [36] proposed a hybrid model combining LSTMs for extracting temporal features and Graph GNNs to produce the patient neighborhood information. They exploited diagnoses as relational information by connecting similar patients in a graph.

To the best of our knowledge, our work is the first to improve 30-day hospital readmission prediction by leveraging clinical notes and a patient network. We adopt a feature-based approach to extract powerful representations of clinical notes. These patient representations are then utilized in the patient network to reflect the similarity between patients, which improves the prediction of 30-day readmission.

3. NEURAL NETWORK ARCHITECTURE

The advancement of data collection techniques in recent years has resulted in the availability of a greater volume of data. Large datasets can be used by researchers to build models with more complex architecture than conventional machine learning and statistical models.

The efficiency of traditional machine learning models improves as the amount of data increases until it plateaus. Artificial neural networks are built to overcome this limitation and improve performance as the amount of data fed into them grows. As a result, artificial neural networks are data-hungry complex models with a high generalization ability that can be used instead of classic models to deal with larger datasets. Technically speaking, artificial neural networks are computational processing systems inspired by the way biological nervous systems operate [37]. They consist of multiple neurons connected to pass a message from an input to an output.

A simple version of an artificial neural network is built by a layered organization of neurons, presented in Figure 3.1. In each layer, the weighted sum of multiple inputs is calculated, and activation values are returned, which will be used as inputs for the next layer. The activation functions are necessary to introduce non-linearity to the model. Otherwise, a complex deep neural network is equivalent to a simple linear function. Depending on the model design, different activation functions are available. Neurons in a deep network can have various activation functions, such as ReLU [38], Tanh, and Softmax functions.

A layer with neurons directly connected to neurons in the two neighboring layers, but not to any neurons within them, is called a fully connected layer. The connection between the neurons of consecutive layers is associated with a weight matrix. The initial weights are selected randomly. Each neuron is responsible to pass a message to other neurons that follow. The message is computed based on the neuron's inputs and the weights learned through a feedback mechanism. The final neuron generates the output, which is used in the model's loss function. The goal of a deep neural network is to reduce the loss function. For this purpose, the gradient of loss function with respect to the weights are computed. This process is called backpropagation. Different optimization methods can be used to perform backpropagation, such as stochastic gradient descent and Adam optimizer [39].

Based on the arrangement of neurons and the model architecture design, deep neural networks are categorized into different groups, including Convolutional Neural Networks [37], Recurrent Neural Networks (RNNs) [40], and Graph Neural Networks (GNNs) [41]. In the following sections, we demonstrate each of the aforementioned neural networks, which are used as the building blocks of our model architectures.

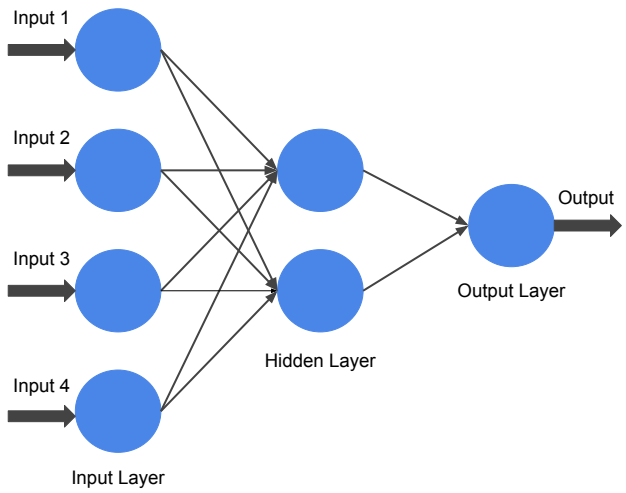


Figure 3.1. A two-layer neural network architecture

3.1 Convolutional Neural Network

One of the most popular deep neural networks is the Convolutional Neural Network (CNN), which takes its name from mathematical linear operation between matrixes called convolution [42]. Convolutional neural networks, also known as ConvNets, have multiple layers, including convolutional layer, non-linearity layer, pooling layer, and fully connected layer. The convolutional and fully connected layers have parameters, but pooling and non-linearity layers do not have parameters.

Compared to feed-forward neural networks, a smaller number of parameters is required to be learned with convolutional layers. Because convolutional layers use kernels, also called filters, as shared weights. Therefore, in a convolutional layer, the number of weights equals the size of the kernel. The convolution operation performed by kernels will be controlled by

several hyperparameters, including the size of the kernel, the padding size, and the size of the stride.

In each layer of a ConvNet, multiple kernels are used to learn various features. Using multiple kernels can add to the dimensionality of deep representations, which can be reduced gradually using pooling layers. Hence, pooling layers help further reduce the number of parameters as well as the computational complexity of the model [37]. Two common pooling approaches used in deep learning models are max pooling and average pooling. Max pooling returns the maximum value from the receptive field, while the average of the values is calculated and returned by an average pooling.

Convolutional layers can be implemented in a variety of deep neural networks. In this study, we do not use convolutional networks directly, but we use Graph Convolutional Networks (GNNs) with built-in convolutional and fully connected layers.

3.2 Graph Neural Network

The underlying relationships among data can be represented using graph structures in several application areas, including computer vision, molecular biology, social recommendation, and pattern recognition. The basic graph structures include single nodes and sequences. However, based on the application, the information can be organized in more complicated graph structures such as trees, acyclic graphs, and cyclic graphs [15].

Deep neural network techniques for graph data have advanced significantly in recent years. These deep neural network architectures are known as Graph Neural Networks (GNNs). GNNs use neural networks to iteratively aggregate feature information from local graph neighborhoods. Meanwhile, node information can be propagated through a graph after transformation and aggregation. As a result, GNNs can learn meaningful representations for graph data by combining node information with topological structure [43].

In this study, we create graphs explicitly to build a patient network, where a patient's admission is represented as a node, and the relationships between several admissions are represented as edges. Therefore, we remodel our clinical prediction tasks into node classification tasks.

In a node classification setup, each node v is represented by its feature x_v and has a label. To predict the labels of unlabeled nodes, the graph neural network learns to improve the representation of each node to a d-dimensional vector h_v which contains the information of its neighborhood [41], demonstrated in Equation 3.1.

$$h_v = f\left(x_v, x_{co[v]}, h_{ne[v]}, x_{ne[v]}\right) \tag{3.1}$$

where $x_{co[v]}$ is the features of the edges connecting with node v. $h_{ne[v]}$ and $x_{ne[v]}$ respectively denote the embedding and features of the neighboring nodes of v. Function f projects these inputs onto a d-dimensional space.

We repeat our experiments using two powerful GNN models: Graph Convolutional Network (GCN) [16] and Graph Attention Network (GAT) [44].

3.3 Graph Convolutional Network

Graph Convolutional Networks (GCNs) are a variation of graph neural networks that can learn from graph data using convolutional layers.

In a GCN network, each graph node receives the features from its neighbors and iteratively aggregates those features using an aggregation function like average. The aggregation function must apply an isotropic aggregation, i.e., each neighbor contributes equally to update the central node representation. Next, the aggregated features are fed into a neural network with convolutional layers. The neural network generates the new vector representation of the node. The whole process is then repeated using the new node features. The math behind the aggregation performed by the GCN layer is demonstrated in Equation 3.2.

$$f\left(H^{(l)}, A\right) = \sigma\left(\widehat{D}^{-\frac{1}{2}}\widehat{A}\widehat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{3.2}$$

where $H^{(l)}$ and $W^{(l)}$ denote the hidden state of the l-th layer and its corresponding weight. A is the adjacency matrix and D is the degree matrix, with $\widehat{A} = A + I$, where I is the identity matrix, and $\widehat{D}$ the diagonal node degree matrix of $\widehat{A}$.

For the above equation, it is important to sum up the identity matrix I with the adjacency matrix A because otherwise, for every node, we sum up all the feature vectors of neighboring nodes but not the node itself. Adding the identity matrix to the adjacency matrix is a technique proposed by [16] to enforce self-loops in the graph. Also, a symmetric normalization is essential to keep the scale of the feature vectors. The scale of the feature vectors can be understood by looking at the eigenvalues of A.

GCNs also benefit from parameter sharing the same as ConvNets. The function responsible for updating the node features in the graph is parametrized the same way across every location in the graph. When it comes to node classification tasks, GCNs show robust performance in many cases. In this study, we employ a graph convolutional network to model the patient network and classify a patient's admission to a specific category.

3.4 Graph Attention Network

The other category of Graph Neural Networks (GNNs) is Graph Attention Networks (GATs) [44]. GATs are neural architecture networks that operate on graph-structured data and use masked self-attentional layers to overcome the drawbacks of Graph Convolutional Networks (GCNs) [44].

In a GAT model, the attention layers are utilized to assign weights to each node's neighbors based on their contributions. Therefore, in a GAT, the edges are weighted based on the relative importance between two nodes that they connect. This process is done in four steps as follows:

- Step 1: The feature vectors of the nodes are transformed linearly, parametrized by a weight matrix, W.
- Step 2: The attention coefficients that determine the relative importance of neighboring features are calculated using Equation 3.3.

$$\mathrm{e_{ij}} = a\left(W\overrightarrow{h}_{\mathrm{i}}, W\overrightarrow{h}_{\mathrm{j}}\right) \tag{3.3}$$

where a is a shared attentional mechanism that captures the attention coefficients, indicating the importance of the neighboring nodes i and j.

- Step 3: Since the number of neighbors varies for different nodes in different graph structures, the attention coefficients are normalized to have a common scaling across all neighborhoods using Equation 3.4.

$$\alpha_{\mathrm{ij}} = \frac{\exp\left(LeakyReLU\left(\mathrm{e}_{\mathrm{ij}}\right)\right)}{\sum_{k \in N} \exp\left(LeakyReLU\left(\mathrm{e}_{\mathrm{ij}}\right)\right)} \tag{3.4}$$

where N is the neighborhood of the node i.

- Step 4: The learned features of the nodes are computed using Equation 3.5.

$$\overrightarrow{h}_{\mathrm{i}} = \sigma\left(\sum_{\mathrm{j} \in N} \alpha_{\mathrm{ij}} W \overrightarrow{h}_{\mathrm{j}}\right) \tag{3.5}$$

where σ is a non-linear activation function.

In the case of employing a multi-head attention mechanism, various attention maps are first calculated. Then all learned representations are aggregated to generate the final output feature for each node, demonstrated in Equation 3.6.

$$\overrightarrow{h}_{\mathrm{i}} = \sigma\left(\frac{1}{K} \sum_{k=1}^{K} \sum_{\mathrm{j} \in N} \alpha_{\mathrm{ij}}^{k} W^{k} \overrightarrow{h}_{\mathrm{j}}\right) \tag{3.6}$$

where K is the number of independent attention maps.

3.5 Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a class of artificial neural networks well-suited to processing variable-length sequential data, such as time-series. RNNs can be implemented in various applications, including text processing, speech recognition, DNA sequences, and more, where the output depends on the previous computations [45].

Since RNNs have a significant representation for keeping the information about the past time steps, they can be applied into clinical time-series data such as lab measurements and vital signs where the order of data affects the outcomes.

RNNs are an extension of a conventional feed-forward neural network with a recurrent hidden state whose activation at each time is dependent on that of the previous time. At each time step, the input at present time and the output produced at previous time affect the available parameter,

$$\begin{aligned} a^{<t>} &= g_1\left(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a\right) \\ y^{<t>} &= g_2\left(W_{ya}a^{<t>} + b_y\right) \end{aligned} \tag{3.7}$$

where $x^{<t>}$, $a^{<t>}$, and $y^{<t>}$ respectively represent the input, activation, and output at time t, with the shared parameters between them noted as W_{ax}, W_{aa}, W_{ya}, b_a, b_y. The activation functions are represented by g_1, g_2.

The main advantages of RNNs are their ability to process the input of any size and produce variable-length outputs. Furthermore, weights are shared across time. As a result of parameter sharing, the model size does not increase as the size of the input increases. On the other hand, deep RNNs are computationally expensive and may suffer from the problem of vanishing or exploding gradients. As deep autoencoders, the gradient in RNNs may become smaller and smaller and make the parameter updates insignificant, which means no real learning is done.

Two variations of RNNs are proposed to solve the vanishing gradient problem: Long Short-Term Memory units (LSTMs) [46] and Gated Recurrent Units (GRUs) [47]. In this study, we repeated our experiments, regarding the mortality and length of stay prediction tasks, by feeding time-series data into both LSTM and GRU models. Despite having a less complex architecture, the gated recurrent unit has shown a better performance. Therefore, we preferred GRU over LSTM in our final model architecture design.

3.6 Gated Recurrent Unit

Recurrent neural networks (RNNs) have a short-term memory problem, which means that for lengthy sequences, they fail to convey a message from earlier time steps to later ones. Technically speaking, this issue is called the vanishing gradient problem. As the gradient back propagates through time, it shrinks, and the weight update becomes insignificant.

Gated Recurrent Units (GRUs) [47] are variations of RNNs that can deal with the vanishing gradient problem encountered by traditional RNNs. A gated recurrent unit modulates information inside the unit using an update gate z and a reset gate r. These are two vectors which decide what information should be passed to the output, shown in Figure 3.2.

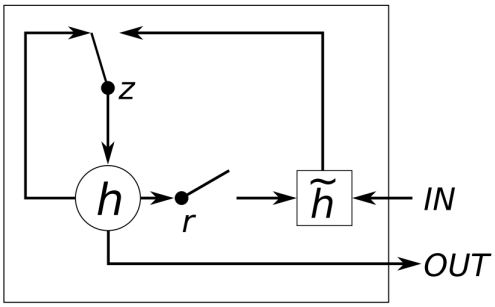


Figure 3.2. Gated Recurrent Unit

The update gate determines how much of the information from the previous time steps needs to be passed along to the future, while the reset gate decides how much of the past information to forget. The mathematical formulation of a GRU model can be represented as

$$\begin{aligned} z_t &= \sigma\left(W_z x_t + U_z h_{t-1} + b_z\right) \\ r_t &= \sigma\left(W_r x_t + U_r h_{t-1} + b_r\right) \\ \widehat{h}_t &= \tanh\left(w_h x_t + r_t \circ U_h h_{\mathrm{i}-t} + b_h\right) \\ h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \widehat{h}_t \\ \text{prediction} &= \text{sigmoid}\left(W_h h_t + b_h\right) \end{aligned} \tag{3.8}$$

where $\circ$ represent the element-wise multiplication. The notations z_t, r_t, $\widehat{h}_t$, h_t respectively are the update gate, the reset gate, the candidate activation unit, and the current activation.

3.7 Bidirectional Encoder Representations from Transformers

To understand the architecture design of DeepNote-GNN, first, we need to know how BERT [9] model works. BERT, which stands for Bidirectional Encoder Representations from Transformers, has been created and published by Jacob Devlin and his colleagues at Google AI language in 2018.

In recent years, BERT model has gained attention and been widely used by the machine learning community for its state-of-the-art results in a wide range of natural language processing (NLP) tasks, including question answering and language inference, while using almost the original structure across the tasks.

Previous language representation models, such as [48], looked at a text input sequentially either from left to right or fused the left-to-right and right-to-left representations trained independently. In contrast, BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers [9]. As a result, compared to unidirectional language models, BERT can capture long-range dependencies of words in a text and produce contextual word representations. In other words, BERT learns a word representation based on all of its surrounding words in a context.

BERT learns text embeddings by utilizing a transformer encoder architecture [49]. The transformer encoder architecture is built on a self-attention mechanism, with two unsupervised tasks used to specify the model's pre-training objective function: masked language modeling and next sentence prediction [3]. For the masked language model, BERT takes in sequences of words as inputs where 15% of words in each sequence has been replaced by a [MASK] token. The model attempts to predict the original value of masked words based on the un-masked context words. For this purpose, a classification layer is added on top of the encoder output. For next sentence prediction, the model receives pairs of sentences as inputs and predicts whether the second sentence is the subsequent sentence in the original document or not. In this regard, the beginning and end of sentences are determined using [CLS] and [SEP] tokens.

The attention mechanism in BERT model increases the interpretability of the results and has a clinical significance. Attention assigns weights to input features based on their

importance to a task. The attention function takes as input a key, a query, and a value vector and computes a distribution over all keys for each query. Next, it multiplies these distribution weights with values. However, in a self-attention mechanism, the input tokens are multiplied with a set of learned weights to construct queries, keys, and values. Equation 3.9 indicates the attention function for a set of queries, keys, and values denoted by Q, K, and V, each with the length of d.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{3.9}$$

An attention weight with a high value signifies that the interaction between the query and key token is predictive about the task at hand. Transformer encoders utilize this attention to simultaneously process every element of a sequence into a sequence of deep embedding representations. BERT is based on 12 stacked transformer encoders [49] with 12 attention heads [50].

For classification purposes, two strategies can be adopted using the BERT model: fine-tuning and feature-based approaches. In a fine-tuning classification method, a classification layer is added on top of the transformer output for the [CLS] token with the output size equal to the number of labels and generally a sigmoid activation function. BERT has shown successful results when trained on the downstream tasks by simply fine-tuning all pre-trained parameters while keeping almost the same architecture across the tasks.

In a feature-based classification approach, pre-trained word representations are extracted as features and are fed into an inexpensive machine learning architecture to make predictions. Consequently, compared to fine-tuning approach, the feature-based methods introduce higher task-specific parameters.

Before explaining how the BERT is applied to medical and biomedical predictive models, it is worth mentioning that researchers primarily benefit from transfer learning technique on the pre-trained BERT model. In the following sections, first, the application of transfer learning in NLP, especially in medical and biomedical fields, is explained. Then a recent BERT model pre-trained on a clinical dataset, ClinicalBERT [3], is introduced.

3.8 BERT-base Transfer Learning

Instead of training a predictive model on a task from scratch using a single dataset, researchers tend to use transfer learning. Transfer learning refers to a set of methods that leverage data from additional domains or tasks to train a model and improve its generalization properties [51].

It is a popular approach in the NLP domain where a pre-trained model is fine-tuned on a downstream task using a new dataset and objective function. Depending on the application, the fine-tuning approach can differ. For training using a small amount of data, researchers generally freeze the earlier layers and learn the rest of parameters. However, for larger datasets, the whole network can be retrained with the pre-trained network weight initialization.

BERT model has been trained on general domain corpora, including English Wikipedia and BooksCorpus. As the state-of-the-art language model pre-training model, BERT, has achieved strong results in many language understanding tasks, researchers utilize a transfer learning technique for downstream NLP tasks. They fine-tune the original pre-trained BERT model on downstream tasks for which the understanding of the workings of the English language is required. The performance improvement reported in various NLP tasks has proven the effectiveness of BERT's contextualized word representations [9].

However, the representations provided by the original BERT [9] cannot achieve high performance in the medical and biomedical domains since BERT is pre-trained on only general domain corpora, which vary from medical and biomedical texts in terminology and grammatical structure.

To address this problem, researchers apply transfer learning technique and pre-train BERT on in-domain text. For example, ClinicalBERT [3] has been developed by pre-training the original BERT model on clinical notes of MIMIC-III dataset and achieved high performance in predicting 30-day hospital readmission prediction. Since ClinicalBERT is the primary baseline of the current research work, it has been fully demonstrated in the next section.

3.9 ClinicalBERT

Clinical notes are full of jargons, abbreviations and have an unusual grammatical structure. Therefore, it is challenging to extract deep representation of clinical notes that can uncover clinical insights. Figure 3.3 shows an example of a clinical note in MIMIC-III dataset.

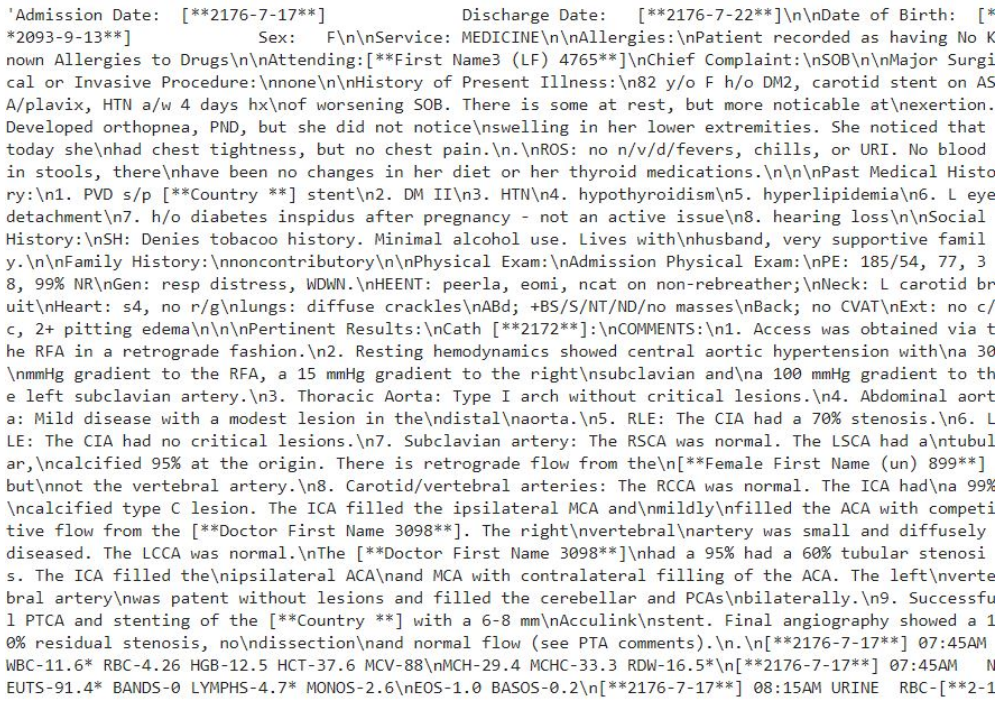


Figure 3.3. A clinical note sample in MIMIC-III dataset

Huang et al. [3] apply BERT model to clinical notes of MIMIC-III dataset so that the pre-trained BERT model, named ClinicalBERT, be able to learn the deep representations of clinical notes. They randomly sample 100,000 notes from the MIMIC-III dataset and pre-train the standard BERT model architecture on them, followed by BERTbase [9] parameter initialization and hyperparameter setting. As a result, ClinicalBERT is able to transform clinical notes into embeddings that can provide interpretable clinical insights.

A clinical note is represented in ClinicalBERT as a set of tokens. In a preprocessing stage, these tokens are subword units extracted from text [52]. A token in a clinical note is computed by adding the token embedding, a learned segment embedding, and a location embedding in ClinicalBert. The segment embedding determines which sequence each token

belongs to. The position embedding of a token is a set of parameters that have been learned to correspond to the token's position in the input sequence. For classification purposes, a classification token [CLS] is inserted at the beginning of each sequence of input tokens [3].

ClinicalBERT utilizes the original BERT architecture and its pre-training tasks, masked language modeling and next sentence prediction, on MIMIC-III clinical notes. The sum of the log-likelihood of the masked tokens and the log-likelihood of the binary variable indicating whether two sentences are consecutive is the pre-training objective function [53].

Besides language modeling tasks, Huang et al. ran an experiment to prove that ClinicalBERT outperforms the original BERT in capturing the relationships between clinical terms as physicians assess. They benchmarked embedding models using the clinical concept dataset in Pedersen et al. [54]. In this regard, they extracted the clinical text embeddings using ClinicalBERT and computed the similarity between two terms using the cosine similarity metric. The Pearson Correlation between the cosine similarity of embeddings learned by BERT and two other language models, Word2Vec [55] and FastText [56], has been computed with the ratings given by physicians. The results presented in [3] show that ClinicalBERT has a high correlation score, i.e., the embeddings created using ClinicalBERT can capture human-rated similarity between clinical concepts [3].

To evaluate the effectiveness of ClinicalBERT representations, Huang et al. adopted a fine-tuning approach and evaluated the ClinicalBERT on 30-day hospital readmission prediction task. For this purpose, a binary classification layer has been added on top of the ClinicalBERT to classify each clinical note deep representation h_{CLS} as readmit (1) or not (0), shown in Figure 3.4. Given a clinical note as input, the probability of readmission is computed using Equation 3.10.

$$P(readmit = 1|h_{CLS}) = \sigma\left(W h_{CLS}\right) \tag{3.10}$$

where W is the weight of the last classification connected layer and σ is a non-linear activation function.

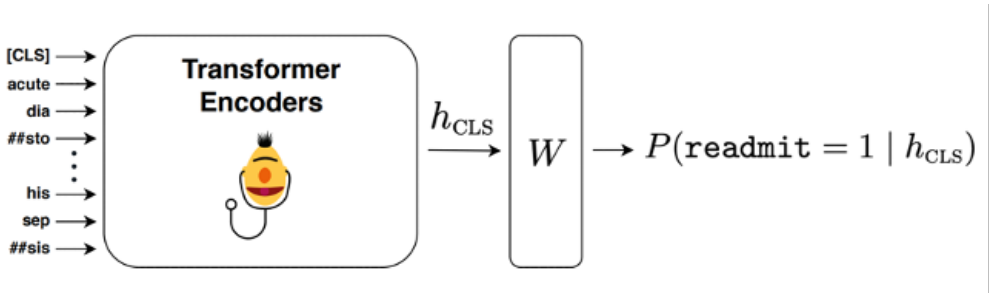


Figure 3.4. ClinicalBert model architecture

To find the prediction outcome at an admission-level, Huang et al. use a score aggregation technique and combine all probabilities corresponding to the same admission using equation 3.11.

$$P(readmit = 1|h_{patient}) = \frac{p^n_{max} + p^n_{mean} * n/c}{1 + n/c} \tag{3.11}$$

where p_{max} and p_{mean} are respectively the maximum and mean probability across n clinical notes belonging to the same admission. c is a scaling factor that has been obtained practically and is equal to 2. Score aggregation improves the prediction results by 3-8%.

However, using the score aggregation has a computational drawback, especially in clinical datasets with larger data table size, which reduces speed and increases memory requirements. Another issue is degree disparity [31], which has been addressed in ClinicalBERT by hand-engineering score aggregation. Note that hand-engineered methods reduce the generalization of the machine learning models.

In DeepNote-GNN, a feature-based approach has been obtained. Also, instead of score aggregation, a feature aggregation has been designed that averages the deep representation of clinical notes corresponding to an admission to compute its deep representation.

4. MODEL DESCRIPTION

We describe the models proposed in this study, DeepNote-GNN and a multimodal model, and their components in the following sections.

4.1 DeepNote-GNN

In this section, we explain how we build a novel hybrid model, DeepNote-GNN, that improves readmission prediction task by leveraging clinical notes and creating a patient network from scratch.

The typical deep language framework used for text classification tasks, including readmission prediction, generally consists of a classification layer on top of a deep language presentation model, shown as Figure 4.1. The language model takes in text data, such as clinical notes, as input and converts them into latent representations, which effectively capture all the important information needed to represent the original data point. This natural language understanding task is popularly known as representation learning.

In the next step, a classification layer takes the learned deep representations and classifies them into the final output classes (readmission or not) with probabilistic scores. The clinical predictive models that are developed based on this typical framework suffer from two main limitations: high computational cost and poor generalization ability.

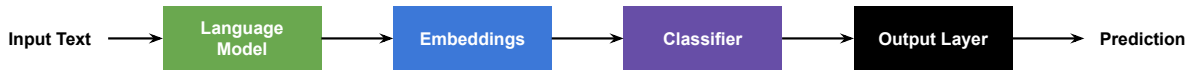


Figure 4.1. Typical deep learning pipeline used in text classification models

Clinical notes have a heavy data pre-processing pipeline and demand greater processing power because they are large in numbers and full of abbreviations, jargon, and unusual grammar structure [3]. Hence, it is computationally expensive to develop heavily engineered task-specific architectures and train them on large clinical note datasets. As a solution, researchers generally employ a fine-tuning approach by selecting an existing pre-trained deep

network and fine-tune it on a downstream task [3][57]. However, the fine-tuning approach introduces minimal task-specific parameters [9].

To address the limitations mentioned above, we develop a novel hybrid model, DeepNote-GNN, that adopts a feature-based approach and evaluates it on a 30-day hospital readmission prediction task. DeepNote-GNN extracts the deep features of clinical notes once and leverages them to build a patient network. The overall architecture of DeepNote-GNN is shown in Figure 4.2. The framework consists of two main modules: DeepNote for obtaining deep clinical note representation and patients network.

DeepNote extracts the context-dependent word representations of clinical notes using the state-of-the-art natural language processing algorithm, BERT, pre-trained on the MIMIC-III medical dataset [9][3], so that it can capture the long-range dependencies of words, which is vital in the clinical setting.

DeepNote-GNN uses a feature-based approach for clinical note representation using a pre-trained BERT model. There are major computational benefits to pre-compute an expensive representation of the training data once and then run many experiments with cheaper task-specific classification models on top of this representation [9].

We built a patient network, then trained it with a Graph Neural Network (GNN) algorithm - Graph Convolutional Networks (GCN). A patient network is modeled as a graph with each node representing a patient's admission using deep note representation and each edge representing a connection between two admissions if they are similar enough based on a similarity measurement.

Our goal is to mimic the hospital team's decision-making process. In addition to a patient's medical information, doctors decide about the patient's medication or treatment procedure based on their experience with other similar patients. In the following sections, we demonstrate each module of DeepNote-GNN in details.

4.1.1 DeepNote Representation

We develop an adaptation over the ClinicalBERT model [3] to extract the deep representations of clinical notes and create an admission feature. DeepNote module is built by

adding an aggregator on top of the ClinicalBERT model to output fixed deep representations of clinical notes at an admission level, shown as Figure 4.2.

To formalize the definition of DeepNote, let $A = \{a^1, a^2, ..., a^n\}$ be the set of admissions in a selected dataset, with n being the total number of admissions. We define $N^{\mathrm{i}} = \{n_1^{\mathrm{i}}, n_2^{\mathrm{i}}, ..., n_{m_{\mathrm{i}}}^{\mathrm{i}}\}$ as the set of clinical notes that exist for a patient's admission i, with m_{i} being the total number of notes for admission i. We concatenate all notes that exist per admission i and create n^{i} as the clinical note of admission i, see Equation 4.1.

$$n^{\mathrm{i}} = n_1^{\mathrm{i}} \,||\, n_2^{\mathrm{i}} \,||\, ... \,||\, n_{m_{\mathrm{i}}}^{\mathrm{i}} \tag{4.1}$$

ClinicalBert shares the same architecture as the BERT model, which only takes in 512 sequence inputs at a time. Therefore we cannot feed long text sequences n^{i} to the model. As a result, we split n^{i}'s into text chunks with a smaller length. Each text chunk is then converted to token sequences with a length of 512, using the BERT tokenizer, as in Equation 4.2.

$$t_{\mathrm{j}}^{\mathrm{i}} = \text{Bert-Tokenizer}(c_{\mathrm{j}}^{\mathrm{i}}) \tag{4.2}$$

where $t_{\mathrm{j}}^{\mathrm{i}}$ is the token sequence associated with text chunk $c_{\mathrm{j}}^{\mathrm{i}}$. $C^{\mathrm{i}} = \{c_1^{\mathrm{i}}, c_2^{\mathrm{i}}, ..., c_{k_{\mathrm{i}}}^{\mathrm{i}}\}$ and $T^{\mathrm{i}} = \{t_1^{\mathrm{i}}, t_2^{\mathrm{i}}, ..., t_{k_{\mathrm{i}}}^{\mathrm{i}}\}$ represent the set of text chunks and token sequences for admission i, with k_{i} being the total number of text chunks for admission i.

The first token of every sequence is the special classification token ([CLS]). The final hidden state of size 768 corresponding to this token is used as the aggregate sequence representation for classification tasks. $F^{\mathrm{i}} = \{f_1^{\mathrm{i}}, f_2^{\mathrm{i}}, ..., f_{k_{\mathrm{i}}}^{\mathrm{i}}\}$ represents the set of 768-value embedding vectors of text chunks for admission i.

DeepNote implements an average feature aggregator on top of the pre-trained BERT model to compute the admission features directly from the text chunk latent representations, shown as Equation 4.3. The performance of a single aggregator operator can be set differently across different datasets [31]. We utilize an average aggregator so that an admission feature becomes an overall picture of all text chunks related to it. Also, an average aggregator

makes all admission features remain on the same scale, which is required toward training the patient network.

$$f^{\mathrm{i}} = \frac{\sum_{\mathrm{j}=1}^{k_{\mathrm{i}}} f_{\mathrm{j}}^{\mathrm{i}}}{k_{\mathrm{i}}} \tag{4.3}$$

Since DeepNote uses the pre-trained BERT model to generate data representations and then feed these fixed representations into computationally efficient classification models, this data representation approach is computationally efficient compared to a model that needs to fine-tune the BERT on the downstream tasks.

DeepNote-GNN utilizes the DeepNote representation to build a patient network. The patient network learns the admission structure alongside the features to predict hospital readmission.

4.1.2 Patient Network

DeepNote-GNN builds a patient network inspired by the social network-based recommendation methods. Social Networks are broadly modeled as Graph Neural Networks (GNNs) [15][16][17], with nodes representing the users and items and edges representing the connections between them. The interaction between a user and its neighbors and the similarity between their preferences are valuable sources of information used in node classification or social prediction tasks.

If the patient network is modeled as a social graph, patients can be modeled as users with various items (e.g., diagnosis, prescriptions, lab tests) connected to each other based on their similarity. Since in the 30-day hospital readmission prediction task, we make predictions at an admission level, in the patient network, we model each patient as a node using the admission information and connect two patients based on their similarity metric. Due to a large number of nodes, it is more computationally feasible to reduce the connections between nodes into a few powerful connections. In this research, we connect admissions with a cosine similarity of more than a threshold and leave the rest of the nodes intact [58].

The patient network is built as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{a^1, a^2, ..., a^n\}$ is the set of n unique admissions as nodes, and $\mathcal{E}$ is the set of edges. The deep clinical note representations

are used as node feature vectors. An edge connects two admissions if the cosine similarity between their features is more than a threshold θ, see Equation 4.4.

$$\mathrm{e_{ij}} = 1 \ \textbf{if} \ \mathrm{similarity}(f^{\mathrm{i}}, f^{\mathrm{j}}) = \frac{f^{\mathrm{i}} \cdot f^{\mathrm{j}}}{||f^{\mathrm{i}}||\,||f^{\mathrm{j}}||} > \theta \tag{4.4}$$

where $\mathrm{e_{ij}}$ represents an edge between admission nodes i and j.

The value of cosine similarity threshold can be adjusted based on the downstream task and the dataset. In DeepNote-GNN, we set θ to be 0.99 after hyperparameter tuning, because the mean similarity between all nodes is high. Moreover, our goal is to connect admissions with high similarity to decrease the rate of false positive predictions.

Many previous graph algorithms can be applied to a patient network. The choice of graph algorithm can depend on the computational resources and downstream tasks. In this research, the patient network utilizes a two-layer Graph Convolutional Network (GCN) [16] for node classification (readmission or not) on a graph with a symmetric adjacency matrix $A \in \mathcal{R}^{N \times N}$ and a degree matrix $D_{\mathrm{ii}} = \sum_{\mathrm{j}} A_{\mathrm{ij}}$. In pre-processing step, the adjacency matrix A is normalized into $\hat{A}$, as in Equation 4.5.

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \tag{4.5}$$

where $\tilde{A} = A + \mathcal{I}_n$ is the adjacency matrix of the undirected graph $\mathcal{G}$ with added self-connections. $\mathcal{I}_n$ is the identity matrix and $\tilde{D}_{\mathrm{ii}} = \sum_{\mathrm{j}} \tilde{A}_{\mathrm{ij}}$.

The GCN forward model computes class scores Z in the form of Equation 4.6.

$$Z = \mathrm{softmax}(\hat{A}\,\mathrm{ReLU}(\hat{A}XW^{(0)})W^{(1)}) \tag{4.6}$$

where $W^{(0)}$ and $W^{(1)}$ are input-to-hidden and hidden-to-output weight matrices respectively. X is a matrix of admission node feature vectors, f^{i}'s.

4.2 Multimodal Model

In order to assess the generalization of a model, it is important to add necessary modifications to the model and run it on new prediction tasks. After measuring the performance of the methods suggested in this study on readmission prediction task using the clinical notes, we design a new set of experiments to analyze model performance on new prediction tasks using unstructured data, structured data, and a combination of them.

The experiments that follow evaluate the methods established in this study on a new set of risk prediction tasks: patient mortality and length of stay. Patient mortality and length of stay can be analyzed in two ways, depending on whether the event occurred in the ICU or not. As a result, in terms of patient mortality, two prediction tasks will be performed, one for in-hospital mortality and the other for in-ICU mortality. For the length of stay, it is important to figure out whether the patient's stay will be long or short. The decision about the length of stay generally depends on the application.

In this research work, two prediction tasks are performed in this regard, such as a three-day short length of stay and a seven-day long length of stay. It is worth noting that length of stay prediction task is a binary classification. For example, when predicting a three-day length of stay, the admission would be categorized depending on whether or not it took longer than three days.

To summarize, the prediction tasks that have been studied are as follows:

- In-hospital mortality: to evaluate if a patient died during a hospital admission after his ICU admission.
- In-ICU mortality: to assess whether a patient died during an ICU admission.
- Three-day length of stay: to classify patients who stayed in the ICU for more than three days.
- Seven-day length of stay: to categorize patients who stayed in the ICU for more than seven days.

As a result, we have to deal with four binary classification tasks. A multimodal architecture has been developed to evaluate the methods proposed in this study on new prediction tasks.

The progress in deep learning has paved the way for more challenging machine-learning problems, which often include several data modalities. The aim is to use the data in a complementary manner to learn a complex task. Multimodal datasets can accomplish this goal since they combine data from several sensors observing the same phenomenon. One of the key benefits of deep learning is that instead of manually designing or handcrafting modality-specific features that are then fed to a machine-learning algorithm, a hierarchical representation can be automatically learned for each modality. To sum it up, multimodality yields a richer representation that could be used to produce much improved performance compared to using only a single modality [59].

To conduct model analysis on new prediction tasks, a data cohort has been selected from the MIMIC-III dataset, which contains both structured data and unstructured data so that the proposed model learns effectively by integrating medical texts and time-series ICU signals of patients. Furthermore, the impact of various data representations has been explored by applying a learned score aggregation to predict outcomes.

The findings indicate that a multimodal architecture can improve the results. Also, implementing a patient network leverages the representations. Consequently, it is a good practice to implement the patient network on deep neural models and generalize it into new prediction tasks.

For the mortality and length of stay prediction tasks, since we use an integration of structured and unstructured data, we cannot use DeepNote-GNN directly. Therefore, we develop a new mutimodal model and apply the DeepNote and patient network modules to it with a slight modification. The details of the model are represented in Figure 4.3.

The proposed model contains four primary modules: DeepNote, DeepTemporal, patient network, and score aggregation. We explain each module and its functionality in mortality and length of stay prediction tasks in the following sections.

4.2.1 DeepNote Representation

DeepNote has the same architecture and details as explained in the DeepNote-GNN literature. It applies the BERT model, pre-trained on random text data of the MIMIC-

III dataset, on clinical notes to generate a deep representation for each note. A feature aggregation is added to create a final admission feature by merging all the clinical note representations associated with that admission. The only difference here lies in the data selection approach.

We only use clinical notes obtained during the first day of a patient's admission for the mortality and length of stay prediction tasks. Consequently, the text chunk density for each admission is low compared to the readmission prediction task. As we average a smaller number of representations, the final admission feature has lower variance and better represents its associated text chunks.

The notes collected on the first day of admission, on the other hand, may offer general information about a patient's condition, such as allergies or status prior to hospitalization, but they lack more detailed information for predicting mortality and length of stay. We assign a weight to control the effectiveness of note features in the final prediction.

Assuming $A = \{a^1, a^2, ..., a^n\}$ is the set of admissions with n being the total number of admissions, and N^{i} refers to the set of notes collected during the first day of admission i. In that case, the output of DeepNote module associated with admission i is represented by f^{i}_{note}, with a size 768. We add a dense layer, with a size of two, on top of the DeepNote module to extract a score vector for note features. The score vector for admission i is represented by s^{i}_{note}. Equation 4.7 demonstrates the math behind the note score extraction.

$$\begin{aligned} f^{\text{i}}_{note} &= \text{DeepNote}(N^{\text{i}}) \\ s^{\text{i}}_{note} &= W_{note} f^{\text{i}}_{note} \end{aligned} \tag{4.7}$$

where W_{note} presents the weights between the note features extracted from the DeepNote module and the fully connected layer of size two.

4.2.2 DeepTemporal Representation

The second module of the proposed model is DeepTemporal, which has taken its name inspired by the DeepNote network. Similar to how DeepNote generates deep representations

for clinical notes, DeepTemporal is used to extract the deep representations of structured time-series data.

Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are two recurrent neural network models that perform well on sequential data, such as temporal samples, and alleviates its vanishing gradient problem. Therefore, we repeated our experiments using each of these networks. Although the LSTM contains an extra memory gate, which complicates its architecture, the GRU outperformed on our data. As a result, we use the gated recurrent unit to capture the deep representations of temporal data in our final model.

To formulate the DeepTemporal network, the input sequence for admission i is represented with T_{i}. We apply a one-layer gated recurrent unit RNN, with a size of 256, to the input sequence T_{i}. The output of the GRU is then flattened and fed into a dense layer with size 512 to create a deep temporal feature for admission i, noted as $f^{\text{i}}_{temporal}$. We add a dropout with a rate of 0.5 after the first fully connected layer to alleviate the overfitting problem. A dense layer of size two is stacked on top of the temporal features to generate a temporal score, represented by $s^{\text{i}}_{temporal}$. Equation 4.8 demonstrates how to compute $s^{\text{i}}_{temporal}$ mathematically.

$$
\begin{aligned}
h^{\text{i}}_{temproal} &= \text{flatten}(\text{GRU}^{(2)}(T^{\text{i}})) \\
f^{\text{i}}_{temporal} &= W^{1}_{temporal} h^{\text{i}}_{temporal} \\
s^{\text{i}}_{temporal} &= W^{2}_{temporal} f^{\text{i}}_{temporal}
\end{aligned}
\tag{4.8}
$$

where $W^{1}_{temporal}$ and $W^{2}_{temporal}$ respectively determine the weights corresponding to dense layers of size 512, and two.

4.2.3 Patient Network

The patient network defined for mortality and length of stay prediction tasks has the same implementation details as explained in the DeepNote-GNN literature, except we use the deep representations of temporal data instead of clinical note feature vectors.

Creating a patient network is challenging because of the number of hyperparameters that we need to tune. Hyperparameters are a set of parameters that cannot be estimated by the model using the given data. Hyperparameter tuning is an essential step in developing robust

machine learning algorithms that allow the model to minimize cost. Manual and automatic hyperparameter tuning are two approaches that can be used in this regard.

The following hyperparameters for a patient network can make a significant difference in the results. We explain each of them and the methods we used to deal with them:

- The node feature vectors: The feature vectors representing each node of the graph could be created using the deep representations of temporal data, clinical notes, or a combination of them. We created a patient network using each of these feature vectors and evaluated them separately. The observations show that the best and worst performance is obtained when the temporal and note features are employed, respectively. As a result, the patient network is built using only the temporal feature vectors.

- The cosine similarity threshold: After computing the cosine similarity between two node feature vectors, we compare its value with a pre-defined threshold. If the similarity between the node representations is greater than the threshold, we are allowed to connect them by an edge in the patient network. Therefore, the cosine similarity threshold is another important hyperparameter in building a patient graph structure. To find the best threshold, first, we chose pairs of node features randomly and calculated the similarity between them. The results showed that the similarity distribution between the feature vectors is roughly between 30% to 41%. After examining different values for the threshold, we selected a threshold of 40%. That is, an edge connects two nodes if the cosine similarity between their feature vectors is more than 0.4. In medical problems, it is critical to choose a high threshold value to eliminate redundant edges. Otherwise, we connect unrelated patients and make decisions regarding them based on inaccurate relationships and information. The patient network of multimodal model contains 22,973 nodes connected with 11,836,509 edges.

First, we used the temporal features of size 512 produced by DeepTemporal as node feature vectors, but the patient network has made poor classification predictions on them. One reason could lie in the different training speeds for DeepTemporal and the patient

network. In other words, we cannot train the GRU perfectly using the gradients of both the DeepTemporal network and the patient network. Therefore, we need to obtain the deep representations for each of these modules separately.

We feed the temporal input associated with the admission i, noted as $T^{\mathrm{i}}_{temporal}$, to a GRU recurrent network of size 256, different from what we had for the DeepTemporal, and flatten its output. The result is used as a feature vector for admission i used in patient network, represented by $f^{\mathrm{i}}_{network}$.

We build a patient network by using these node feature vectors in a Graph Convolutional Network (GCN) structure. Finally, a dense layer of size two is added to the end of the GCN to get the corresponding patient network score, $s^{\mathrm{i}}_{network}$. Equation 4.9 demonstrates the whole process mathematically.

$$\begin{aligned} f^{\mathrm{i}}_{network} &= \mathrm{flatten}(\mathrm{GRU}^{(2)}(T^{\mathrm{i}})) \\ s^{\mathrm{i}}_{network} &= W_{network}(\mathbf{GCN}(f^{\mathrm{i}}_{network})) \end{aligned} \tag{4.9}$$

where $W_{network}$ is the weight matrix corresponding to the dense layer with size two. To make a difference between the GRU models of DeepTemporal and patient network, we represent them by $\mathrm{GRU}^{(1)}$ and $\mathrm{GRU}^{(2)}$ respectively.

4.2.4 Score Aggregation

The last building block of our proposed multimodal model is a score aggregation module. The score aggregation turns each of three scores on or off, to measure the effectiveness of the other ones. We also assign different weight to each of three scores and aggregate them linearly to output a final score for admission i, defined as s^{i}. The prediction is made based on the final score, described in Equation 4.10.

$$s^{\mathrm{i}} = w_{note} * s^{\mathrm{i}}_{note} + w_{temporal} * s^{\mathrm{i}}_{temporal} + w_{network} * s^{\mathrm{i}}_{network} \tag{4.10}$$

where w_{note}, $w_{temporal}$, $w_{network}$ are respectively the weights between zero and one associated with the note, temporal, and patient network scores, with $w_{note} + w_{temporal} + w_{network} = 1$.

Note that the weights mentioned above are hyperparameters fixed during the model training, and we specify them by running an automated grid search hyperparameter tuning, with a window size of 0.1. Since we have three metrics for model evalutation, it is not feasible to define these weights as model parameters. However, it is possible to adjust the model in the future so that it can learn these weights from the data by minimizing a loss function of metrics. This function mainly depends on the application.

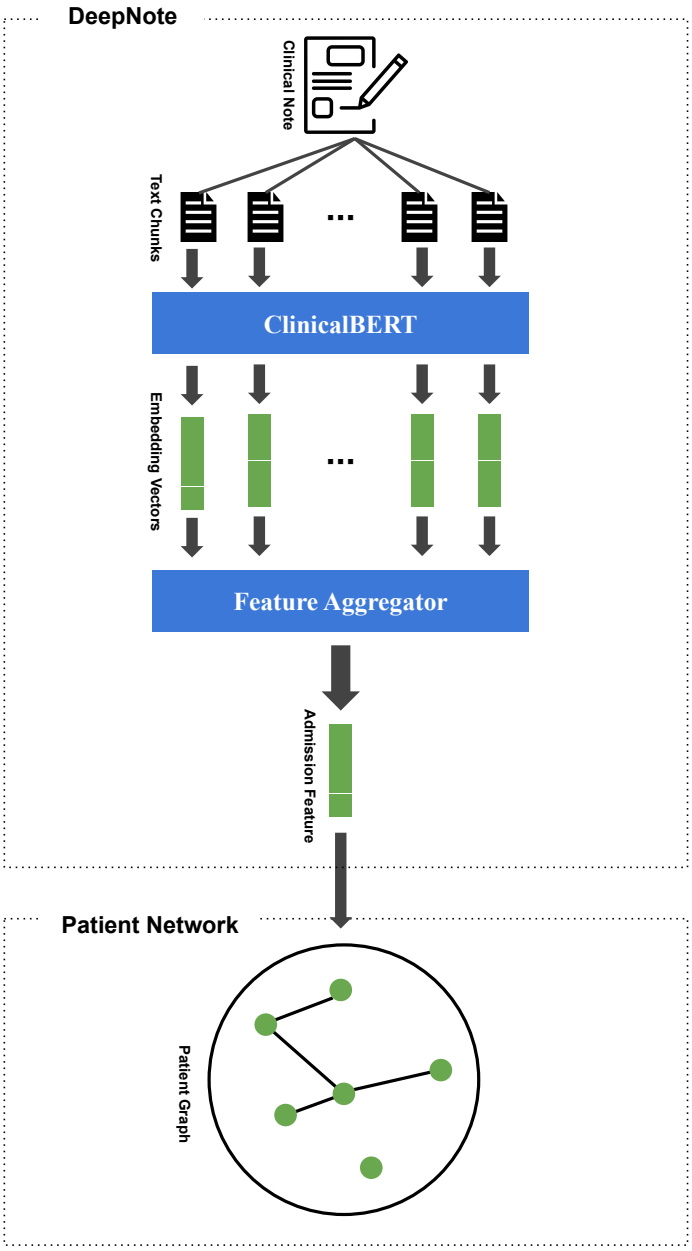


Figure 4.2. An overview of the DeepNote-GNN model architecture

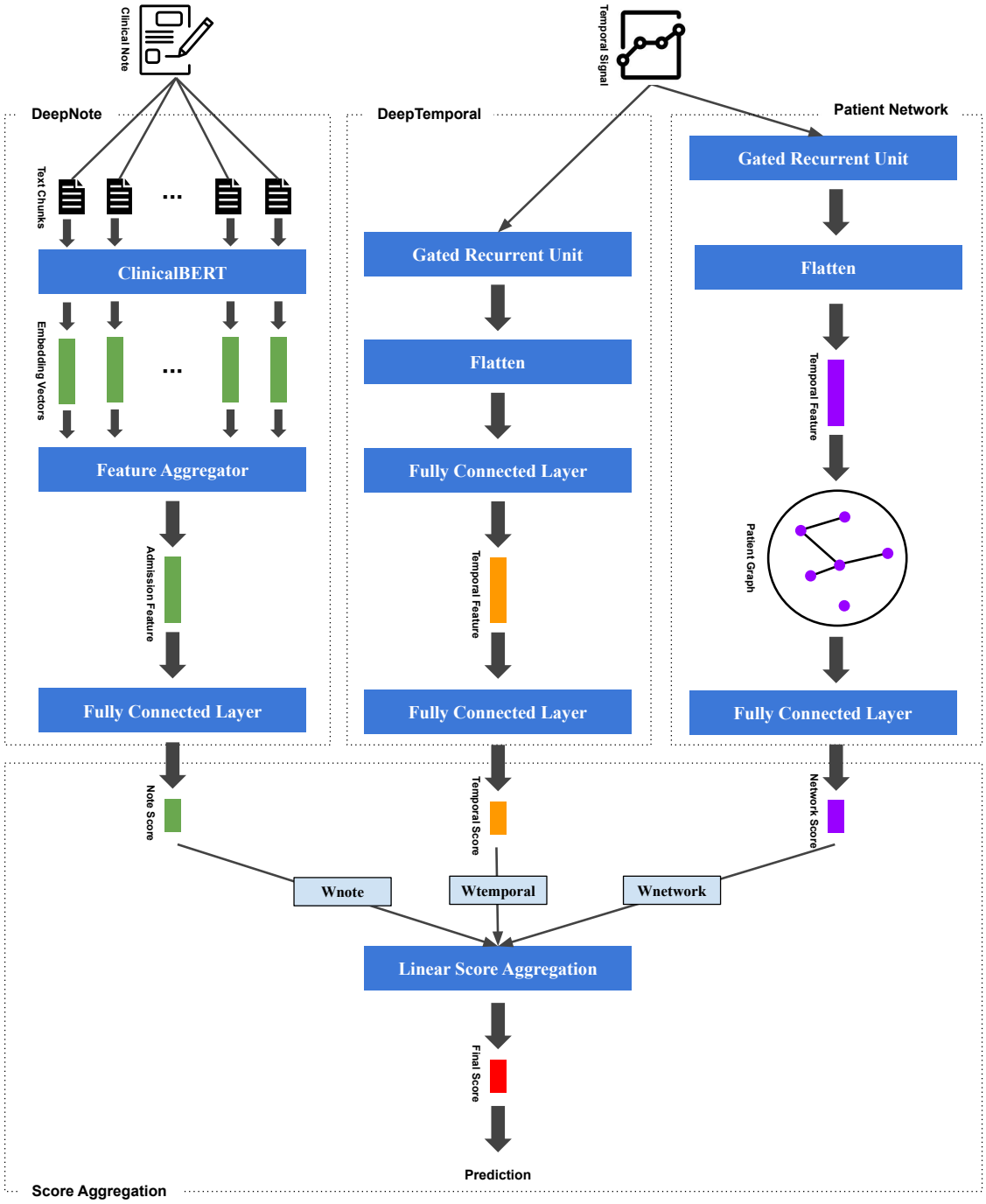


Figure 4.3. An overview of the multimodal model architecture

5. EXPERIMENTAL SETUP

5.1 Prediction Task

In this research work, we train our proposed models on three prediction tasks with high clinical significance. In the following section, we demonstrate each prediction task and its importance.

5.1.1 Patient Hospital Readmission

Hospital readmissions are described as admissions to the hospital within a typically short period after being discharged from the hospital. Readmissions are common and costly. Not only do they waste hospital money and limited resources, but they also bring patients and healthcare facilities under a great deal of stress [60][61].

Readmissions to hospitals are becoming a major concern for hospitals and policymakers. Many organizations have adopted the readmission rate as a metric to measure the quality of inpatient care provided [62]. Studies reveal that higher hospital-level patient satisfaction scores are independently associated with lower 30-day hospital readmission rates [63].

Centers for Medicare and Medicaid Services (CMS) in the USA has imposed financial penalties to hospitals with high readmission rates by reducing the payment for patients readmitted within 30-day of discharge. Therefore, there is a growing interest within the research community to address this problem from a data analysis perspective [64].

In recent years, compared to traditional statistical models such as random forest or logistic regression, deep neural networks have shown a higher performance in predicting hospital readmission. Barbieri et al. [24] benchmarked a number of deep learning models for predicting 30-day Intensive Care Unit (ICU) readmission using MIMIC-III dataset. Recurrent neural networks such as Long Short-Term Memory (LSTM) and Gated recurrent units (GRUs), Convolutional Neural Networks (CNNs), and attention-based models are examples of the deep learning models that have been used for readmission prediction task. Among them, the attention-based models are the most popular because the interpretability of the

attention-based models can be used to describe patients who are at risk. Also, it can help decision-makers to trust the predictive model's predictions and improve patient treatment.

It is worth mentioning that deep clinical models that predict readmission are data-sensitive. Readmission rates and unnecessary readmissions vary depending on data collection methodology [65].

5.1.2 Patient Mortality

One of the outcomes of interest of hospital admission is mortality prediction. There are three types of mortality predictions that researchers are interested in: in-hospital, short-term, and long-term mortality predictions.

In-hospital mortality predictive models predict whether the patient dies after being admitted to the Intensive Care Unit (ICU) of the hospital. Short-term mortality prediction predicts whether the patient lost his life a short period of time, typically one to three days, after his ICU admission. Finally, long-term mortality prediction estimates whether the patient lost his life a long time after his admission, i.e., at least 30 days after discharge [66].

Early prediction of hospital mortality can help caregivers make efficient medical decisions about the severely ill patients staying in intensive care units [67]. In addition to that, it aids them in determining the value of novel treatments, quality of inpatient services, and health care policies.

Mortality prediction task can be formulated as a binary classification task, with the label indicating the death event for a patient. Compared to the readmission prediction task, traditional statistical models have been more successful in predicting hospital mortality. One reason lies in the data nature that has generally been used for mortality prediction tasks, which gives the models a rich picture of a patient's information. For example, vital signs and lab measurements are commonly utilized, which are considered by physicians as a qualified indicator of a patient's life or death status. Still, in comparison with the conventional logistic regression models, the deep neural network models are more accurate in predicting in-hospital mortality and have higher overall performance and generalization indices [68].

The mortality of patients, in hospital and after discharge, in the MIMIC-III database is 23.2%. Data analysis shows that older people are more likely to be admitted to the ICU and had a higher risk of death [69].

5.1.3 Patient Length of Stay

Length of stay refers to the duration of a unique patient's admission to the hospital or intensive care unit (ICU). Accurately predicting hospital length of stay can benefit hospital management, resource planning, and reduce costs.

Length of stay explains 85% to 90% of the variation in hospital costs between patients. [70] For hospitals, length of stay prediction can help to optimize the usage of wards and beds to best provide care. For caregivers, it can offer adjunctive clinical decision support. Additionally, patients can better manage their expenses by estimating the duration of their stay in the hospital [71].

Clinical predictive models are typically trained to determine whether a patient's stay will be long or short, with a stay of more than five days being considered long. As a result, they deal with a binary classification problem. A challenge for such predictive models is that length of stay is influenced by a variety of factors, with complex relationships between them. To overcome this challenge, neural network-based approaches have been adopted. Neural networks are prominent tools for analyzing big datasets with data that comes from different modalities [71].

Among models developed on MIMIC-III dataset, although neural network-based models have a higher performance in predicting length of stay, the gap between their results and the results obtained from the conventional statistical models is not significant.

5.2 Dataset

In this section, we explain two types of data, structured and unstructured data. We also provide a detailed demonstration of the MIMIC-III dataset as we use its structured and unstructured data to run our experiments. Finally, the data selection pipelines used for each prediction task are illustrated.

5.2.1 Structured Data

Structured data is often numerical and easy to analyze. It can be organized into pre-defined structure formats, for example, rows in a Structured Query Language (SQL) database table. Structured data can be managed by technology that allows for querying and reporting against predetermined data types and understood relationships, such as a Relational Database Management System (RDBMS) [72].

Researchers are implementing solutions to bring together unstructured and structured datasets to enable machine learning algorithms to understand the task on hand better. Data fusion approaches have received significant attention in recent years, particularly in medical settings, where different data types are complementary. Blood pressure, lab measurements, and respiratory rate are examples of structured data that are all part of the MIMIC-III dataset.

5.2.2 Unstructured Data

Unstructured data refers to datasets that are not stored in a structured database format. Although unstructured data has an internal structure, it is not pre-defined through data models. There are two basic categories of unstructured data [72]:

- Bitmap Objects: Inherently non-language based, such as image, video, or audio files.
- Textual Objects: Based on a written or printed language, such as Microsoft Word documents, e-mails, and clinical notes.

Unstructured data can be generated by humans or machines. An example of unstructured data generated by machines is satellite imagery data. For the MIMIC-III dataset, both types of unstructured data exist. For example, the patient's chest X-Ray images provided by devices and clinical notes, which we use in this study, written by caregivers. Developing tools and models to manipulate and process unstructured data are a more challenging task compared to structured data.

5.2.3 MIMIC-III Dataset

An Electronic Health Record (EHR) data is described as a repository of patient data in digital form, stored and shared securely, and can be accessed by multiple authorized users [73].

The amount of digital information contained in Electronic Health Records (EHRs) has increased dramatically over the last decade. Although these EHR data were initially intended for archiving patient information and conducting administrative healthcare tasks such as billing, several researchers have discovered that they can also be used for clinical informatics applications and epidemiology [2][74][75].

The early analysis of EHR data relied on traditional statistical techniques [76]. Although these techniques are interpretable and straightforward, they have a number of issues. For instance, they require hand-engineered feature design or cannot deal with high-dimensional inputs [77]. With the recent developments in machine learning and deep learning techniques, their use in building clinical predictive models has been increased.

Although deep learning methods are able to alleviate the issues raised by traditional statistical techniques, their performance is limited by the amount of clinical data they process. As a result, the collection and pre-processing of EHR data is important to improve the quality of predictions in clinical models. For this research work, we employ today's most popular public EHR database, MIMIC-III dataset.

MIMIC-III, or Medical Information Mart for Intensive Care, is a massive, single-center database that contains data on patients admitted to critical care units at Beth Israel Deaconess Medical Center in Boston, Massachusetts. MIMIC-III is the only freely accessible critical care database with comprehensive information about individual patient care [78]. An overview of MIMIC-III database is represented in Figure 5.1.

MIMIC-III consists of the data for 53,423 distinct hospital admissions of adult patients that have been admitted to the hospital between 2001 and 2012. The data includes patient demographic information such as age, gender, marital status, structured temporal data such as lab measurements and vital signs, and unstructured data, including discharge summaries and clinical notes.

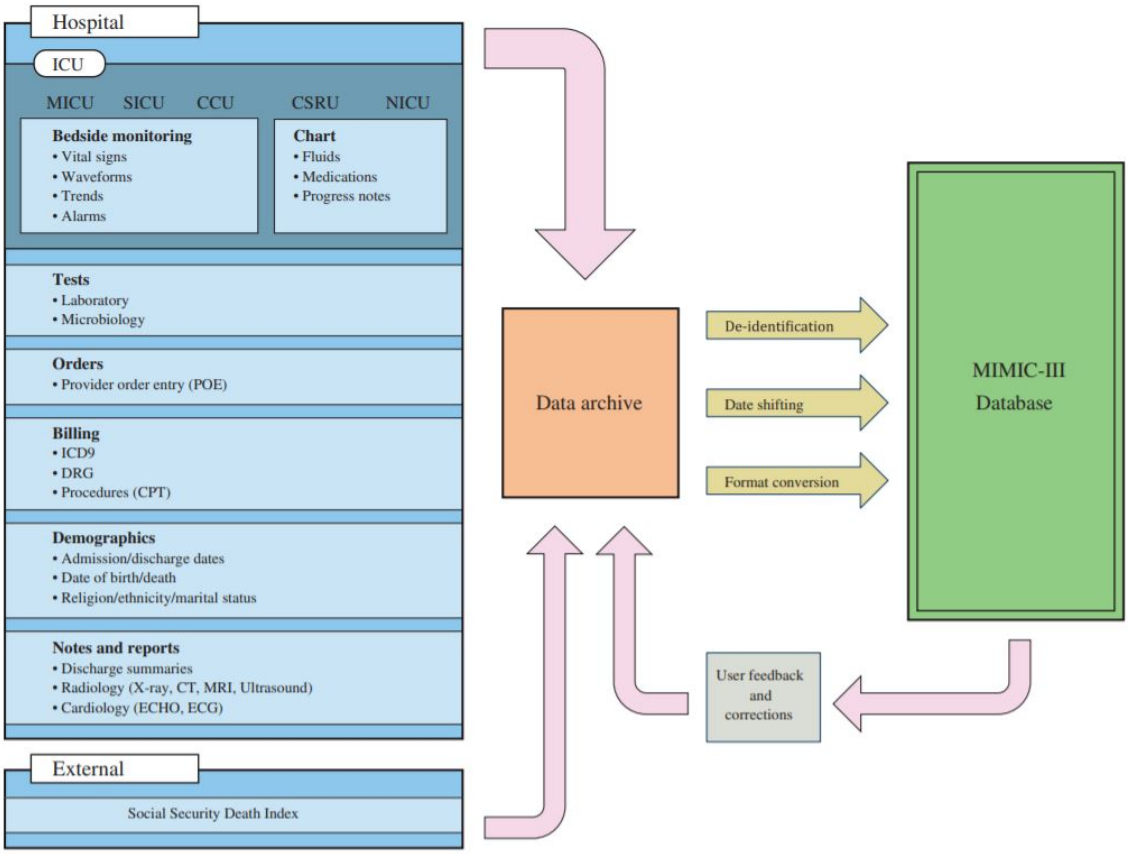


Figure 5.1. An overview of the MIMIC-III critical care database

The MIMIC-III database is a relational database of 26 tables. Tables are related by identifiers, which are normally suffixed with the letter 'ID'. SUBJECT_ID, for example, applies to a single patient, HADM_ID, to a unique hospital admission, and ICUSTAY_ID, to a unique intensive care unit admission. The following tables from the MIMIC-III dataset, as well as their corresponding data columns, are used in the experiments conducted in this study.

ADMISSIONS Table: the admissions table contains details about a patient's hospitalization. Data available includes admission and discharge dates, demographic data, and the source of admission. The columns of data that have been used form ADMISSIONS table are:

- SUBJECT_ID: a unique ID for identifying a patient admitted to the hospital.

- HADM_ID: a unique ID that represents a single patient's admission to the hospital.

The admission ID and subject ID are used as the primary and secondary keys to connect several tables.

- ADMITTIME: gives the date and time of the patient's hospital admission.
- DISCHTIME: gives the data and time of the patient's discharge from the hospital.
- DEATHTIME: gives the data and time of the patient's death.

The data extracted from the time columns are used to create labels. For example, for 30-day readmission prediction, the gap between two consecutive admissions of a patient is calculated, and if this gap is less than 30 days, the readmit label is set to one.

- ADMISSION_TYPE: describes the type of the admission.

There are four types of admission in MIMIC-III dataset including elective, emergency, urgent, and newborn. The newborn admissions are discarded from all experiments because the data distribution for newborn admissions is entirely different from that of other admission types.

NOTEEVENTS Table: this table contains all clinical notes for patients during their hospital admissions. Two important columns of NOTEEVENTS dataset are as follows:

- CATEGORY: the type of the note that is given to a patient.

In this study, we use three different types of notes listed as discharge summary, notes given by nurses, and notes provided by physicians. Discharge summaries contain a comprehensive information about a patient's admission and are used for billing purposes. Therefore, the performance of models trained on discharge summaries are higher.

- TEXT: it contains notes provided by caregivers throughout a patient's hospitalization.

Clinical texts have an expensive computational preprocessing. For this research project, clinical notes are split based on their category before being fed into models. Discharge summaries, for instance, are fed to DeepNote-GNN separately.

ICUSTAYS: the table contains the information of each ICU stay.

- ICUSTAY_ID: a unique ID for a patient's stay at ICU.
- LOS: The length of stay for a given ICU stay.

In this research, the labels corresponding to a three-day short stay and a seven-day long stay have been generated using LOS data column.

- INTIME: date and time recorded when the patient has been transferred into the ICU.
- OUTTIME: date and time recorded when the patient has been moved out of the ICU.

INTIME and OUTTIME are used to compute a patient's length of stay at ICU. The value obtained can help to filter out the admissions that have spent in the ICU for less than 24 hours.

CHARTEVENTS: it contains all charted data for all patients, such as patients' vital signs. From this table, structured temporal data has been extracted. Examples of structured data in CHARTEVENTS table are heart rate, systolic blood pressure, arterial blood pressure, respiratory rate, temperature, and more.

LABEVENTS: it contains all laboratory measurements for all patients. Examples of structured data obtained from LABEVENTS table include anion gap, albumin, bands, bicarbonate, bilirubin, creatinine, and more.

In this study, an open-source pipeline, MIMIC-Extract [79], is used to transform MIMIC-III time-series data into directly usable features. Their pipeline converts raw vital sign and laboratory data into multivariate time-series before performing preprocessing steps like unit conversion, outlier handling.

5.2.4 Data Selection Pipeline

Readmission Prediction Task

We follow the data selection and preparation pipeline proposed by [3] for a fair comparison with baseline models. The study cohort in this research is a subset of the MIMIC-III, which contains 34,560 patients with 2,963 positive 30-day readmission labels and 48,150 negative labels. It is worth noting that DeepNote-GNN is not trained with all clinical notes that exist per admission in our study cohort. We split the clinical notes into two subsets and

downsample negative labels of each subset to address the unbalanced ratio of positive and negative labels.

We experiment with the readmission prediction based on two different clinical note sets as follows:

- $D_{discharge}$: This clinical note set contains discharge summaries. A discharge summary is a written summary of a patient's status during an admission. It includes valuable information such as reasons for hospitalization, diagnosis, a list of prescribed medications, and even a radiology report. The discharge summaries set contains 6162 unique admissions. We split the discharge summary associated with an admission into text chunks of size 318. A discharge summary has an average of 5 text chunks.
- D_{3days}: This clinical note set contains notes except discharge summaries from the first three days of admission. Clinical notes other than discharge summaries are more comprehensive as they address a specific issue or service. D_{3days} contains 6261 admissions, where an admission has an average of 9 text chunks of size 318.

We evaluate DeepNote-GNN on each dataset separately. The discharge summaries are an overview of a patient's stay that physicians can use in patient's future visits. Making accurate readmission predictions using discharge summaries can feedback the post-hospital team to modify their processes if needed. On the other hand, predicting readmission using D_{3days} allows patient treatments to be adjusted dynamically while the patient is still in the hospital. Statistics of MIMIC-III sub-cohort and selected subsets of data are summarized in Table 5.1.

Table 5.1. Summary statistics of data used in this study.

Data	Num of Patients	Num of Admissions	Avg Chunks Num
Study Cohort	34,560	45,321	4.2
$D_{discharge}$	5,775	6,162	5.3
D_{3days}	5,770	6,261	9.5

The patient network built on $D_{discharge}$ and D_{3days} contain 6,162 and 6261 admission nodes, and 3,667,733 and 5,496,127 edges respectively. It is not out of the ordinary to have hundreds of thousands of nodes and millions of edges in a graph. Graph Neural Networks (GNNs) have proven effective in dealing with large-scale graph data [58].

Mortality and Length of Stay Prediction Tasks

To perform a fair comparison, we follow the data selection and preparation pipeline suggested by [80] to select note and time-series data from the MIMIC-III dataset. In this regard, MIMIC-Extract, an open-source data extraction pipeline, has been utilized. With certain patient inclusion criteria, MIMIC-Extract focuses primarily on the patient's first ICU visit. They discard data from patients under the age of 15 who have a length of stay of less than 12 hours and more than 10 days. This pipeline produces a cohort of 34,472 patients and 104 clinically aggregated time-series variables [80]. It is worth noting that only the first 24 hours of a patient's data after an ICU admission were included in this study.

As a result, the experiments ran on mortality and length of stay prediction tasks add a new type of data to model evaluation and complement our clinical note data selection efforts. Remember that, in addition to discharge summaries, we evaluated DeepNote-GNN on clinical notes collected during the first three days of an admission. In the following analyses, we assess our methods on early clinical notes for an admission, gathered during the first 24 hours after a patient has been admitted to the ICU.

For mortality and length of stay prediction, it is critical to remove discharge summaries from the final cohort to avoid information leak. Discharge summaries include all information related to a patient that can be used for medical history documentation or billing purposes, including mortality and the number of days patient has hospitalized in the ICU.

Furthermore, clinical notes without a recorded chart time have been discarded. Chart time data is used to generate the corresponding length of stay labels. By subtracting the admission time from the discharge time, the length of stay will be calculated. For a three-day length of stay prediction, if this value is greater than three days, the label is equal to one; otherwise, it equals zero.

For each prediction task, the data split follows 70%, 10%, 20% ratio for respectively train, validation, and test sets. A critical characteristic of the selected dataset is its imbalanced label distribution, where the distribution of examples across positive and negative classes is biased. The degree of class imbalance varies, but a severe imbalance is more challenging to model and may demand specialized techniques. Imbalanced class distribution is very common in real-world classification problems, such as fraud detection, spam detection, or in our case, mortality and long-term length of stay predictions.

A summary of label distribution for each prediction task has been reported in Table 5.2. Among all prediction tasks, the dataset has the highest and slightest imbalanced positive and negative label distribution for in-ICU mortality and three-day length of stay tasks, respectively.

Table 5.2. A summary of positive label ratio for each prediction task

Prediction Task	Train Set	Validation Set	Test Set
In-hospital Mortality	10.38%	9.41%	10.56%
In-ICU Mortality	6.82%	6.35%	7.08%
Three-day Length of Stay (LOS> 3)	43.55%	42.25%	42.62%
Seven-day Length of Stay (LOS> 7)	7.78%	7.40%	8.02%

5.3 Evaluation Metric

We use four metrics, to evaluate the DeepNote-GNN and the proposed multimodal model, as follows:

5.3.1 Area under the Receiver Operating Characteristic Curve

The area under the ROC curve (AUROC) is one of the important metrics to evaluate the performance of a classification model. The ROC curve is plotted with the true positive rate,

known as recall, on y-axis and the false positive rate, also called precision, on the x-axis. Equation 5.1 present how to compute the precision and recall.

$$\begin{aligned} \text{recall} &= \frac{TP}{TP + FN} \\ \text{precision} &= \frac{TN}{TN + FP} \end{aligned} \tag{5.1}$$

where TP and TN are associated with the outcomes where the model predicts the positive and negative classes correctly. Also, FP and FN are linked to the outcomes where the model incorrectly predicts the positive and negative classes.

A model with an excellent performance on a downstream task has an AUROC as high as one. On the other hand, the worst model has an AUROC close to zero. For a binary classification task, a random classifier has an AUROC close to one-half. In other words, it has high TP and TN values.

AUROC can perform well on evaluating the classification models trained on severely imbalanced datasets because it is not biased to majority or minority class. We use the AUROC metric to evaluate DeepNote-GNN on readmission prediction task, and the multimodal model on the mortality and length of stay (LOS) prediction tasks.

5.3.2 Area under the Precision-Recall Curve

The area under the precision-recall curve (AUPRC) is another suitable metric to measure a classifier's performance on an imbalanced dataset where finding the positive examples has a higher priority. For instance, in a mortality prediction task, the AUPRC value assists researchers in ensuring that the classifier correctly identifies all positive mortality samples without mistakenly classifying a negative sample as positive. Accurately detecting positive samples by a classifier has a clinical significance. To continue with our example, it is critical to identify patients who are at the risk of dying during an admission so that doctors can alter their care plan or medication. Furthermore, incorrectly predicting a patient's death can endanger the patient's life and waste hospital resources.

For a precision-recall curve, we have recall on the x-axis and precision on the y-axis. To get AUPRC, the area under the curve must be calculated. There are different ways to compute the area, such as the lower trapezoid estimator, the interpolated median estimator, and the average precision.

In contrast to ROC curve where the baseline is fixed and equals to one-half, the baseline for precision-recall curve is determined by the ratio of positives [81], presented by y in Equation 5.2.

$$y = \frac{P}{P + N} \tag{5.2}$$

with P stands for positive samples and N is for negative samples. For a balanced class distribution y value is close to 0.5.

To evaluate a classifier's performance by looking at its precision-recall curve, one needs to consider the relationship between precision and recall. Note that a naïve classifier that marks all samples as positive classes has a high recall and a low precision. On the other hand, a basic classifier which labels all samples as negative classes has high precision and a low recall. Therefore, the best classifier must have a high recall and a high precision, or based on the application, the trade-off between them must be determined.

High precision relates to a low false-positive rate, and high recall relates to a low false-negative rate. As a result, high AUPRC scores show that the classifier is returning accurate results (high precision) and returning a majority of all positive results (high recall).

We use AUPRC to evaluate our proposed multimodal model on a severely imbalanced dataset. Also, it is used to evaluate DeepNote-GNN on readmission prediction task using unstructred data.

5.3.3 Recall at Precision of 80%

Frequent alarms, many of which are non-actionable, reduce caregivers' awareness and finally lead to desensitization to alarms, called Alarm Fatigue, which can severely impact patient safety. Research has demonstrated that 72% to 99% of clinical alarms are false [82].

Consequently, it is important to minimize the false-positive rates of clinical predictive models. Computing recall at a precision of 80% (R@P80%) is a clinically-relevant metric

that enables us to build models that control the false positive rate. We use this metric to evalute the performance of DeepNote-GNN, which is trained on balanced cohort of data.

5.3.4 F1 Score

Since the relationship between precision and recall must be determined when evaluating a classification model, the F1 score can be used as a metric to establish an equal balance between precision and recall, which is particularly useful in most cases when working with imbalanced datasets.

The F1 score is interpreted as a weighted average of the precision and recall, demonstrated in Equation 5.3.

$$\text{F1 Score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{5.3}$$

The F1 score equals one, its maximum value, for a perfect classifier with precision and recall values both equal to one. Either precision or recall is zero for a weak classifier with poor class distinguishing capability, resulting in an F1 score approaching zero.

5.4 Baseline Models

In this research, our DeepNote-GNN model is compared with ClinicalBert and two other methods, which are considered as strong baselines for hospital readmission prediction using MIMIC-III clinical notes.

- ClinicalBert [3]: Pre-trains the state-of-the-art BERT architecture with MIMIC-III notes. Then fine-tunes the pre-trained BERT model with a fully connected layer added on top to make binary classification predictions. ClinicalBert uses a score aggregation to compute the prediction scores at an admission level.
- Bag-of-Word [83][4]: A vocabulary of top 5,000 Term Frequency-Inverse Document Frequency (TF-IDF) words is selected as features in this model. A clinical note is represented by a 5,000-dimensional vector where each entry is the count of the corresponding vocabulary word occurring in the note. Logistic regression with L2 regularization is used to fit the training readmission labels.

- Word2Vec with BiLSTM [55][84][4]: Word2Vec model is used to extract the input note embedding. These embeddings are then fed into a BiLSTM model to capture the temporal relationship between words in a note.

For the mortality and length of stay prediction tasks, we only use one baseline model, proposed by [80]. Bardak et al., [80] proposed a multimodal model that benefits from Med7 [85], a publicly available Named Entity Recognition (NER) algorithm trained on the MIMIC-III dataset, to extract seven different named entities related to patients' medications, such as drug, strenght, duration, route, form, dosage, and frequency.

After obtaining medical entities, word embedding techniques are used to generate deep representations out of them. The word embedding methods include Word2Vec [86] and FastText [87] models pre-trained on MIMIC-III dataset. Additionally, the concatenation of the representations obtained by them is used for further evaluation.

After extracting text embeddings, they apply convolutional layers, followed by a global max pooling layer, on them and concatenate the output features with temporal features. Temporal features are obtained by using a one-layer GRU recurrent model on time-series data, as shown in Figure 5.2.

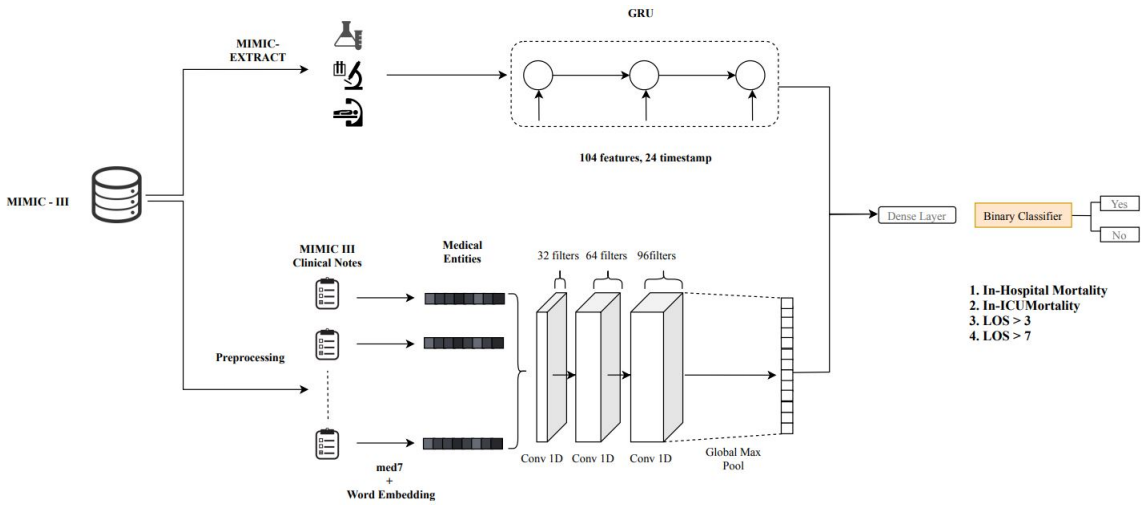


Figure 5.2. An overview of the baseline model for the mortality and length of stay prediction tasks

We compare our proposed multimodal model to their proposed model with the highest performance for each prediction task. Note that the difference between the models they

suggest lies in the embeddings of text data. The embeddings of text data in [80] can be obtained using the FastText algorithm, or Word2Vec algorithm, or the concatenation of their representations. The other architecture design, for example extracting temporal features, stays the same.

One weakness of the baseline model is that it uses Med7, which only extracts the patient's medication information from clinical notes. However, clinical notes may contain rich information other than medications useful to boost the model performance. For example, the observation of clinical notes obtained during the first day of hospitalization indicates some information about the patients' past diseases, allergies, age, etc. Each of these pieces of information can be helpful in the prediction process.

We apply the DeepNote network on clinical notes to extract their deep representations instead, which benefits from the BERT model. Not only does DeepNote consider all the information in clinical notes to generate representations, but also, unlike Word2Vec and FastText algorithms, it learns contextualized word representations.

5.5 Hyperparameter Setting

5.5.1 DeepNote-GNN

We follow the Bertbase hyperparameter setting released by [9] to extract the fixed admission feature vectors of size 768. Clinical notes for each subset of data are split into test, validation, and train sets with percentage values 10%, 10%, and 80%, respectively, and 5-fold cross-validation is conducted.

In the patient network, we train a two-layer GCN because it helps alleviate over-smoothing and overfitting issues. We train each patient network separately for a maximum of 200 epochs using Adam optimizer with a learning rate of 0.01 and early stopping with a window size of 10.

5.5.2 The Multimodal Model

For the mortality and length of stay prediction tasks the data is split into train, validation, and test sets with ratios of 70%, 10%, and 20%. We run the multimodal model with an Adam

optimizer with a learning rate of 0.001 and a weight decay equal to 0.0001. All components of the model are trained together for 50 epochs to minimize the negative log-likelihood loss. For the patient network, we train a two-layer GCN with a hidden layer of size 256. Each reported result is obtained by running the proposed model for ten times using ten different random seeds.

To obtain the best combination of note, temporal, and network weights for the score aggregation module, we run a grid search with a window of 0.1. For each prediction task, the best performance is obtained using the weights mentioned in Table 5.3

Table 5.3. The weights of the score aggregation for the multimodal model with the highest performance on a selected prediction task.

Prediction Task	W_{note}	$W_{temporal}$	$W_{network}$
Seven-day Length of Stay (LOS> 7)	0.1	0.1	0.8
Three-day Length of Stay (LOS> 3)	0.1	0.1	0.8
In-hospital Mortality	0.3	0.1	0.6
In-ICU Mortality	0.4	0.0	0.6

6. RESULTS

6.1 DeepNote-GNN

Discharge summaries contain valuable information about patients, including reasons for hospitalization, procedures, and treatment provided [88]. The information mentioned in a discharge summary note is comprehensive enough to be used by post-hospital teams. Consequently, discharge summaries have predictive value for hospital readmission.

DeepNote-GNN model outperforms baseline models evaluated on $D_{discharge}$, represented in Table 6.1. Feature aggregation added on top of the BERT architecture in DeepNote extracts a richer representation of admission features compared to score aggregation. Besides, the patient network learns from the connections between admissions with similar discharge summary.

Table 6.1. 30-day readmission prediction results using 5-fold cross-validation.

Dataset	Model	AUROC	AUPRC	R@P80%
$D_{discharge}$	DeepNote-GNN	$\mathbf{0.858 \pm 0.012}$	$\mathbf{0.847 \pm 0.015}$	$\mathbf{0.375 \pm 0.036}$
	ClinicalBert	0.768 ± 0.027	0.747 ± 0.029	0.255 ± 0.113
	Bag-of-Word	0.684 ± 0.025	0.674 ± 0.027	0.217 ± 0.119
	BiLSTM	0.694 ± 0.025	0.686 ± 0.029	0.223 ± 0.103
D_{3days}	DeepNote-GNN	$\mathbf{0.676 \pm 0.010}$	0.671 ± 0.010	0.161 ± 0.032
	ClinicalBert	0.674 ± 0.043	$\mathbf{0.677 \pm 0.044}$	$\mathbf{0.171 \pm 0.107}$
	Bag-of-Word	0.654 ± 0.035	0.657 ± 0.026	0.122 ± 0.106
	BiLSTM	0.656 ± 0.035	0.668 ± 0.028	0.150 ± 0.081

The performance improvement of DeepNote-GNN on D_{3days} compared to the ClincialBert is not significant. Clinical notes associated with an admission in D_{3days} contain an average of 9 text chunks. Therefore, compared to $D_{discharge}$, DeepNote aggregates a larger number of text chunk embeddings to build an admission feature for a clinical note at D_{3days} (look at the Equeation 4.3). As a result, the expressiveness of the final admission feature decreases.

Moreover, as the number of text chunks increases for admission, we have more text chunks that do not have valuable information about readmission.

Since ClinicalBert [3] predicts the readmission probabilities using a hand-engineered score aggregation, it avoids this limitation of the DeepNote. However, DeepNote treats all text chunk embeddings equally and obtains competitive results with minor apriori knowledge, and the model is more generalizable for hospital readmission prediction of a specific disease.

Two other baseline models, Bag-of-words and BiLSTM, cannot capture the context-dependent representations of words and perform poorly compared to models with built-in BERT architecture. The results gained using BiLSTM are close to that shown in the most recent literature [26].

6.2 The Multimodal Model

We compare the results of our proposed multimodal model on mortality and length of stay prediction tasks to a model proposed by [80] with the highest performance for that task. The results presented in Tables 6.2, 6.3, 6.4, and 6.5 show that the baseline models generally outperform our proposed model with an insignificant margin.

Table 6.2. The comparison between the proposed multimodal model and its best baseline on seven-day length of stay prediction task (LOS > 7).

Model	AUROC	AUPRC	F1 SCORE
Multimodal Model	0.7137	**0.1925**	**0.0534**
Baseline (Word2Vec)	**0.7255**	0.1878	0.0158

Table 6.3. The comparison between the proposed multimodal model and its best baseline on three-day length of stay prediction task (LOS > 3).

Model	AUROC	AUPRC	F1 SCORE
Multimodal Model	0.6793	0.6175	0.5347
Baseline (Word2Vec and FastText)	**0.6993**	**0.6277**	**0.5582**

Table 6.4. The comparison between the proposed multimodal model and its best baseline on in-hospital mortality prediction task.

Model	AUROC	AUPRC	F1 SCORE
Multimodal Model	0.8614	0.5265	0.4297
Baseline (Word2Vec)	**0.8755**	**0.5587**	**0.4723**

Table 6.5. The comparison between the proposed multimodal model and its best baseline on in-ICU mortality prediction task.

Model	AUROC	AUPRC	F1 SCORE
Multimodal Model	0.8629	0.4722	**0.4313**
Baseline (Word2Vec)	**0.8835**	**0.4923**	0.4302

Based on the weights provided in Table 5.3, we figure out that the DeepTemporal module has the least contribution to the model. Therefore, one reason behind the performance of our proposed model can be the architecture design of DeepTemporal.

Also, DeepNote utilizes BERT architecture to extract deep representations of clinical notes, while the baseline models benefit from word embedding algorithms. Although word embedding algorithms cannot capture context dependencies between words, they can provide a better solution for a specific dataset or an in-domain task.

7. MODEL ANALYSIS

7.1 DeepNote-GNN Model Analysis

We fully evaluate the DeepNote-GNN model for model robustness using less training data and the contribution of the two major components – DeepNote and Patient Network. Finally, we also evaluate DeepNote using the outputs of various hidden layers of the clinicalBert.

7.1.1 Model Robustness

To evaluate the robustness of DeepNote-GNN, we conduct a K-fold cross-validation technique on the discharge summaries dataset with a smaller value of K, equal to 2, and report the results. In a K-fold cross-validation experiment, the dataset is split into K sections or folds of approximately equal size. Iteratively, one fold is used as a validation set for model evaluation, and the rest is used as a train set.

A stable model should have similar performance metrics at every fold. i.e., the variance between folds remains insignificant. Cross-validation works well as the value of K increases considering that the training folds will cover a large proportion of data points. We can evaluate the model's robustness and generalization ability by decreasing the value of K. For K equal to 2, we train the model on half of the data and test it on the other half. We expect a robust model to generalize consistently on new unseen test samples.

Table 7.1 presents the performance of DeepNote-GNN on discharge summaries dataset for different values of K. As the number of K decreases, the model performance does not change significantly. Moreover, the variance between folds remains insignificant. The result tells that the DeepNote-GNN is a robust model. We think this attributes to the patient network. Since the patient network takes advantage of the data from similar patients within the network, less training data has a minimum impact on the model performance.

7.1.2 Model Component

We evaluate the contribution of patient network and DeepNote representation separately on the discharge summaries dataset. To evaluate the deep note representation, we add simple

Table 7.1. DeepNote-GNN performance on $D_{discharge}$ set using K-fold cross-validation.

Value of K	AUROC	AUPRC	R@P80%
5	0.858 ± 0.012	0.847 ± 0.015	0.375 ± 0.036
2	0.854 ± 0.001	0.844 ± 0.004	0.368 ± 0.003

classification layers - a fully connected layer of size 16 and an output layer of size 2 (binary classification) on top of the feature aggregator to replace the patient network and train it on the discharge summaries dataset. Dropout with a rate of 0.5 is used for the fully connected layer. We utilize the Adam optimizer with a learning rate of 0.001 to minimize the binary cross-entropy loss. The model is trained for 50 epochs with a window size of 10 and a batch size of 8.

The results in Figure 7.1 show that the DeepNote representation with simple classification layers outperforms the baseline ClinicalBert model. This demonstrates that the feature aggregation in the DeepNote representation model provides a richer representation of hospital admissions. Moreover, adding a patient network to the deep note representation boosts the overall performance more. The patient network is a computationally inexpensive architecture that benefits from pre-computed admission features.

7.1.3 DeepNote Representation Analysis

To build deep note representations, we can extract the activations from one or more layers of ClinicalBert without fine-tuning any parameters to represent the admission features. The choice of hidden states depends on the downstream task. Devlin et al. [9] conclude that the concatenation or summation of the last four hidden states of the encoders in BERT has the best performance on downstream tasks than other combinations of hidden states. Therefore, we repeat our experiment and use the sum of the last four hidden states of encoders as a text chunk representation. The ROC curve, demonstrated in Figure 7.2, show that DeepNote-GNN architecture outperforms when we model text chunk features using the last hidden

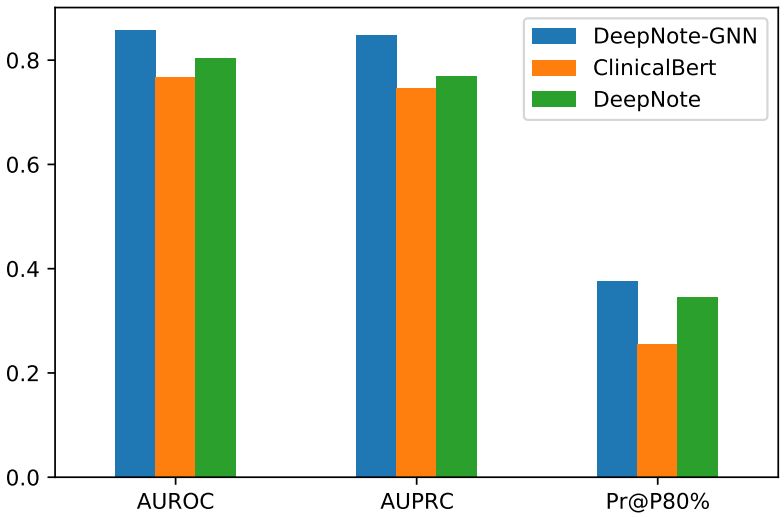


Figure 7.1. Analysis on the effect of the patient network using $D_{discharge}$ set

state. Note that the area under ROC curve is a metric to evaluate the model's ability in distinguishing between positive and negative labels.

Extracting the last four hidden states and processing them to build an admission representation is computationally more expensive than extracting the last hidden state. Also, summing more hidden states and applying feature aggregation reduces the expressiveness of admission features and loses the information in the data [31].

7.1.4 GCN vs. GAT for Patient Network

The most recent research on graph neural network for classification show that graph attention networks (GAT) [17] works better than the Graph Convolutional Networks (GCN). Hence, we also train the patient network with a Graph Attention Network and with the DeepNote representation using $D_{discharge}$ set, and compare with GCN that is used in our original design. GAT utilizes masked self-attentional layers to specify different weights to different nodes in a neighborhood.

We apply a two-layer GAT model on $D_{discharge}$ set. The first layer consists of 8 attention heads computing 8 features each (for a total of 64 features), followed by an Exponential

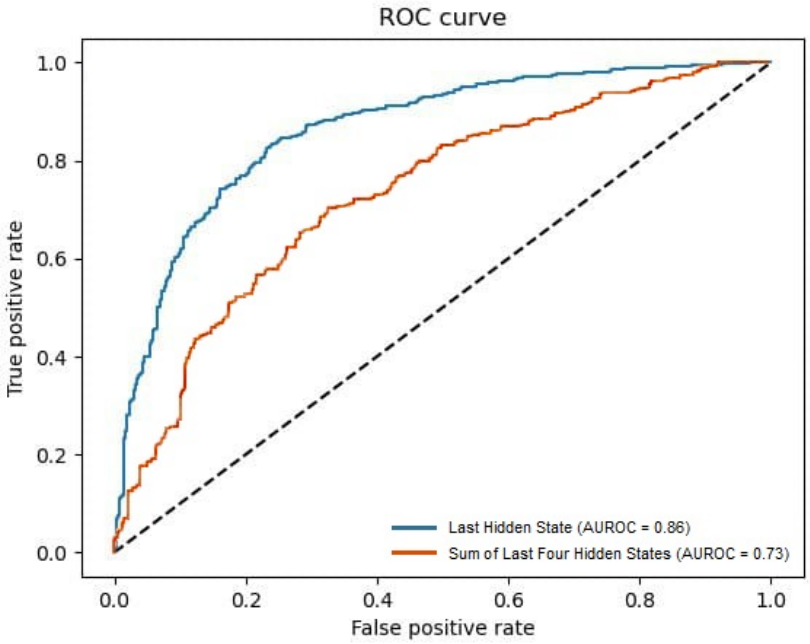


Figure 7.2. Deep note representation analysis on $D_{discharge}$ set using ROC curve.

Linear Unit (ELU) [89] nonlinearity. The second layer contains a single attention head, followed by a softmax activation for binary classification. We optimize the model using an Adam optimizer with a learning rate of 0.005 for 900 epochs with a patience value of 50.

Table 7.2 shows that there is an insignificant performance gap between GAT and GCN. The reason could lie in the number of classes and the neighborhood structure of our data. GAT handles a graph with a more complex neighborhood structure well. Since we perform a binary classification and the neighborhood is constructed implicitly using cosine similarity metric with a specified threshold, GCN performs sufficiently well to classify the nodes.

Table 7.2. The comparison between the performance of DeepNote-GNN using the Graph Convolutional Network (GCN) and Graph Attention Network (GAT) as patient network.

Patient Network	AUROC	AUPRC	R@P80%
Graph Convolutional Network (GCN)	0.858 ± 0.012	0.847 ± 0.015	0.375 ± 0.036
Graph Attention Network (GAT)	0.856 ± 0.011	0.845 ± 0.016	0.373 ± 0.037

7.2 The Multimodal Model Analysis

To determine the contribution of various components of our proposed multimodal model, we perform an ablation study. For each prediction task, we measure the performance of each module by setting its corresponding weight into one and other weights to zero. The results show that for all prediction tasks, our proposed multimodal model has the highest performance. Therefore, combining the modules, as we did, is the best approach.

The patient network outperforms two other modules, and the DeepTemporal module has the poorest performance. Although DeepTemporal cannot perform well on prediction tasks separately, it adds to the model's performance when it is combined with DeepNote and patient network. As a result, integrating structured and unstructured data and adding a patient network can boost the model performance on mortality and length of stay prediction tasks.

The results are demonstrated in the following tables and figures. Tables 7.3, 7.4, 7.5, and 7.6 present the results for respectively the three-day length of stay, seven-day length of stay, in-hospital mortality, and in-ICU mortality prediction tasks. Keeping the same order as mentioned above, the receiver operating characteristic curve are presented in Figures 7.6, 7.4, 7.8, and 7.10. Also, the precision-recall curves are demonstrated in Figures 7.5, 7.3, 7.7, and 7.9.

Table 7.3. The performance of our proposed model and its various modules on the three-day length of stay (LOS > 3) prediction task.

Model	W_{note}	$W_{temporal}$	$W_{network}$	AUROC	AUPRC	F1 SCORE
Proposed Model	0.1	0.1	0.8	**0.6802**	**0.6179**	**0.5411**
DeepNote	1.0	0.0	0.0	0.6374	0.5690	0.4515
DeepTemporal	0.0	1.0	0.0	0.5668	0.4972	0.3439
Patient Network	0.0	0.0	1.0	0.6599	0.5926	0.5050

Table 7.4. The performance of our proposed model and its various modules on the seven-day length of stay (LOS > 7) prediction task.

Model	W_{note}	$W_{temporal}$	$W_{network}$	AUROC	AUPRC	F1 SCORE
Proposed Model	0.1	0.1	0.8	**0.7177**	**0.1892**	**0.0546**
DeepNote	1.0	0.0	0.0	0.5955	0.1299	0.0097
DeepTemporal	0.0	1.0	0.0	0.5835	0.1076	0.0052
Patient Network	0.0	0.0	1.0	0.6716	0.1537	0.0515

Table 7.5. The performance of our proposed model and its various modules on in-hospital mortality prediction task.

Model	W_{note}	$W_{temporal}$	$W_{network}$	AUROC	AUPRC	F1 SCORE
Proposed Model	0.35	0.05	0.6	**0.8613**	**0.5248**	**0.4333**
DeepNote	1.0	0.0	0.0	0.8238	0.4375	0.3666
DeepTemporal	0.0	1.0	0.0	0.7696	0.2834	0.1060
Patient Network	0.0	0.0	1.0	0.8476	0.5077	0.4246

Table 7.6. The performance of our proposed model and its various modules on in-ICU mortality prediction task.

Model	W_{note}	$W_{temporal}$	$W_{network}$	AUROC	AUPRC	F1 SCORE
Proposed Model	0.4	0.0	0.6	**0.8624**	**0.4731**	**0.4527**
DeepNote	1.0	0.0	0.0	0.8127	0.3789	0.3453
DeepTemporal	0.0	1.0	0.0	0.7482	0.2237	0.0950
Patient Network	0.0	0.0	1.0	0.8538	0.4574	0.4309

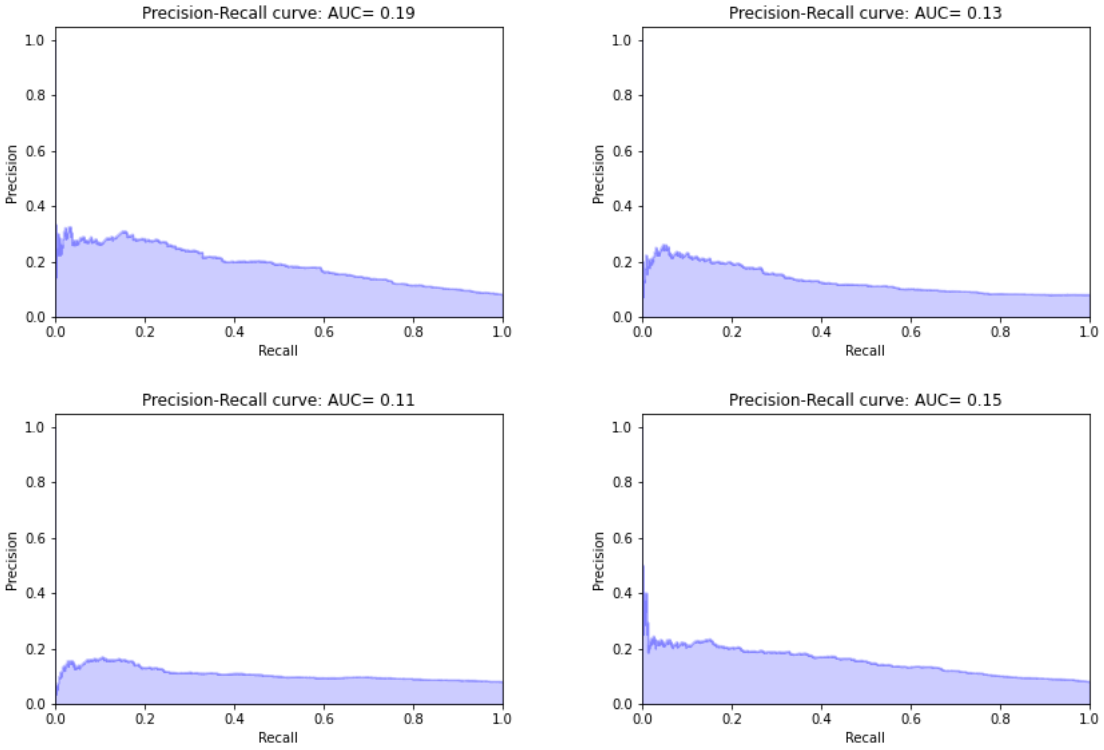


Figure 7.3. The precision-recall curves for the proposed model (top-left) and its modules on the seven-day length of stay (LOS > 7) prediction task: DeepNote (top-right), DeepTemporal (bottom-left), and the patient network (bottom-right).

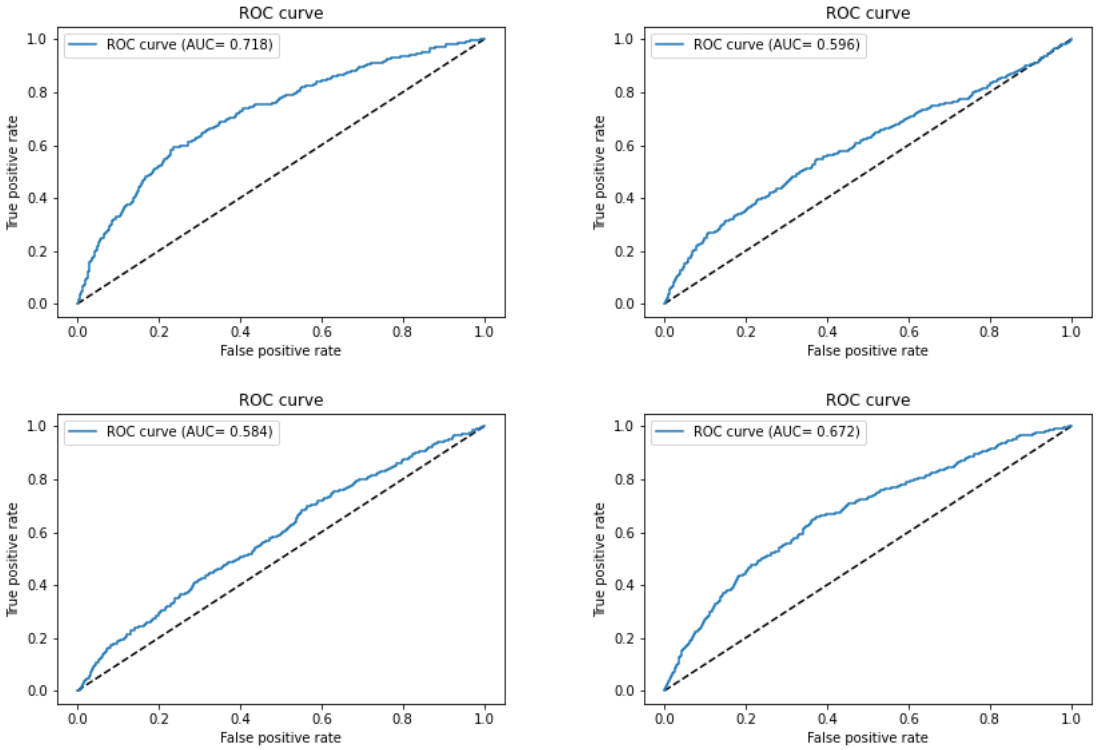


Figure 7.4. The receiver operating characteristic curves for the proposed model (top-left) and its modules on the seven-day length of stay (LOS > 7) prediction task: DeepNote (top-right), DeepTemporal (bottom-left), and the patient network (bottom-right).

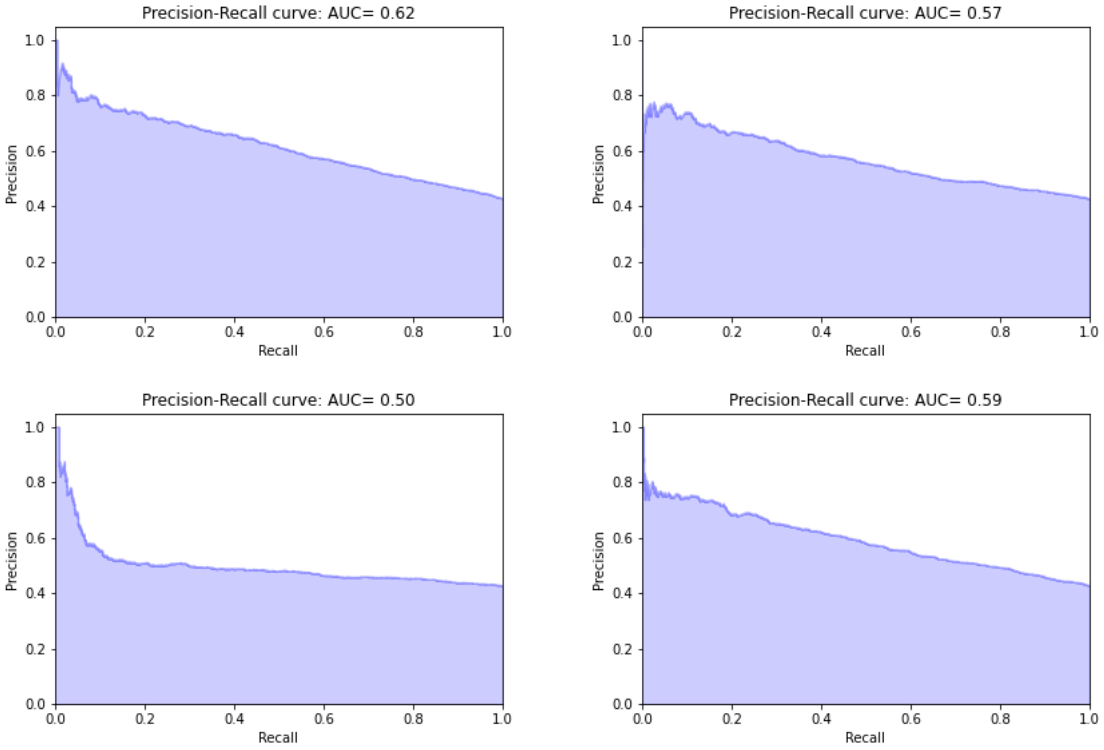


Figure 7.5. The precision-recall curves for the proposed model (top-left) and its modules on the three-day length of stay (LOS > 3) prediction task: DeepNote (top-right), DeepTemporal (bottom-left), and the patient network (bottom-right).

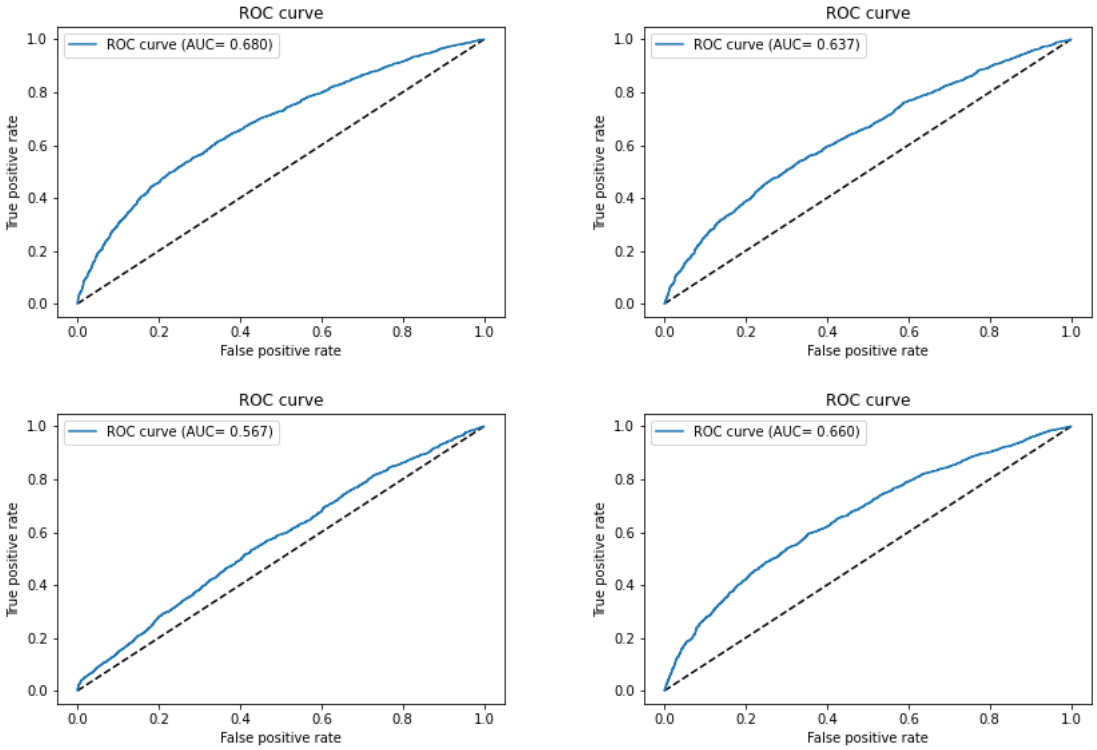


Figure 7.6. The receiver operating characteristic curves for the proposed model (top-left) and its modules on the three-day length of stay (LOS $>$ 3) prediction task: DeepNote (top-right), DeepTemporal (bottom-left), and the patient network (bottom-right).

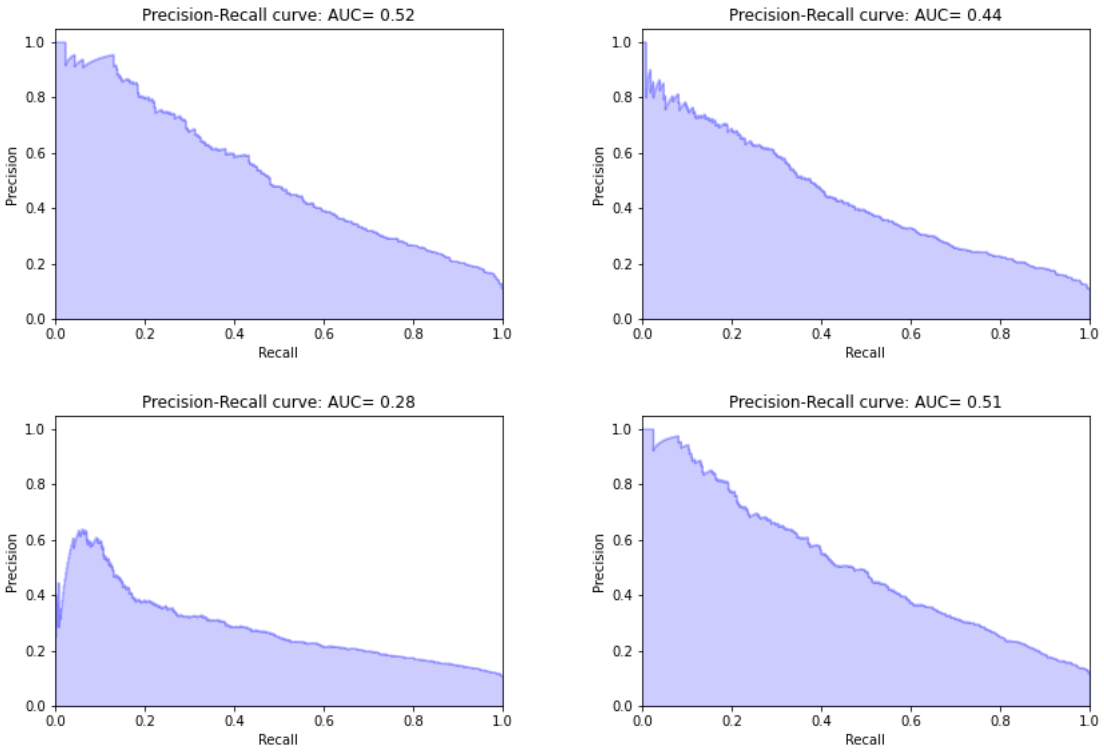


Figure 7.7. The precision-recall curves for the proposed model (top-left) and its modules on in-hospital mortality prediction task: DeepNote (top-right), DeepTemporal (bottom-left), and the patient network (bottom-right).

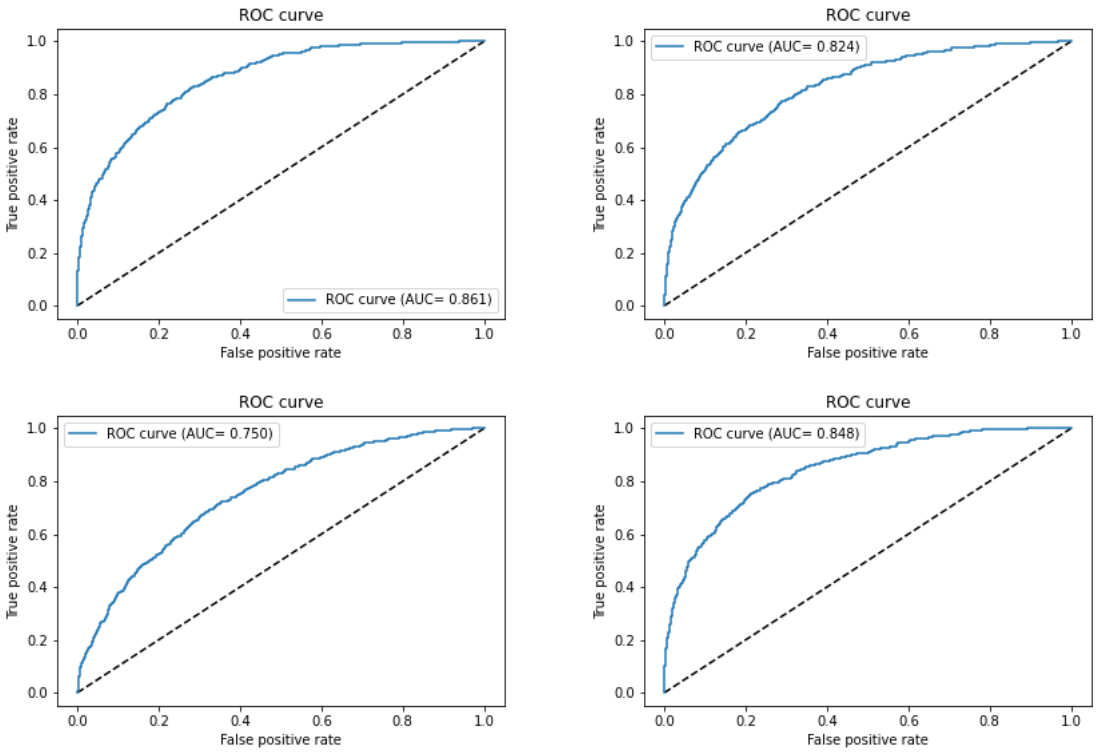


Figure 7.8. The receiver operating characteristic curves for the proposed model (top-left) and its modules on in-hospital mortality prediction task: DeepNote (top-right), DeepTemporal (bottom-left), and the patient network (bottom-right).

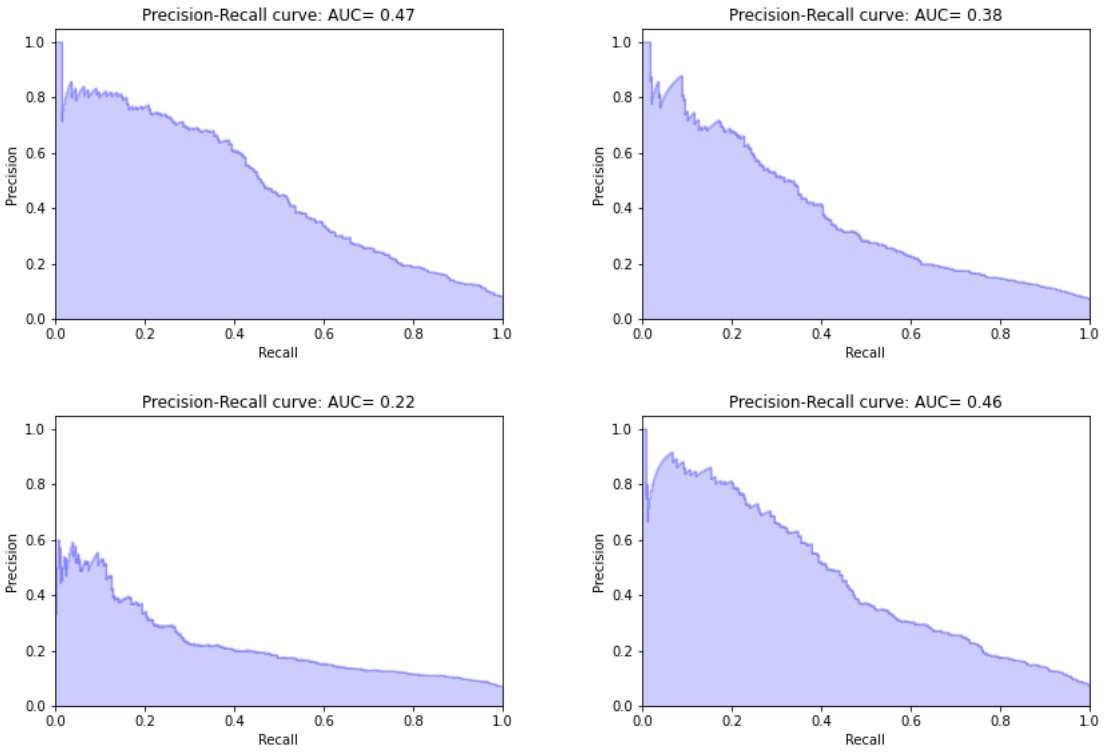


Figure 7.9. The precision-recall curves for the proposed model (top-left) and its modules on in-ICU mortality prediction task: DeepNote (top-right), DeepTemporal (bottom-left), and the patient network (bottom-right).

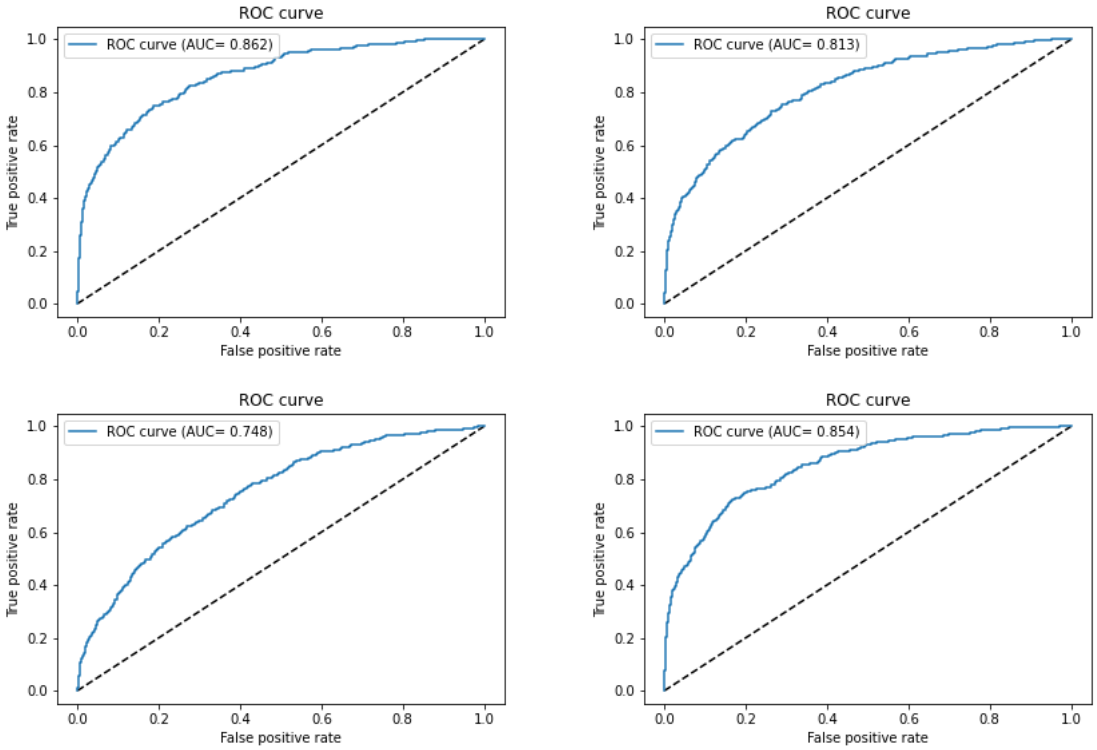


Figure 7.10. The receiver operating characteristic curves for the proposed model (top-left) and its modules on in-ICU mortality prediction task: DeepNote (top-right), DeepTemporal (bottom-left), and the patient network (bottom-right).

8. CONCLUSION

In this research, we investigated a novel deep learning framework - DeepNote-GNN to predict 30-day hospital readmission outcomes. DeepNote-GNN consists of two main components: DeepNote that is in charge of extracting deep contextual representations of clinical notes using a feature aggregator added on top of the clinicalBERT algorithm, and a patient network that extracts information from the topological structure of the patient graph and trains the graph using a convolutional neural network.

DeepNote-GNN adopts a feature-based approach which makes its training process computationally efficient and introduces additional task-specific parameters to its structure. We evaluated DeepNote-GNN on 30-day hospital readmission prediction task using MIMIC-III dataset. The performance results show that DeepNote-GNN performs significantly better than the baseline models when discharge summary is used.

Our model analysis shows that the performance of DeepNote-GNN does not impact much by the size of the training data, and the DeepNote representation is robust. The ablation study confirms that patient network contributes significantly to the overall performance. This also reflects the need to use patient network to utilize similar patient data for prediction.

To evaluate the generalization capability of the DeepNote-GNN to new prediction tasks and new sets of data, we also install its main components to a multimodal model and train it on mortality and length of stay prediction tasks. The results show that a patient network built by incorporating the structured data can also improve the overall performance.

The future work includes modifying the architecture of the proposed multimodal model to improve its performance so that it outperforms its baseline models. We also look forward to implementing the DeepNote and patient network modules into other deep learning models and evaluating them on new clinical prediction tasks.

REFERENCES

[1] G. S. Birkhead, M. Klompas, and N. R. Shah, "Uses of electronic health records for public health surveillance to advance public health," *Annual review of public health*, vol. 36, pp. 345–359, 2015.

[2] B. Shickel, P. J. Tighe, A. Bihorac, and P. Rashidi, "Deep ehr: A survey of recent advances in deep learning techniques for electronic health record (ehr) analysis," *IEEE journal of biomedical and health informatics*, vol. 22, no. 5, pp. 1589–1604, 2017.

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

[4] W. Boag, D. Doss, T. Naumann, and P. Szolovits, "What's in a note? unpacking predictive value in clinical note representations," *AMIA Summits on Translational Science Proceedings*, vol. 2018, p. 26, 2018.

[5] J. Kemp, A. Rajkomar, and A. M. Dai, "Improved patient classification with language model pretraining over clinical notes," *arXiv preprint ArXiv:1909.03039*, 2019.

[6] C. Xiao, E. Choi, and J. Sun, "Opportunities and challenges in developing deep learning models using electronic health records data: A systematic review," *Journal of the American Medical Informatics Association*, vol. 25, no. 10, pp. 1419–1428, 2018.

[7] K.-H. Yu, A. L. Beam, and I. S. Kohane, "Artificial intelligence in healthcare," *Nature biomedical engineering*, vol. 2, no. 10, pp. 719–731, 2018.

[8] 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*, 2017, pp. 787–795.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[10] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

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

[12] D. J. Morgan, B. Bame, P. Zimand, P. Dooley, K. A. Thom, A. D. Harris, S. Bentzen, W. Ettinger, S. D. Garrett-Ray, J. K. Tracy, *et al.*, "Assessment of machine learning vs standard prediction rules for predicting hospital readmissions," *JAMA network open*, vol. 2, no. 3, e190348–e190348, 2019.

[13] J. Donzé, D. Aujesky, D. Williams, and J. L. Schnipper, "Potentially avoidable 30-day hospital readmissions in medical patients: Derivation and validation of a prediction model," *JAMA internal medicine*, vol. 173, no. 8, pp. 632–638, 2013.

[14] A. E. Nijhawan, L. R. Metsch, S. Zhang, D. J. Feaster, L. Gooden, M. K. Jain, R. Walker, S. Huffaker, M. J. Mugavero, P. Jacobs, *et al.*, "Clinical and sociobehavioral prediction model of 30-day hospital readmissions among people with hiv and substance use disorder: Beyond electronic health record data," *JAIDS Journal of Acquired Immune Deficiency Syndromes*, vol. 80, no. 3, pp. 330–341, 2019.

[15] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

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

[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, 2018. arXiv: 1710.10903 [stat.ML].

[18] D. M. Shahian, G. S. Meyer, E. Mort, S. Atamian, X. Liu, A. S. Karson, L. D. Ramunno, and H. Zheng, "Association of national hospital quality measure adherence with long-term mortality and readmissions," *BMJ quality & safety*, vol. 21, no. 4, pp. 325–336, 2012.

[19] R. B. Zuckerman, S. H. Sheingold, E. J. Orav, J. Ruhter, and A. M. Epstein, "Readmissions, observation, and the hospital readmissions reduction program," *New England Journal of Medicine*, vol. 374, no. 16, pp. 1543–1551, 2016.

[20] J. Futoma, J. Morris, and J. Lucas, "A comparison of models for predicting early hospital readmissions," *Journal of biomedical informatics*, vol. 56, pp. 229–238, 2015.

[21] D. Kansagara, H. Englander, A. Salanitro, D. Kagen, C. Theobald, M. Freeman, and S. Kripalani, "Risk prediction models for hospital readmission: A systematic review," *Jama*, vol. 306, no. 15, pp. 1688–1698, 2011.

[22] X. Min, B. Yu, and F. Wang, "Predictive modeling of the hospital readmission risk from patients' claims data using machine learning: A case study on copd," *Scientific reports*, vol. 9, no. 1, pp. 1–10, 2019.

[23] H. Wang, Z. Cui, Y. Chen, M. Avidan, A. B. Abdallah, and A. Kronzer, "Predicting hospital readmission via cost-sensitive deep learning," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 15, no. 6, pp. 1968–1978, 2018.

[24] S. Barbieri, J. Kemp, O. Perez-Concha, S. Kotwal, M. Gallagher, A. Ritchie, and L. Jorm, "Benchmarking deep learning architectures for predicting readmission to the icu and describing patients-at-risk," *Scientific reports*, vol. 10, no. 1, pp. 1–10, 2020.

[25] A. Ashfaq, A. Sant'Anna, M. Lingman, and S. Nowaczyk, "Readmission prediction using deep learning on electronic health records," *Journal of biomedical informatics*, vol. 97, p. 103 256, 2019.

[26] D. Zhang, C. Yin, J. Zeng, X. Yuan, and P. Zhang, "Combining structured and unstructured data for predictive models: A deep learning approach," *BMC medical informatics and decision making*, vol. 20, no. 1, pp. 1–11, 2020.

[27] X. Liu, Y. Chen, J. Bae, H. Li, J. Johnston, and T. Sanger, "Predicting heart failure readmission from clinical notes using deep learning," in *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, 2019, pp. 2642–2648.

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

[29] Y. Goldberg and O. Levy, "Word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.

[30] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," *arXiv preprint arXiv:1712.09405*, 2017.

[31] O. Schulte and K. Routley, "Aggregating predictions vs. aggregating features for relational classification," in *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, IEEE, 2014, pp. 121–128.

[32] J. Schrodt, A. Dudchenko, P. Knaup-Gregori, and M. Ganzinger, "Graph-representation of patient data: A systematic literature review," *Journal of medical systems*, vol. 44, no. 4, pp. 1–7, 2020.

[33] E. Choi, Z. Xu, Y. Li, M. W. Dusenberry, G. Flores, Y. Xue, and A. M. Dai, "Graph convolutional transformer: Learning the graphical structure of electronic health records," *arXiv preprint arXiv:1906.04716*, 2019.

[34] B. Malone, A. Garcia-Duran, and M. Niepert, "Learning representations of missing data for predicting patient outcomes," *arXiv preprint arXiv:1811.04752*, 2018.

[35] A. García-Durán and M. Niepert, "Learning graph representations with embedding propagation," *arXiv preprint arXiv:1710.03059*, 2017.

[36] E. Rocheteau, C. Tong, P. Veličković, N. Lane, and P. Liò, "Predicting patient outcomes with graph representation learning," *arXiv preprint arXiv:2101.03940*, 2021.

[37] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[38] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[39] D. P. Kingma and J. Ba, "Adam: A methodfor stochastic optimization," in *International Conference onLearning Representations (ICLR)*, 2015.

[40] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, 2001.

[41] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[42] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, Ieee, 2017, pp. 1–6.

[43] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The World Wide Web Conference*, 2019, pp. 417–426.

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

[45] T. Navamani, "Efficient deep learning approaches for health informatics," in *Deep Learning and Parallel Computing Environment for Bioengineering Systems*, Elsevier, 2019, pp. 123–137.

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

[47] 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.

[48] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[50] L. Franz, Y. R. Shrestha, and B. Paudel, "A deep learning pipeline for patient diagnosis prediction using electronic health records," *arXiv preprint arXiv:2006.16926*, 2020.

[51] S. Ruder, M. E. Peters, S. Swayamdipta, and T. Wolf, "Transfer learning in natural language processing," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, 2019, pp. 15–18.

[52] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[53] S. Biswal, C. Xiao, L. M. Glass, E. Milkovits, and J. Sun, "Doctor2vec: Dynamic doctor representation learning for clinical trial recruitment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 557–564.

[54] C. A. Pedersen, P. J. Schneider, and D. J. Scheckelhoff, "Ashp national survey of pharmacy practice in hospital settings: Prescribing and transcribing—2016," *American Journal of Health-System Pharmacy*, vol. 74, no. 17, pp. 1336–1352, 2017.

[55] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *arXiv preprint arXiv:1310.4546*, 2013.

[56] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[57] K. Huang, A. Singh, S. Chen, E. Moseley, C.-y. Deng, N. George, and C. Lindvall, "Clinical xlnet: Modeling sequential clinical notes and predicting prolonged mechanical ventilation," *arXiv:1912.11975*, 2019.

[58] A. Salamat, X. Luo, and A. Jafari, "Heterographrec: A heterogeneous graph-based neural networks for social recommendations," *Knowledge-Based Systems*, vol. 217, p. 106 817, 2021.

[59] D. Ramachandram and G. W. Taylor, "Deep multimodal learning: A survey on recent advances and trends," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 96–108, 2017.

[60] R. Wallmann, J. Llorca, I. Gómez-Acebo, Á. C. Ortega, F. R. Roldan, and T. Dierssen-Sotos, "Prediction of 30-day cardiac-related-emergency-readmissions using simple administrative hospital data," *International journal of cardiology*, vol. 164, no. 2, pp. 193–200, 2013.

[61] K. Dharmarajan, A. F. Hsieh, Z. Lin, H. Bueno, J. S. Ross, L. I. Horwitz, J. A. Barreto-Filho, N. Kim, S. M. Bernheim, L. G. Suter, *et al.*, "Diagnoses and timing of 30-day readmissions after hospitalization for heart failure, acute myocardial infarction, or pneumonia," *Jama*, vol. 309, no. 4, pp. 355–363, 2013.

[62] C. A. Baillie, C. VanZandbergen, G. Tait, A. Hanish, B. Leas, B. French, C. William Hanson, M. Behta, and C. A. Umscheid, "The readmission risk flag: Using the electronic health record to automatically identify patients at risk for 30-day readmission," *Journal of hospital medicine*, vol. 8, no. 12, pp. 689–695, 2013.

[63] W. Boulding, S. W. Glickman, M. P. Manary, K. A. Schulman, and R. Staelin, "Relationship between patient satisfaction with inpatient care and hospital readmission within 30 days.," *The American journal of managed care*, vol. 17, no. 1, pp. 41–48, 2011.

[64] A. Artetxe, A. Beristain, and M. Grana, "Predictive models for hospital readmission risk: A systematic review of methods," *Computer methods and programs in biomedicine*, vol. 164, pp. 49–64, 2018.

[65] E. M. Hechenbleikner, M. A. Makary, D. V. Samarov, J. L. Bennett, S. L. Gearhart, J. E. Efron, and E. C. Wick, "Hospital readmission by method of data collection," *Journal of the American College of Surgeons*, vol. 216, no. 6, pp. 1150–1158, 2013.

[66] S. Purushotham, C. Meng, Z. Che, and Y. Liu, "Benchmark of deep learning models on large healthcare mimic datasets," *arXiv preprint arXiv:1710.08531*, 2017.

[67] R. Sadeghi, T. Banerjee, and W. Romine, "Early hospital mortality prediction using vital signals," *Smart Health*, vol. 9, pp. 265–274, 2018.

[68] H.-Y. Shi, K.-T. Lee, H.-H. Lee, W.-H. Ho, D.-P. Sun, J.-J. Wang, and C.-C. Chiu, "Comparison of artificial neural network and logistic regression models for predicting in-hospital mortality after primary liver cancer surgery," *PloS one*, vol. 7, no. 4, e35781, 2012.

[69] Z. Dai, S. Liu, J. Wu, M. Li, J. Liu, and K. Li, "Analysis of adult disease characteristics and mortality on mimic-iii," *PloS one*, vol. 15, no. 4, e0232176, 2020.

[70] J. Rapoport, D. Teres, Y. Zhao, and S. Lemeshow, "Length of stay data as a guide to hospital economic performance for icu patients," *Medical care*, pp. 386–397, 2003.

[71] T. Gentimis, A. Ala'J, A. Durante, K. Cook, and R. Steele, "Predicting hospital length of stay using neural networks on mimic iii data," in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, IEEE, 2017, pp. 1194–1201.

[72] G. Weglarz, "Two worlds of data unstructured and structured," *Information Management*, vol. 14, no. 9, p. 19, 2004.

[73] K. Häyrinen, K. Saranto, and P. Nykänen, "Definition, structure, content, use and impacts of electronic health records: A review of the research literature," *International journal of medical informatics*, vol. 77, no. 5, pp. 291–304, 2008.

[74] T. Botsis, G. Hartvigsen, F. Chen, and C. Weng, "Secondary use of ehr: Data quality issues and informatics opportunities," *Summit on Translational Bioinformatics*, vol. 2010, p. 1, 2010.

[75] P. B. Jensen, L. J. Jensen, and S. Brunak, "Mining electronic health records: Towards better research applications and clinical care," *Nature Reviews Genetics*, vol. 13, no. 6, pp. 395–405, 2012.

[76] F. E. Harrell Jr, K. L. Lee, R. M. Califf, D. B. Pryor, and R. A. Rosati, "Regression modelling strategies for improved prognostic prediction," *Statistics in medicine*, vol. 3, no. 2, pp. 143–152, 1984.

[77] J. R. A. Solares, F. E. D. Raimondi, Y. Zhu, F. Rahimian, D. Canoy, J. Tran, A. C. P. Gomes, A. H. Payberah, M. Zottoli, M. Nazarzadeh, *et al.*, "Deep learning for electronic health records: A comparative review of multiple deep neural architectures," *Journal of biomedical informatics*, vol. 101, p. 103 337, 2020.

[78] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-Wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[79] S. Wang, M. B. McDermott, G. Chauhan, M. Ghassemi, M. C. Hughes, and T. Naumann, "Mimic-extract: A data extraction, preprocessing, and representation pipeline for mimic-iii," in *Proceedings of the ACM Conference on Health, Inference, and Learning*, 2020, pp. 222–235.

[80] B. Bardak and M. Tan, "Improving clinical outcome predictions using convolution over medical entities with multimodal learning," *Artificial Intelligence in Medicine*, p. 102 112, 2021.

[81] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," *PloS one*, vol. 10, no. 3, e0118432, 2015.

[82] S. Sendelbach and M. Funk, "Alarm fatigue: A patient safety concern," *AACN advanced critical care*, vol. 24, no. 4, pp. 378–386, 2013.

[83] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: A statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1-4, pp. 43–52, 2010.

[84] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[85] A. Kormilitzin, N. Vaci, Q. Liu, and A. Nevado-Holgado, "Med7: A transferable clinical natural language processing model for electronic health records," *Artificial Intelligence in Medicine*, p. 102 086, 2021.

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

[87] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[88] A. J. Kind and M. A. Smith, "Documentation of mandated discharge summary components in transitions from acute to subacute care," *Advances in patient safety: new directions and alternative approaches (Vol. 2: culture and redesign)*, 2008.

[89] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.