Deep Learning Models for Clinical Data: Addressing Task Specific Structure

by

Jeeheh Oh

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2021

Jeeheh Oh

jeeheh@umich.edu

ORCID iD: 0000-0002-8537-3070

Acknowledgements

A big thanks to everyone involved on this PhD journey; advice and acts of kindness big and small were all appreciated. Special thanks to Jenna Wiens for being an awesome advisor and friends and family for being there.

Table of Contents

List of Figures

List of Tables

List of Appendices

Abstract

As the adoption of electronic health records (EHRs) increases, so do the opportunities to improve patient care using these data. Applied to high-dimensional EHR data, machine learning techniques can help identify complex relationships between patient covariates and outcomes. However, in order to augment clinical care these models must generalize to (i.e., perform well on) never-before-seen data. In healthcare settings, generalization performance is often hindered by limited training data. Though clinical data are high dimensional, there is often a limited number of examples that can be used for training, due to low incidence rates of the outcomes of interest. To address this challenge, we develop and evaluate methods that combine deep learning techniques with knowledge about the task structure to improve sample efficiency.

Throughout this dissertation, we augment learning algorithms by exploiting known task structure pertaining to i) invariances, ii) signal dynamics, and address challenges associated with iii) class imbalance driven low homophily. First, different types of temporal invariances (e.g., phase invariance) are present in clinical time-series tasks. However, such invariances may vary across tasks. We propose a novel approach, 'Sequence Transformer Networks' that learns to recognize and exploit task-specific invariances, reducing intra-class variance and improving performance. Second, risk factors for a given adverse outcome may change as a patient's admission progresses. Though techniques like recurrent neural networks (RNNs) should in theory be able to capture such time varying dynamics, when training data are limited performance can suffer. We propose a novel RNN-based architecture in which we *relax weight sharing* over time to capture time-varying relationships. Finally, graph neural networks (GNNs) are a popular method for learning feature representations from graphs but are often evaluated on tasks with graphs presenting high homophily (or high similarity among connected nodes). In fact, GNNs are known to do poorly in low homophily settings. In clinical tasks, high class imbalance leads to asymmetrical homophily: high homophily with respect to the majority class and on average, but low homophily with respect to the minority class of interest. We address class imbalance driven low homophily by evaluating an attention-based mechanism against recently proposed methods for dealing with low homophily in general. By adapting techniques to better leverage task structure such as class imbalance driven low homophily, we are able to improve sample efficiency and predictive performance on tasks such as estimating the risk for adverse patient outcomes.

Chapter 1

Introduction

The electronic health record (EHR) contains vast amounts of data that have the potential to improve multiple facets of patient care from predicting future diagnoses using longitudinal patient data [1] to estimating patient risk for in-hospital mortality [2]. However, it is difficult for humans to gather meaningful or actionable conclusions that leverage the entirety of the EHR. These data consist of thousands of longitudinal time series (*e.g.*, medications, vitals, diagnosis, patient locations) across hundreds if not thousands of admissions. The heterogeneity and scale of the data make it difficult for clinicians to draw useful or actionable conclusions. **We are interested in the development, evaluation and application of algorithms that learn to distill clinical time-series data into patient specific risk estimates for adverse outcomes** (*e.g.*, infection, mortality, exposure).

There are many technical challenges that arise when applying machine learning to EHR data for patient risk stratification. In this dissertation, **we explore challenges that arise when the amount of training data are limited**. Within an EHR system, though there are potentially thousands of hospital admissions each year and thousands of features representing all the medications and procedures, the outcome of interest (e.g., hospital associated infections) may occur a relatively small number of times. Moreover, labelling data for complex outcomes such as severity of disease often requires manual chart review resulting in even smaller sample sizes. Limited or highly unbalanced training data make it difficult to learn models that generalize, especially when using deep learning approaches. Though deep learning has recently been heralded for its ability to adapt to tasks with minimal domain knowledge, it relies on large sample sizes to learn complex relationships and prevent overfitting. To address the limited data issue we present task structures often found in clinical tasks and propose ways to adapt machine learning methods to exploit and address these structures. **Figure** 1.1 summarizes the task structures and informed approaches that will be addressed in this dissertation.

Throughout this dissertation we use predicting patient risk for adverse outcomes from structural EHR data as our motivating application. We focus specifically on the tasks of predicting in-hospital mortality, acute respiratory failure, shock (i.e., inadequate perfusion of blood oxygen to organs or tissues), and risk of hospital onset *Clostridioides difficile* infection (CDI). These tasks have high clinical relevance. ARF contributes to over 380,000 deaths in the US per year [3] and represents a challenging prediction task due to its multifactorial etiology. Shock refers to the inadequate perfusion of blood oxygen to organs or tissues and can result in severe organ dysfunction and death when not recognized and treated immediately [4]. CDI is one of the most common healthcare associated infections in the United States [5] and is known for lingering on hospital surfaces [6]. All three of these conditions are upstream events that contribute to patient risk of in-hospital mortality, our fourth prediction task. The ability to identify patients at risk of developing ARF, shock or CDI could facilitate timely intervention, preventing irreversible damage and yielding better patient outcomes.

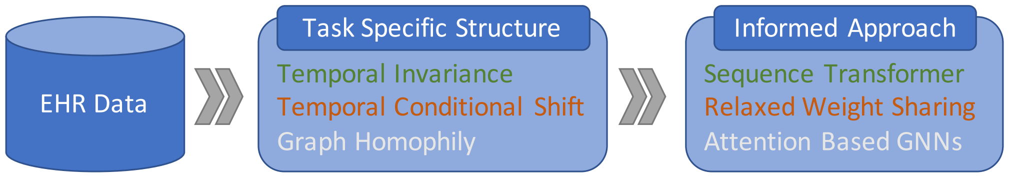


FIGURE 1.1: Structure of the dissertation's contributions summarized by the task specific structure and the corresponding informed approach.

1.1 Challenges and Opportunities

To overcome challenges associated with a paucity of training data, **we present problem settings commonly found in healthcare tasks and propose new deep learning approaches that consider known structure about these settings**. We leverage task structures such as (1) temporal invariances and (2) signal dynamics and address challenges associated with (3) low task homophily. These architectures aim to exploit these structures in a way that leads to improved generalization performance. We briefly introduce each of these task structures and motivate our proposed solutions, in turn, below.

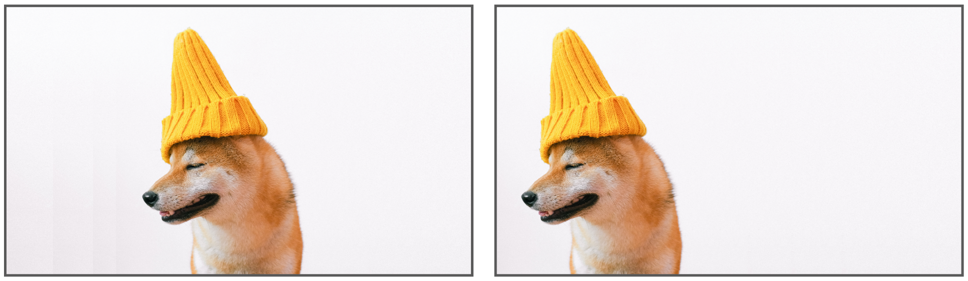


FIGURE 1.2: Invariances are transformations that, when applied to the data, do not change the task-specific meaning of the data. An example is shown with respect to object detection of a dog in an image: the image on the right has been shifted but still contains a dog. This task is, to an extent, shift invariant.

Temporal invariances. Invariances are transformations that, when applied to the data, do not change the task-specific meaning of the data [7]. An example of shift invariance with respect to object detection in images can be seen in **Figure** 1.2. In healthcare, when classifying chest X-rays, pneumonia may be recognized based on the presences of infiltrates or white spots in the lungs and not necessarily the precise location of those white spots [8]. These represent a type of spatial invariance. In our work, we aim to exploit *temporal* invariances with the goal of reducing intra-class variation and improve generalization performance. There are some obvious temporal invariances that exist in clinical data e.g., the phase invariance of electrocardiogram (ECG) waveforms (i.e., the precise location of the arrhythmia in the ECG reading does not change the interpretation of the arrhythmia [9], [10]). However, the exact parameters of such transformations are generally unknown beforehand and many temporal invariances are likely to be task specific. In contrast to previous work that aligns signals based on some known invariance [11]–[13], our approach, presented in Chapter 3 aims to *learn* how to best align signals [14]. We propose a method to learn the parameterizations for multiple task specific temporal invariances directly from the data. The learned invariances are used to align patient signals and reduce input variance prior to classification.

Signal dynamics. Patient risk stratification tasks often exhibit a specific type of signal dynamic known as *temporal conditional shift* [15], [16]. Temporal conditional shift arises when the dynamics between covariates and outcome change over time. For example, what contributes to patient risk for a particular adverse outcome at the beginning of a hospital admission may differ from what contributes to risk at the end of the admission [17]. We could train individual models for each time period but this would require determining an appropriate delineation between periods and reduce sample efficiency by not leveraging the similarities between sequential tasks. Given sufficient data and model capacity, models such as recurrent neural networks (RNNs), have the potential to learn these dynamics. However, when training data are limited, it may be difficult for models to adapt, especially if they share parameters across time steps as is the case in RNN architectures. In Chapter 4, we propose a novel, RNN-based architecture that relaxes weight sharing by smoothly interpolating between multiple sets of weights.

Low graph homophily When modeling infectious disease transmission dynamics, one often represents the hospital as a graph. One important aspect when learning representations from graphs is the degree of homophily in the graph. Homophily refers to when the nodes in a graph are connected to other nodes of the same class or of similar characteristics [18]. If we represent the hospital as a graph, with patients as nodes and edges connecting co-located patients, we would expect there to be high homophily with respect to infection status. Homophily can be measured by the fraction of intra-class edges in a graph. The majority of nodes are not infected, therefore the majority of edges will connect two uninfected nodes. However, if we only consider the edges involving an infected node, we are likely to observe the opposite: low homophily. Due to the sheer number of uninfected patients, infected patients are likely to have many uninfected neighbors.

Low homophily can cause challenges when applying graph neural networks (GNNs), a popular and successful class of models for learning representations from graph data [19], to low homophily graphs [20]. To date GNNs have been largely benchmarked on data sets with high homophily and often make architectural decisions based on the assumption of high homophily [20]. While our scenario only has low homophily with respect to the minority class, many common architectural assumptions (i.e., mean aggregation, pooling) can be detrimental for similar reasons: most neighbors are unlike the ego node and most neighbors are not relevant to the ego node's prediction. However, existing methods for improving GNN performance under low homophily may not help to the same degree due to the asymmetrical difference in setting. In order to better adapt to this problem scenario, we propose using an attention-based solution. In Chapter 5, we explore the use of graph attention networks [21] to learn pairwise importance of neighbors in order to focus on the few but highly influential contagious neighbors.

1.2 Contributions

To address the limited data challenges that arise when learning from EHR data, we present a series of contributions in this dissertation, summarized here:

- **Learning temporal invariances.**: In Chapter 3, we propose sequence transformer networks, an end-to-end trainable network that learns parameters for patient and task specific temporal transformations directly from the data. These transformations are used to leverage temporal invariances and align patient signals before classification [14]. When evaluated on the task of predicting in-hospital mortality using MIMIC III data [22], we show our proposed approach leads to better generalization performance. Our results suggest that a variety of valuable invariances can be efficiently learned directly from the data. While we demonstrate the utility of the proposed approach in the context of temporal invariances in clinical time series we hypothesize that such techniques apply more broadly. Since publication, a similar approach has been applied to activity detection from joint position data [23].
- **Efficiently learning time varying covariates.** In Chapter 4, we present and explore the setting of temporal conditional shift that considers the scenario of a time-varying relationship between covariates and outcomes. We show evidence for temporal conditional shift on three clinically relevant tasks. We propose mixture of LSTMs to relax LSTM weight sharing across time steps. When evaluated on synthetic and

real-world data exhibiting temporal conditional shift dynamics, our method outperforms baseline methods (*e.g.*, LSTM, HyperLSTM [24]), demonstrating the benefits of relaxed weight sharing [16]. We also show that the mixture of LSTMs model can adapt to settings with limited training data and learn meaningful, time-varying relationships.

- **Addressing class imbalance driven low homophily.** In Chapter 5, we present and explore the setting of class imbalance driven low homophily. In this setting, we compare an attention-based aggregation mechanism to recently proposed solutions to low homophily. Overall, attention proves to be an effective mechanism for addressing low homophily in the minority class. In addition, we demonstrate that GNNs are able to learn useful representations for predicting transmission events in both real clinical data and across a variety of synthetic networks [25]. Such a data-driven approach can outperform approaches based on potentially flawed expert knowledge.

Creating accurate risk estimators from EHR data presents technical machine learning challenges, but if appropriately addressed has the potential to improve clinical decision making. **In this dissertation, we characterize and present problem settings commonly found in healthcare and present novel approaches to improve the generalization performance of deep models trained using EHR data**.

Chapter 2

Background

In this chapter, we present a high-level overview of the clinical and technical background relevant to this dissertation. In the first section, we focus on the clinical background for our work, including applications and clinical tasks. In the second section, we cover topics relating to machine learning. More detailed notation and background is presented on a chapter-by-chapter basis for material specific to subsets of the dissertation.

2.1 EHR Clinical Time Series

Throughout this dissertation we use EHR data for real world evaluations of our methods as well as motivation for developing them. In this section, we briefly review our motivation, what is contained in the EHR, and some special considerations for designing clinical tasks using EHR data. Since the early 1990's, the use of EHR in hospitals has grown, driven by developments in technology, the affordability of data storage and the limitation of paper records [26]. Then the Patient Protection and Affordable Care Act of 2010, mandated that all healthcare practitioners use EHRs [27], furthering adoption. As EHR data become available at more and more institutions, so does the potential impact of methods that leverage the EHR to improve patient care.

The structured contents of the EHR contain data types such as medications, vitals, laboratory results, procedures, locations, demographics, etc. These range from time-varying data (*e.g.*, vitals, medications) to time-invariant data such as height. These data tend to be sparse and irregularly sampled [28]. For example, time stamps for procedures indicate when they occur and are not limited to set intervals. The administration of drugs has a long tail distribution resulting in sparse features. Beyond the structured contents, EHR can contain free text such as doctors notes and patient correspondences. However in this dissertation we will focus entirely on the structured contents.

When designing clinical tasks using EHR data, considerations must be given to accommodate the time-varying and snapshot nature of the EHR. The hospital setting is constantly changing, from EHR vendor to hospital protocols. This is reflected in the EHR. For example, one may use laboratory results to identify outcomes of interest. However, the protocols surrounding ordering laboratory tests can change, along with the false positive rate of the test [29]. Therefore, although there may be decades of data, it may be useful to focus on only the most recent years. EHR data seldom contain the whole picture because they are limited to the time period the patient is in the hospital. For example, if the patient visits another hospital, information about that visit is not readily available and this would limit studies involving recurrence [30]. Lastly, the most crucial consideration is the clinical relevance of a task. It is not uncommon to discuss a machine learning for healthcare paper with a clinician only to conclude that the great performance of the model is useless because the task itself is not useful [31]. In order to develop methods for clinical time series, it is important to understand the clinical and physiological processes that underlie the data generation process and to solve clinically relevant problems using accurate evaluations.

2.2 Clinical Tasks

In this section, we review the datasets and clinical tasks used to develop and evaluate our methods.

2.2.1 Clinical Datasets

We used data from adult, inpatient visits from University of Michigan Hospitals (UM) and MIMIC III [22]. Using the UM dataset facilitates collaboration with clinicians at UM and the potential for developing models that can be implemented in a real hospital setting. MIMIC III is a large, de-identified, publicly available EHR dataset from Beth Israel

Deaconess Medical Center in Boston, Massachusetts and is useful for comparing performance across different published models. We explain each dataset in detail below along with common preprocessing techniques that we used to process each institution's data.

University of Michigan Hospitals (UM)

We focus on adult inpatient admissions to the University of Michigan Hospitals that started and ended during the 4-year period between January 1, 2016, and January 1, 2020. We focus on inpatient visits because they contain denser features and are less impacted by data censoring. This resulted in 268,645 admissions before any additional exclusion criteria. This study population is majority white (81.49%) and female (52.52%) with a median age of 58 years and a median length of stay of 3 days.

MIMIC III

The majority of MIMIC III consists of adult patient visits from critical care units from 2001 to 2012. We considered adult admissions with a single, unique ICU visit. This excludes patients with transfers between different ICUs. Patients without labels or observations in the ICU were excluded. We focused on patients who remained in the ICU for at least 48 hours. Using the full 48 hours allows us to focus on the temporal trends that are more likely to be present in longer visits and a longer time frame for capturing time varying dynamics. This resulted in 21,139 admissions before any additional exclusion criteria.

Preprocessing Techniques

For UM we considered all the structured contents of the EHR. For MIMIC III we extracted 17 physiological features (*e.g.*, heart rate, respiratory rate, Glasgow coma scale, see Appendix A.1) using the same feature extraction procedure as detailed in [2][1]. We briefly describe some of the common aspects of the feature extraction process here. All categorical data (*e.g.*, medications, Glasgow coma scale) were mapped to one-hot feature vectors that indicate if medication A was administered, or if the Glasgow coma verbal score is confused, etc. Continuous values were either mean normalized or mapped to reference

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

ranges (*e.g.*, "normal", "high") and quintiles. Data were resampled uniformly, once per hour for MIMIC III with carry-forward imputation and once per day for UM without imputation. Separate mask features were used to indicate imputed or missing values.

2.2.2 Clinical Outcomes

We consider four different outcomes: acute respiratory failure (ARF), Shock, *Clostridium (Clostridioides) difficile* infection (CDI) and in-hospital mortality. We define each outcome in turn below.

- **ARF.** ARF contributes to over 380,000 deaths in the US per year [3] and represents a challenging prediction task due to its multi-factorial etiology. ARF is defined as the need for respiratory support with positive pressure mechanical ventilation [3], [32].
- **Shock.** Shock refers to the inadequate perfusion of blood oxygen to organs or tissues and can result in severe organ dysfunction and death when not recognized and treated immediately [4].
- ***Clostridium (Clostridioides) difficile* infection (CDI).** CDI is one of the most common healthcare associated infections (HAI) in the United States [5]. There are an estimated 293,300 cases in the US yearly and furthermore, it contributes to increased hospital stay length, higher readmission rate, increased cost per visit and risk of mortality [33]. One of the reasons CDI is so infectious is because it is spread through spores which can survive on contaminated surfaces up to 5 months [6] and cannot be killed using commonly used disinfectants [34]. However, with respect to the hospital population, incidence rates can be low, leading to high class imbalance. During the study period at University of Michigan, 1.04% of admissions were positive for CDI.
- **In-hospital Mortality.** All of these conditions are upstream events that contribute to our fourth prediction task, in-hospital mortality. The ability to identify patients at risk of developing ARF, shock or contracting CDI could facilitate improved patient triage, timely intervention, preventing irreversible damage and ultimately better patient outcomes.

2.3 Deep Learning for Time Series

In this section, we review some deep learning methods that are commonly used for time-series classification. We will reference these methods throughout the dissertation.

2.3.1 Problem Setup & Notation

In Chapters 3 and 4 we consider a supervised learning setup in which we aim to learn a mapping from patient time series to outcome. More formally, given time-series data $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T]$, representing patient covariates over time, where $\boldsymbol{x}_t \in \mathbb{R}^d$, we consider the task of predicting a sequence of outcomes $\boldsymbol{y} = [y_1, y_2, ..., y_T]$, where $y_t \in \mathbb{R}$ for $t \in [1, 2, 3, ..., T]$. Throughout this dissertation we use capitalization to denote matrix notation and bold to denote vectors.

2.3.2 Recurrent Neural Networks (RNNs) and Long Short Term Memory Networks (LSTMs)

RNNs are known for their convenient application to tasks involving time-series data [35]. RNN models have two main traits that contribute to this characterization. First, a RNN model consists of a set of equations, commonly referred to as a RNN cell, that are applied to each time step's input. This allows RNNs to naturally and efficiently accommodate variable length sequences as the cell can be applied repeatedly for additional time steps without learning additional parameters. Second, RNN models have a memory component that is updated at each time step. This memory component informs output as well as how the memory is updated. This allows RNNs to leverage information from any previously seen input and produce outputs that are dynamic and capture temporal dependencies.

LSTMs, one of the most common RNN architectures, are often applied to health data, in part because these data frequently consist of time series and LSTMs can (with enough data) capture complex temporal dynamics [2], [28], [36]. In Chapter 4, we will explore an extension of LSTMs that relaxes weight sharing across time steps.

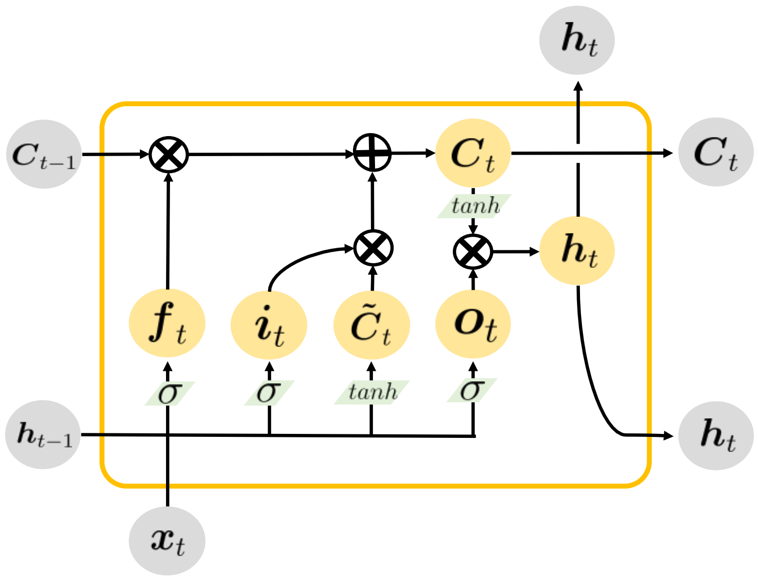


FIGURE 2.1: Visualization of a standard LSTM cell [37]

A standard LSTM cell, which is depicted in **Figure** 2.1, is described formally below,

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W_i}[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b_i}) \tag{2.1}$$

$$\tilde{\boldsymbol{C}}_t = \tanh(\boldsymbol{W_{\tilde{c}}}[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b_{\tilde{c}}}) \tag{2.2}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W_f}[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b_f}) \tag{2.3}$$

$$\boldsymbol{C}_t = \boldsymbol{i}_t * \tilde{\boldsymbol{C}}_t + \boldsymbol{f}_t * \boldsymbol{C_{t-1}} \tag{2.4}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W_o}[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b_o}) \tag{2.5}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t * \tanh(\boldsymbol{C}_t) \tag{2.6}$$

$$\hat{y}_t = \boldsymbol{W}_y \boldsymbol{h}_t + b \tag{2.7}$$

where $*$ represents element-wise multiplication. In an LSTM cell there are two memory components: the hidden state ($\boldsymbol{h}_t$) and the cell state ($\boldsymbol{C}_t$). The size of the memory is determined by d_{hidden} as $\boldsymbol{h}_t$, $\boldsymbol{h}_{t-1}$ $\tilde{\boldsymbol{C}}_t$, $\boldsymbol{C}_t$, $\boldsymbol{C}_{t-1} \in \mathbb{R}^{d_{hidden}}$. $\tilde{\boldsymbol{C}}_t$ acts as the update to the cell state that is combined with the old cell state ($\boldsymbol{C}_{t-1}$) to produce the new cell state ($\boldsymbol{C}_t$). $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, and $\boldsymbol{o}_t$ are referred to as gates. They control the flow of information and due to the sigmoid function ($\sigma(\cdot)$) in each of their equations, the value of each of their elements is within the range $[0, 1]$. All $\boldsymbol{W} \in \mathbb{R}^{d_{hidden} \times (d_{hidden}+d)}$ and $\boldsymbol{b} \in \mathbb{R}^{d_{hidden}}$ with the exception of equation 2.7

where $\boldsymbol{W}_y \in \mathbb{R}^{d_{hidden}}$ and $b \in \mathbb{R}$.

2.3.3 Convolutional Neural Networks (CNNs)

Convolutional neural networks are a popular deep learning method in computer vision. They are known for learning hierarchical feature representations and efficiently learning location invariant patterns. This is achieved through the use of filters that detect certain patterns in the image. Filters are smaller than the input size resulting in efficient weight sharing that takes advantage of the spatial invariance of images. As the filter is passed over the entire image, it finds certain patterns regardless of the pattern's location within the image. Each layer builds upon the outputs of previous filters, resulting in more and more complex patterns being detected in subsequent layers of the network.

In recent years, there has been an increase in research exploring the use of 1-dimensional CNNs in previously RNN dominated tasks in natural language processing and healthcare [11], [12], [38]. We apply CNNs in our time-series classification task in Chapter 3. CNNs allow for parallel computation while RNNs are limited to sequential calculation due to their architecture. This results in much faster training times of CNNs vs RNNs for similar performance [39]. However, CNNs are a less natural fit for time-series analysis and there are many compromises used to retain RNN qualities in CNNs. For example, RNNs can retain memory over the whole input sequence. In CNNs, the field of vision or the range of memory is controlled by the depth of the CNN network, which is determined before training. Therefore, one has to know the longest sequence that will be encountered by the model beforehand.

2.4 Deep Learning for Networks

In Chapter 5 we explore the application of graph neural networks (GNNs), a deep learning method for learning representations from graph or network data.

2.4.1 Problem Setup & Notation

We define graph $\mathcal{G}$ by a tuple $(\mathcal{V}, \mathcal{E})$ that consists of a set of nodes ($\mathcal{V}$) and a set of edges ($\mathcal{E}$). Each node ($v \in \mathcal{V}$) is associated with a feature vector $\boldsymbol{x}_v \in \mathbb{R}^d$. All nodes in $\mathcal{V}$ can be represented via a design matrix $X \in \mathbb{R}^{n \times d}$, where $|\mathcal{V}| = n$. $\mathcal{E}$ can be represented using an adjacency matrix $A \in \{0,1\}^{n \times n}$, where each element $A_{i,j} = 1$ if there exists an edge between v_i and v_j and 0 otherwise.

2.4.2 Graph Neural Networks

Graph neural networks (GNNs) are a popular method for learning feature representations directly from graph data [19]. These models assume that there exists some neighborhood around the ego node that contains the necessary information for a given task. GNNs use a message passing system to calculate an ego node's representation that incorporates messages from all nodes in this neighborhood. Message passing is done iteratively and in a hierarchical manner using network layers. This is illustrated in **Figure** 2.2. Specifically, at each layer k, function $f^{(k)}$ is applied to learn an intermediary representation $\boldsymbol{r}_v^{(k)} \in \mathbb{R}^{d_k}$ of node v:

$$\boldsymbol{r}_v^{(k)} = f^{(k)}(\boldsymbol{r}_v^{(k-1)}, \{\boldsymbol{r}_u^{(k-1)} : u \in N(v)\}; \theta^{(k)}) \tag{2.8}$$

Here $\theta^{(k)}$ are the parameters of $f^{(k)}$ and $N(v)$ is some definition of a neighborhood (i.e., first degree neighbors). $N(v)$ does not need to encompass all relevant nodes for node v's prediction. As f is applied iteratively, the receptive field goes beyond those neighbors that are directly in $N(v)$, to include neighbors of neighbors, neighbors of neighbors of neighbors, etc. At the first layer $\boldsymbol{r}_v^{(0)} = \boldsymbol{x}_v$ and at the final layer, output $\boldsymbol{r}_v^{(K)}$ is used to calculate the final prediction ($\hat{y}_v$). In the case of a classification task, this may be $\hat{y}_v = \arg\max(\log \text{softmax}(\boldsymbol{r}_v^{(K)}\theta))$.

One specific type of GNN that is commonly used is the Graph Convolutional Network (GCN) [40]. At each layer, the GCN uses the previous layer's representation as the "message" (*i.e.*, $\boldsymbol{r}_u^{(k-1)}$) and aggregates all the messages from the ego's neighborhood ($N(v)$) using a normalized average. This aggregated message is transformed to become the ego's new representation $\boldsymbol{r}_v^{(k)}$. The update function $f^{(k)}$ is defined as:

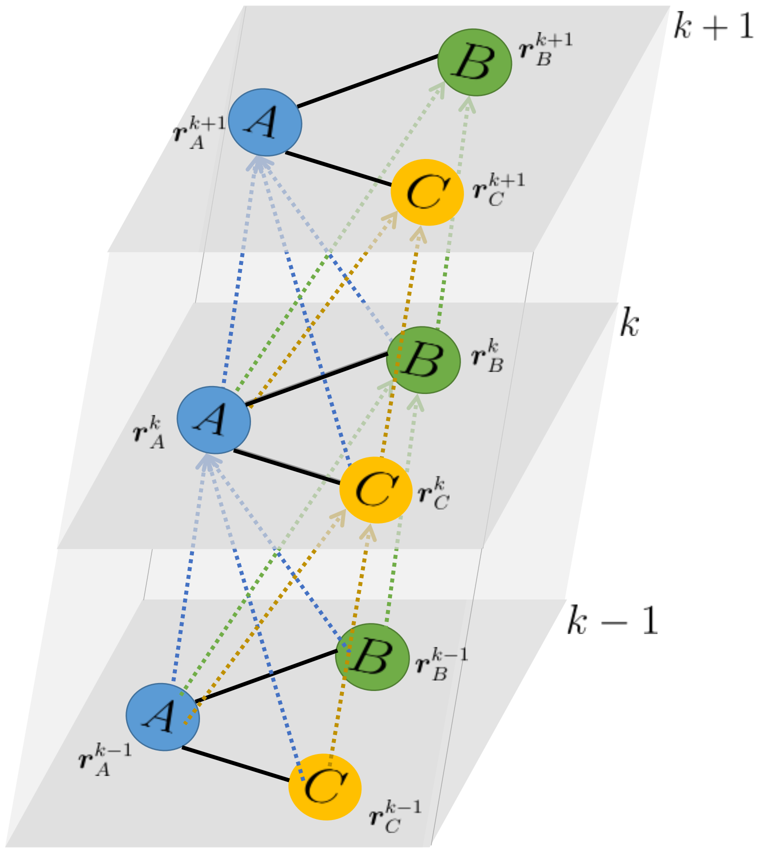


FIGURE 2.2: Visualization of a GNN's k^{th} layer where $N(v)$ is defined as the first degree neighbors of node v and $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{A, B, C\}$ and $\mathcal{E} = \{(A, B), (A, C)\}$. The figure depicts how messages are passed between neighboring nodes between model layers. In this example the messages are the previous layer's representations ($\boldsymbol{r}^{k-1}$).

$$R^{(k)} = f^{(k)}(A, R^{(k-1)}) = ReLU(D^{-\frac{1}{2}} A D^{-\frac{1}{2}} R^{(k-1)} \theta^{(k)}), \tag{2.9}$$

Here we represent the matrix of stacked intermediary representations as $R^{(k)} \in \mathbb{R}^{n \times d_k}$ and D is the diagonal node degree matrix of A. In this formula, the $R^{(k-1)}$ represents the "messages", $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ represents the weights for the normalized average, and the $ReLU()$ and $\theta^{(k)}$ represent the transformation.

Chapter 3

Sequence Transformer Networks

3.1 Introduction

Clinical time-series data consist of a wide variety of repeated measurements/observations, from vitals (*e.g.*, heart rate) and laboratory results to locations within a hospital [28], [41], [42]. These data vary not only in the information they encode, but also in sampling rate and number of measurements. Analogous to how certain tasks in computer vision exhibit spatial invariances, invariances frequently arise in clinical tasks involving time-series data. These invariances describe a set of transformations that, when applied to the data, magnify task-relevant similarities between examples. For example, phase invariance relates to a transformation that shifts a signal, resulting in an alignment in phase. Such transformations can be particularly useful when processing periodic signals *e.g.,* electrocardiogram waveforms [10].

Preprocessing techniques like dynamic time warping are commonly used to exploit warping invariances and align time-series data, facilitating relevant comparisons [43], [44]. However, their computational complexity (*e.g.*, DTW involves solving an optimization problem for each new example) may be a factor leading to their limited use within more general settings. In addition, such approaches require *a priori* knowledge of the types of invariances that are present in one's data. Due to the varied nature of clinical time-series data and their associated prediction tasks, we expect that many such tasks involve multiple invariances that may not be known beforehand. This and the fact that

The following is an adaptation of previously published work [14].

these invariances are likely task specific, are some of the main roadblocks in efficiently exploiting these invariances.

Our Approach. To addresses these challenges, we propose Sequence Transformer Networks, an approach for learning task-specific invariances related to amplitude, offset, and scale invariances directly from the data. Our approach consists of an end-to-end trainable framework designed to capture temporal and magnitude invariances. Applied to clinical time-series data, Sequence Transformer Networks learn input- and task-dependent transformations. In contrast to data augmentation approaches, our proposed approach makes limited assumptions about the presence of invariances in the data. Learned transformations can be efficiently applied to new input data, leading to an improvement in overall predictive performance. We demonstrate the utility of the proposed approach in the context of predicting in-hospital mortality given 48 hours of data collected in the intensive care unit (ICU). Relative to a baseline that does not incorporate any transformations, Sequence Transformer Networks result in significant improvements in predictive performance. Our main contributions can be summarized as follows:

- We propose the use of Sequence Transformer Networks, an end-to-end trainable framework designed to capture temporal and magnitude invariances.
- On a real data task, we evaluate the relative contribution of each individual component of Sequence Transformer Networks towards the overall performance of the network.
- We present visualizations of the types of learned invariances and investigate the effects of Sequence Transformer Networks on intra-class signal similarity.

Organization. The remainder of the chapter is organized as follows. First, we give a background on how previous work has leveraged invariances. Then, we present our proposed method: Sequence Transformer Networks. Next, we describe our clinical task and evaluation details. Finally, we present results on a clinical task demonstrating the utility of levering learned invariances from time series data.

3.2 Background

Tasks involving time-series data may exhibit a number of different invariances. We refer the reader to the following paper for an in-depth discussion of types of invariances present in time-series data [45], but for completeness include a summary of common invariances in Table 3.1. To exploit these invariances, researchers often turn to neural networks. In particular, one-dimensional (1D) convolutional neural networks (CNNs), by design, efficiently exploit phase invariance. This property, in addition to their computational efficiency achieved by weight sharing, has led to their successful application to a variety of tasks involving sequential data [13], [38], [46]–[48], and more specifically clinical time-series data [11], [12], [49], [50]. Recognizing that clinical time-series data exhibit other types of invariance, beyond phase invariance, we propose augmenting CNNs to explicitly account for task-irrelevant variation.

In other domains, to exploit invariances researchers either i) augment their training data by applying a variety of transformations or ii) modify the neural network architecture. The first approach is most popular in domains where it is straightforward to generate realistic training examples (*e.g.*, natural images). Common image invariances include rotation, scale, translation and warping. Such transformations are easily applied to existing images to create additional, realistic training examples. While less common in the healthcare domain, there have been successful examples of data augmentation for health data. For example, [51] augmented multivariate time-series data collected from a wearable sensor placed on a person's wrist in order to improve monitoring of patients with Parkinson's disease. The authors applied transformations such as noise and rotations, selected based on the task. However, in general it is not straightforward to apply such data augmentation schemes to clinical data because of the large number of potential invariances. Moreover, clinical time-series data extracted from electronic health records often consist of high-dimensional data measuring many different aspects of a patient's health. This increases the complexity of identifying reasonable transformations and makes a brute-force search over possible transformations computationally intractable.

Our work is more in-line with the second approach that does not rely on additional data. Instead, the architectures are modified to exploit a particular invariance [11]–[13], [46], [52], [53]. For example, in [11] and [12], the authors tackle warping by using multiple filter sizes. More specifically, three different sized filters were used to capture a range of long- and short-term temporal patterns. These different resolutions corresponded to separate convolutional layers, combined at the final fully connected layer. [13] propose an additional preprocessing step, in which they resample and smooth their input in order to capture multiscale patterns and remove noise. Transformed versions of the inputs were treated as additional channels to the original image. Similar to [11], [12], this method incorporates a local convolution stage that looks at each type of transformation (none, smoothing, down-sampling) independently before combining. Both of these works are geared toward specific invariances, in this case scale invariance, and require the user to determine the different filter sizes or sampling rates.

Recognizing the difficulty in identifying potential invariances or transformation *a priori*, we focus on learning the invariances directly from the data. Our proposed approach extends work by [54], in which a spatial transformer network is used to learn spatial invariances directly from the data. In [54], the parameters of a spatial transformer network are learned jointly with the parameters of a CNN. The transformer network applies a learned set of transformations including affine transformations tailored to each input before passing it through a CNN. Since we focus on clinical time-series, and not images, we adapt the set of possible transformations. Specifically, our proposed method tackles amplitude and offset invariances (which we will refer to as magnitude invariance), phase invariance, and uniform scale invariance, and learns input-specific transformation parameters directly from the data. We describe the details of our approach in the next section.

3.3 Methods

3.3.1 Problem Setup

We consider the application of 1D CNNs to clinical time-series data for predicting a specific outcome. Formally, given a set of n labeled examples consisting of d features measured at T time steps ($X \in \mathbb{R}^{n \times d \times T}$) and the outcome labels $\boldsymbol{y} \in \{0,1\}^n$, our goal is to learn a mapping from $\{\boldsymbol{x}_t^{(i)}\}_{t=1}^T$ to $y^{(i)}$, where $\boldsymbol{x}_t^{(i)} \in \mathbb{R}^d$ and $i \in \{1 \cdots n\}$ is an index into the i^{th} sample. The d features may consist of both time-varying and time-invariant data.

TABLE 3.1: A list of possible invariances summarized from [45]. Any number or combination of invariances may arise in clinical time-series data, or time-series data in general.

Invariance	Description
Amplitude	A transformation of the amplitude of the time series. This can occur when the scale or unit of measurement of two time series differs (*e.g.*, temperature in Celsius vs. Fahrenheit).
Offset	A transformation that uniformly increases/decreases the value of a time series. For example, two patients may have different resting heart rates.
Local Scaling ("Warping")	A transformation that locally stretches or warps the duration of the time series. Local warping is often referenced in conjunction with Dynamic Time Warping (DTW), a good, established measure of similarity between time series with local scaling invariance.
Uniform Scaling	A transformation that globally stretches the duration of the time series. For example, when resting heart rates differ between patients, the progression of the same temporal pattern may be consistently slower in one patient versus another.
Phase	A transformation that shifts the start time of a time series. This occurs in periodic signals such as heartbeat and blood pressure waveforms.
Occlusion	A transformation that randomly removes data. This can arise when measurements are irregularly sampled or missing.
Noise	A transformation that adds or removes noise. For example, many single point sensors are susceptible to noise that might not be indicative of the whole body's condition but indicative of that sensor's particular location.

We represent each feature as a set of T measurements. For time-varying data for which we do not have a measurement at time t, we carry forward the most recent value. For time-invariant data, we copy the measurement across all T time-steps as in [36]. Additional details pertaining to the specific dataset used through our experiments can be found in Section 3.4.

3.3.2 Sequence Transformer

Applied to time-series data, 1D convolutions inherently capture some invariance in the data. In particular, CNNs are capable of efficiently handling phase invariance (*i.e.*, the use of a filter slid along the temporal dimension allows for variability in the starting point of temporal patterns.) CNNs also handle noise invariance, to a degree. Max pooling coupled with multiple layers allows the model to smooth the inputs and learn higher-level abstractions.

However, there are other types of invariances that we would like to consider, in particular temporal invariance such as scaling, in addition to magnitude invariance related to the amplitude and offset of the signal. Figure 3.1 shows examples of these types of invariances on a sine wave. Due to the inherent differences between these types of invariances, we address them separately in the two subsections that follow. For simplicity, in this section, methods are presented in terms of a univariate signal, but later our experiments focus on a multivariate application.

Temporal Transformations

To capture invariance related to warping and scaling, we begin by learning to transform data along the temporal dimension. As in [54], this stage consists of two separate pieces i) learning the transformation parameters and ii) mapping those transformations in terms of discrete data points. We discuss each, in turn, below.

Transformation Network. We begin by learning a transformation that takes points from the original input (*i.e.*, the source) and maps them to a new temporal location in the target. Since we only consider linear transformations along the temporal axis, we respect

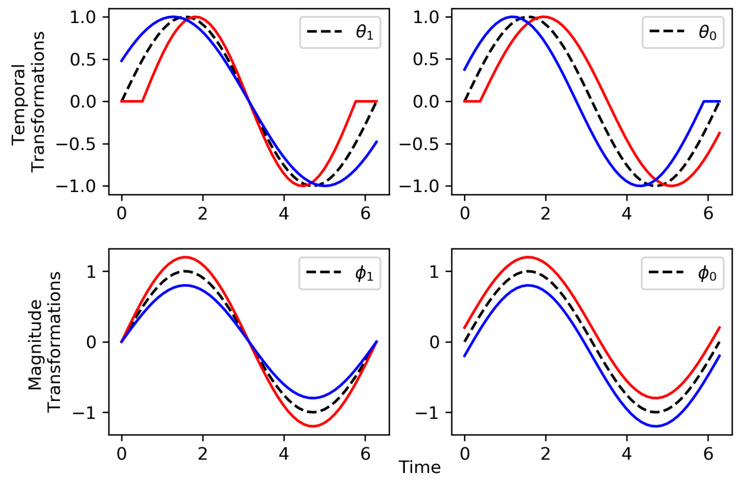


FIGURE 3.1: Examples of the types of transformations/invariances that can be learned by a Sequence Transformer Network applied to a sine wave. θ_1: scaling invariance, θ_0: phase invariance, ϕ_1: amplitude invariance, ϕ_0: offset invariance. The dashed line represents the original signal and the blue lines represent potential transformations of the signal. While CNNs can efficiently exploit phase invariance, Sequence Transformers can augment other types of architectures facilitating the capture of other types of invariances.

the ordering of values, but can stretch, compress, flip and/or shift the signal (across the temporal axis).

$$t = \boldsymbol{\theta}^{(i)} \begin{pmatrix} t' \\ 1 \end{pmatrix} = \begin{pmatrix} \theta_1^{(i)} & \theta_0^{(i)} \end{pmatrix} \begin{pmatrix} t' \\ 1 \end{pmatrix} \tag{3.1}$$

Equation (3.3.2) gives a mapping between the transformed time point t' and original time point t. Given a univariate time-series $\{x_t^{(i)}\}_{t=1}^T$, t represents the t^{th} position along the temporal axis of the time-series. We learn a linear temporal transformation $\boldsymbol{\theta}^{(i)} \in \mathbb{R}^{n\times 2}$ that applies to these indices. Specifically, we generate t' for $t' = 1, ..., T'$. T' represents the length of the transformed sequence and can be set to any positive integer. Here, for convenience, we set $T = T'$. The transformation parameters $\boldsymbol{\theta}^{(i)}$ are learned via a two-layer CNN that is fed inputs $\{x_t^{(i)}\}_{t=1}^T$. Network architecture details are outlined in Figure 3.2. Given a particular position, t', in the target time series, we compute the corresponding position in the original time series and set $x_{t'}$ to refer to $x_{t=\theta_1 t'+\theta_0}$.

Discrete Mapping. Since $\theta_1 t' + \theta_0$ for $t' = 1, ..., T'$ is not guaranteed to map to a positive integer (*i.e.*, an index), we require an additional step to apply the learned transformation. We complete the mapping using linear sampling, in which we take an average over the two nearest neighbors (one from left, one from the right) weighted by the distance from the original transformed point.

Magnitude Transformations

In order to adapt to amplitude and offset invariance, we propose an additional learned transformation, one that is applied to the values instead of the coordinates. Given the temporally transformed inputs $\{x_{t'}^{(i)}\}_{t'=1}^{T'}$, we apply the following linear transformation:

Signals are padded by the last known value so there is no edge case where a point has only one neighbor.

$$x'^{(i)}_{t'} = \boldsymbol{\phi}^{(i)} \cdot x^{(i)}_{t'} = \begin{pmatrix} \phi^{(i)}_1 & \phi^{(i)}_0 \end{pmatrix} \cdot \begin{pmatrix} x^{(i)}_{t'} \\ 1 \end{pmatrix} \tag{3.2}$$

This allows us to shift, flip, stretch, and compress the signal along its magnitude. Since this transformation applies directly to the values of the signal, we do not require a discrete mapping component. It should be noted that the transformation, $\boldsymbol{\phi}^{(i)} \in \mathbb{R}^{n\times 2}$ is a function of x, thus it can vary from example to example.

Sequence Transformer

We refer to the temporal transformation combined with the magnitude transformation as a Sequence Transformer (Figure 3.2). The Sequence Transformer computes both the $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ transformation parameters based on the input and applies them to the signal.

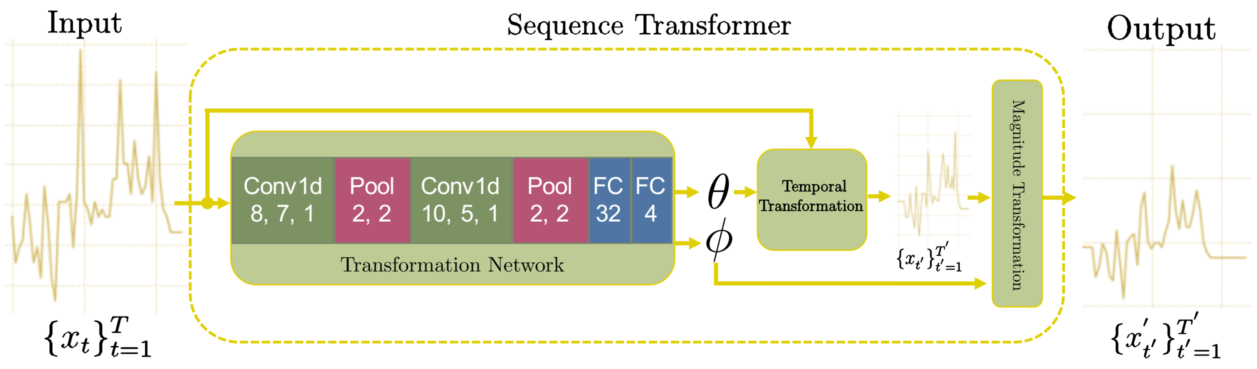


FIGURE 3.2: The architecture of a Sequence Transformer. Inputs $\{x_t\}_{t=1}^{T}$ (shown as univariate for illustration purpose) are fed into a Transformation Network that outputs transformation parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. Convolutional and maxpool layers are annotated with the number of outputted channels (omitted for maxpool), filter size and stride. Fully connected layers (FC) are annotated with the number of neurons. The temporal transformation is applied via discrete mapping and the magnitude transformation is applied via linear transformation. The output $\{x'_{t'}\}_{t'=1}^{T'}$ represents the transformed inputs.

While we presented this approach in the context of a univariate signal, the technique generalizes to multivariate signals. In a multivariate setting, the Transformation Network outlined in Figure 3.2 takes as input $\{\boldsymbol{x_t}\}_{t=1}^{T}$, where $\boldsymbol{x_t} \in \mathbb{R}^d$. The Transformation Network then estimates $[\boldsymbol{\theta}, \boldsymbol{\phi}]$, based these data and the underlying model parameters. Although the model parameters are consistent across all examples, the resulting *transformation* parameters (*i.e.*, $\boldsymbol{\theta} \in \mathbb{R}^{n\times 2}$ and $\boldsymbol{\phi} \in \mathbb{R}^{n\times 2}$) are specific to each example. This transformation is then applied to all signals in the input (note that temporal transformations have no effect on time-invariant data, but these signals can still be transformed in a meaningful way).

3.4 Experimental Setup

In this section, we describe our dataset and prediction task, the baseline CNN architecture and implementation details.

3.4.1 Dataset & Prediction Task

To measure the utility of the proposed approach on a real dataset, we consider a standard sequence-level classification task: predicting in-hospital mortality based on the first 48 hours of data collected during an intensive care unit visit. We use data from MIMIC III [22]. As in [2], we consider adult admissions with a single, unique ICU visit. This excludes patients with transfers between different ICUs. Patients without labels or observations in the ICU were excluded, as were patients who died or were discharged before 48 hours. After applying exclusion criteria, our final dataset included 21,139 patient admissions and 2,797 deaths.

We used the same feature extraction procedure as detailed in [2]. Code to generate these data are publicly available. For completeness, we briefly describe the feature extraction process here. For each admission, we extracted 17 features (*e.g.*, heart rate, respiratory rate, Glasgow coma scale) from the first 48 hours of their ICU visit. We applied mean normalization and discretization, resulting in 59 features. Sampling rates were set uniformly to once per hour using carry-forward imputation. Mask features, indicating

https://github.com/YerevaNN/mimic3-benchmarks if a value had been imputed resulted in an additional 17 features. After preprocessing, each example was represented by $d = 76$ time-series of length $T = 48$ and a binary label indicating whether or not the patient died during the remainder of the hospital visit.

Given these data, the goal is to learn a mapping from the features to the probability of in-hospital mortality, resulting in a single prediction per patient admission. We measured performance by calculating the area under the receiver operating characteristic curve (AUROC) and the area under the precision recall curve (AUPR). We randomly split the data into training (70%), validation (15%), and testing (15%): 14,681 (1,987 deaths) in training, 3,222 (436 deaths) in validation and 3,236 (374 deaths) in test. We learned model parameters and selected hyperparameters using training and validation data and evaluated model performance using held-out test data. Specifics on hyperparameter search are presented in Section 3.4.3. We generated empirical 95% confidence intervals by bootstrapping the test set.

3.4.2 Baseline CNN Architecture

As a baseline with which to compare, we considered a CNN without any additional Sequence Transformer. We compared the discriminative performance of a CNN with original inputs to a CNN with inputs transformed via the Sequence Transformer. We referred to the first method as our Baseline CNN. The second is our proposed method: Sequence Transformer Networks. The only difference between this baseline and our proposed approach is the Sequence Transformer (Figure 3.3). Both models feed either the original or transformed example into a standard 1D CNN. For this CNN, we used the two layer CNN described in Figure 3.3. The CNN consists of two 1D convolutional and pooling layers followed by a single, hidden, fully connected layer.

In addition to considering a baseline consisting of no transformations, we also considered networks that used either i) temporal transformations only or ii) magnitude transformations only. This allowed us to measure the marginal contribution of each transformation in the Sequence Transformer.

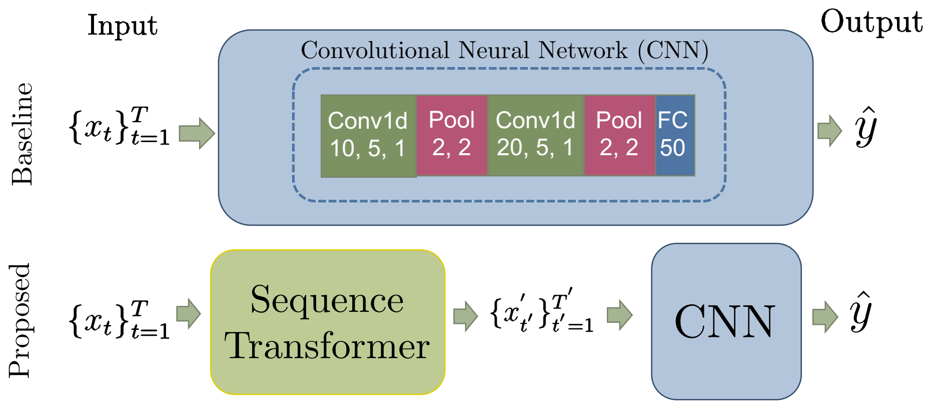


FIGURE 3.3: The architecture of the CNN. Baseline CNN inputs ($\{x_t\}_{t=1}^T$) or Sequence Transformer inputs ($\{x'\}_{t'=1}^{T'}$) are fed into a standard CNN that outputs our in-hospital mortality prediction. Here, the admission indexing i is omitted for simplicity. Convolutional and maxpool layers are annotated with the number of outputted channels (omitted for maxpool), 1D filter size and stride. Fully connected layers (FC) are annotated with the number of neurons.

3.4.3 Implementation Details

We optimized the following hyperparameters: network depth, number of neurons in the final fully connected hidden layer, batch size, and dropout rate. We trained twenty models with randomly selected hyperparameters, for at most 10 epochs. Hyperparameters were randomly chosen from predefined sets of values. Batch size was randomly selected from: 8, 15, 30. The rate of dropout was randomly selected from: 0, .1, .2, . . . , .9. We tested CNN architectures of depth 2, 3 and 4. Finally, the number of neurons in the final fully connected hidden layer was randomly chosen from: 50, 100, 250 and 500. The settings that led to the best performance on the validation data are shown in Figure 3.3.

Since these hyperparameters were tuned for our Baseline CNN using the original input, we also considered a model tuned to the transformed signal. The resulting optimal hyperparameters were largely unchanged, except that we found that a dropout rate of 0.2 (vs. 0.3) worked better for Sequence Transformer Networks. The optimal batch size for both models was 15.

During model training, we included gradient clipping. This consisted of a reduced slope from 1 to .01 outside of a reasonable range of transformation parameter values. In practice, we set this range to $[-2, 2]$. We found this implementation detail to be important. Without it, we witnessed quick increases in the value of the transformation parameters that led to unrecoverable model states.

3.5 Experiments & Results

We present the performance of the Baseline CNN, which takes as input untransformed signals as described in Section 3.4.2, vs. Sequence Transformer Networks. We further break down the Sequence Transformer into its two parts: temporal and magnitude transformations and evaluate their individual contributions. Finally, we investigate the learned transformations through a series of visualizations and analyze the effect of Sequence Transformer Networks on intra-class signal similarity. In our experiments, we seek to answer the following questions:

- Question 1: Are Sequence Transformer Networks able to learn useful transformations directly from the data? (**Table 3.2**, **Section 3.5.3**)
- Question 2: Are the two types of learned transformations (temporal and magnitude) complementary? (**Table 3.2**)
- Question 3: What kind of transformations are being learned? (**Section 3.5.2**)

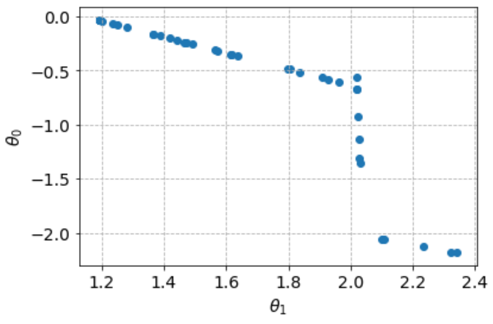
3.5.1 CNN Baseline vs Sequence Transformer Networks

Our proposed method, Sequence Transformer Networks, outperforms the Baseline CNN, in terms of both AUROC and AUPR, on the task of predicting in-hospital mortality using data from the first 48 hours (Table 3.2).

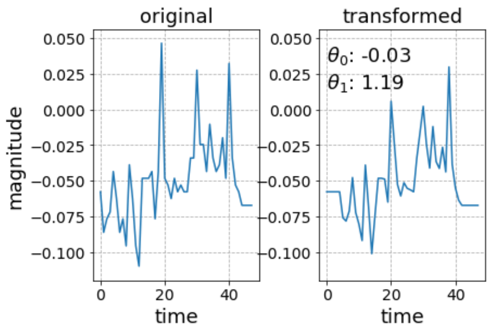
Compared to the Baseline CNN, Sequence Transformer Networks incorporates a secondary, transformation network. However, the improvement in performance is not due to the additional complexity of the model. For both models, we tuned the depth of the CNN architecture. In both cases, the best CNN, determined by validation performance and presented in the results, had a network depth of 2. Therefore a deeper network alone is not sufficient for increasing performance.

TABLE 3.2: Test performance for the task of predicting in-hospital mortality. Relative to the baseline performance, transforming the input before feeding it into the CNN results in consistent improvements in both the area under the receiver operating characteristics curve (AUROC) and the area under the precision recall curve (AUPR).

Method	AUROC (95% CI)	AUPR (95% CI)
Baseline CNN	0.838 (0.820, 0.859)	0.445 (0.393, 0.495)
Sequence Transformer Networks	**0.851** (0.833, 0.871)	**0.476** (0.424, 0.527)
Temporal Transformations Only	0.846 (0.827, 0.867)	0.452 (0.393, 0.500)
Magnitude Transformations Only	0.846 (0.826, 0.867)	0.463 (0.408, 0.516)



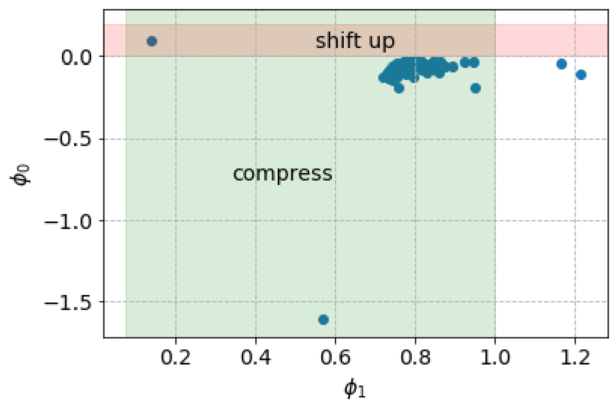
(A)



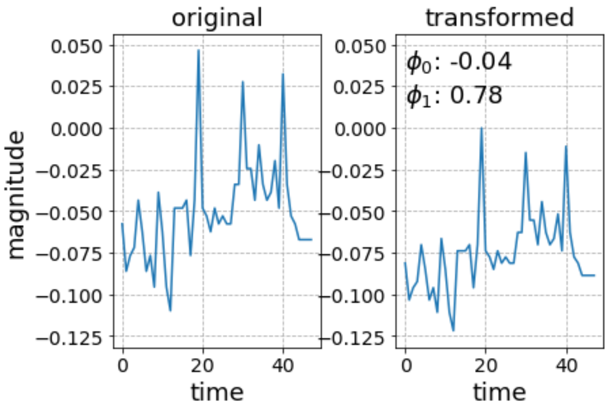
(B)

FIGURE 3.4: Sequence Transformer Networks: Temporal Transformations Only. **(a)** Visualization of temporal transformation parameters applied to the test set. Note that $\theta_1 \geq 0$ indicates signal compression, while $\theta_0 \leq 0$ indicates shifting the signal forward in time. **(b)** A random test patient's normalized diastolic blood pressure before and after $\boldsymbol{\theta}$ transformation ($\theta_1 = 1.19$, $\theta_0 = -0.03$). In addition to signal compression and shifting, the network smooths the signal.

Since the Sequence Transformer consists of two transformations, we further break down the performance increase into: temporal transformations and magnitude transformations. In Table 3.2, we see that both types of transformations lead to marginal improvements over the baseline. Moreover, their combination appears to be complementary, though the difference is small.



(A)



(B)

FIGURE 3.5: Sequence Transformer Networks: Magnitude Transformations Only. **(a)** Visualization of magnitude transformation parameters applied to the test set. Note that $\phi_1 \leq 1$ indicates signal value compression, while $\phi_0 \leq 0$ indicates a downward shift. **(b)** A random test patient's normalized diastolic blood pressure before and after $\boldsymbol{\phi}$ transformation ($\phi_1 = 0.78$, $\phi_0 = -0.04$).

3.5.2 Learned Temporal and Magnitude Transformations

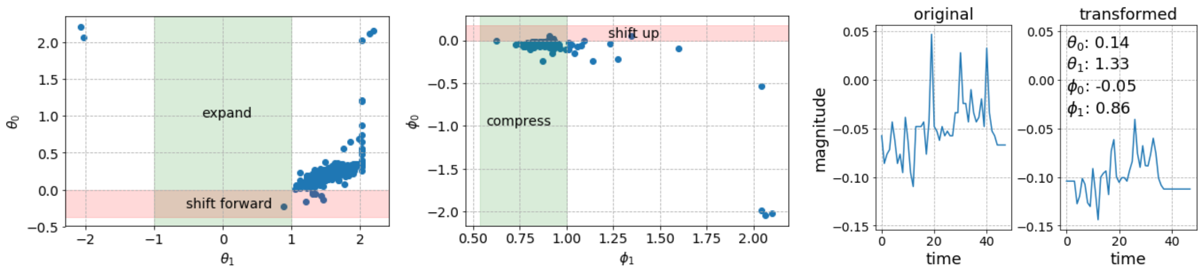
In this section, we qualitatively explore what the Sequence Transformer has learned. Figure 3.4 summarizes the transformation learned using a network that employs only temporal transformations. Recall that the transformation depends on the input. Figure 3.4a shows the empirical distribution of the two temporal transformation parameters (θ_1, θ_0). Each point represents a temporal transformation learned for a specific patient admission in the test set. In this case, most of the data occur around $\theta_1 = 1.19$ and $\theta_0 = -0.03$. Essentially, the network learns to compress the original signal ($\theta_1 \geq 1$) and shift the signal forward in time ($\theta_0 \leq 0$) by various degrees. In doing so, the network learns how to align the time-series data from different patient admissions. Figure 3.4b shows the original and the temporally transformed normalized diastolic blood pressure for a randomly selected patient in the test set. In line with the results shown in the previous figure, the signal is compressed along the x-axis and shifted forward in time. In Figure 3.4b, though the signal is moved forward in time, it is not clipped, but rather compressed. This suggests that θ_0 is helping to center the signals. The sudden drop off at $\theta_1 = 2$ is most likely due to the gradient clipping, since that is where it begins to take effect. In addition, we observe a smoothing effect that is due, in part, to the the linear interpolation.

Figure 3.5, shows the same type of plots as Figure 3.4 but for a network that includes only magnitude transformations. We observe that the signal is, on average, shifted down and compressed. Similar to the temporal transformations, the magnitude transformations help align signals. Amplitude and offset invariances have a clinical significance for many features in this dataset including blood pressure, heart rate, respiratory rate and temperature. We hypothesize that these transformations help account for different physiological baselines.

Finally, we visualize the output of the Sequence Transformer, which learns temporal, amplitude and offset invariances together (Figure 3.6). In Figures 3.6a and 3.6b, each point represents a transformation learned for a specific patient in the test set. We see that the network, on average, compresses the signal and shifts it slightly back in time. In the temporal transformation only network (Figure 3.4), the network shifted signals forward in time. This suggests that the direction of the shift is less important than the overall alignment of the different patients. For magnitude transformations, the network on average compresses the signal and shifts it down. These learned transformation trends align with the magnitude transformation trends learned separately (Figure 3.5). In Figure 3.6c we illustrate the transformations applied to a random test patient's normalized diastolic blood pressure.

3.5.3 Increasing Intra-Class Similarity

Sequence Transformer Networks have the ability to learn transformations that reduce label independent variations in the signal. By reducing irrelevant variance, transformed signals from patients with similar outcomes then appear more similar. We investigate this



(A) (B) (C)

FIGURE 3.6: Sequence Transformer Networks **(a)** Visualizations of temporal transformation parameters applied to the test set. On average, network compresses and shifts signals backwards in time. **(b)** Visualizations of magnitude transformation parameters applied to the test set. On average signal values are compressed and shifted down. **(c)** The learned network smooths and shifts the normalized diastolic blood pressure to the left bottom direction of the frame for a randomly selected patient using the transformations: $\theta_1 = 1.33$, $\theta_0 = 0.14$, $\phi_1 = 0.86$ and $\phi_0 = -0.05$.

property by analyzing the intra-class Euclidean pairwise distance. On each dataset (original vs. transformed), we calculated the Euclidean pairwise distance between admissions labeled positive and the Euclidean pairwise distance between those labeled negative.

The transformed dataset had on average lower pairwise intra-class distances compared to the original (untransformed) data (positive: 28.2 vs. 34.9 and negative: 26.3 vs. 31.8). We hypothesize that this increase in intra-class similarity contributes to the overall improved discriminative performance of the Sequential Transformer Network over the Baseline CNN.

3.6 Summary and Conclusions

In this chapter, we proposed the use of an end-to-end trainable method for exploiting invariances in clinical time-series data. Building off of ideas first presented in the context of transforming images, we extended the capabilities of CNNs to capture temporal, amplitude, and shift invariances. In general, such invariances may be task dependent (*i.e.*, may depend on the outcome of interest or the population studied). Given the large number of possible clinical tasks, techniques that automatically learn to exploit invariances based on the data have a clear advantage over preprocessing techniques.

We demonstrated that this method leads to improved discriminative performance over the Baseline CNN, on the task of predicting in-hospital-morality from multivariate clinical time-series data collected during the first 48 hours of an ICU admission. Though the difference in performance is small, the improvement is evident across both AUROC and AUPR.

The proposed approach is not without limitation. More specifically, in its current form the Sequence Transformer applies the same transformation across all features within an example, instead of learning feature-specific transformations. Despite this limitation, the learned transformations still lead to an increase in intra-class similarity. In conclusion, we are encouraged by these preliminary results. Overall, this work represents a starting point on which others can build off of. In particular, we hypothesize that the ability to capture local invariances and feature-specific invariances could lead to further improvements in performance. Since publication, more recent work [23] has looked at capturing local invariances by learning resampling/warping functions directly from the data.

Chapter 4

Temporal Conditional Shift

4.1 Introduction

Recurrent neural networks (RNNs) capture temporal dependencies between past inputs $x_{1:t-1}$ and output y_t, in addition to the relationship between current input $\boldsymbol{x}_t$ and y_t. Their successful application to date is due in part to their explicit parameter sharing over time [2], [55]. However, while advantageous in many settings, such parameter sharing could hinder the ability of the model to accurately capture time-varying relationships, i.e., tasks that exhibit temporal conditional shift.

In healthcare, temporal condition shift may arise in clinical prediction tasks when the factors that put a patient at risk for a particular adverse outcome at the beginning of a hospital visit differ from those that put a patient at risk at the end of their stay. Failure to recognize conditional shift when building risk stratification models could lead to temporal biases in learned models; models may capture the average trend at the cost of decreased performance at specific points in time. This could be especially detrimental to models deployed and evaluated in real time.

More formally, conditional shift refers to the change in the conditional distribution $P(Y = y|X = \boldsymbol{x})$ across tasks. In particular, we consider *temporal* conditional shift, i.e., the setting in which the relationship between $\boldsymbol{x}$ and y is a function of both $\boldsymbol{x}$ and time ($y_t = f(\boldsymbol{x}, t; \theta_t)$). We hypothesize that RNN's complete sharing of parameters across time steps makes it difficult to accurately model temporal conditional shift. To address this, one could jointly learn a different cell for each time step, but such an architecture may

The following is an adaptation of previously published work [16].

easily lead to overfitting. More importantly, such an approach does not leverage the fact that many relationships are shared, at least in part, across time.

Our Approach. On synthetic data, in which we can control the amount of conditional shift, we explore the trade-offs in performance between models that share parameters across time versus models that do not. Beyond synthetic data, we illustrate the presence of temporal conditional shift in real clinical prediction tasks. To tackle this issue, we propose a novel RNN framework based on a mixture approach that relaxes parameter sharing over time, without sacrificing generalization performance. Applied to three clinically relevant patient risk stratification tasks, our proposed approach leads to significantly better performance relative to a long short-term memory network (LSTM). Moreover, the proposed approach can help shed light on task relatedness across time. Our main contributions can be summarized as follows:

- We formalize the problem setting of temporal conditional shift.
- We illustrate the presence of temporal conditional shift in three clinically relevant tasks.
- We propose a novel approach for relaxed parameter sharing within an RNN framework.
- We explore situations in which relaxed parameter sharing can help.

In theory, given enough data, RNNs should be able to accurately model relationships governed by temporal conditional shift. However, oftentimes in clinical applications, we have a limited amount of data to learn from. Going forward, researchers should check for the presence of conditional shift by comparing the proposed approach with an LSTM. If conditional shift is detected, then one may be able to more accurately model temporal dynamics through relaxed parameter sharing.

In the previous chapter, we looked at exploiting invariances. While invariances and temporal conditional shift may seem to be contradictory concepts, they are actually complementary. Exploiting invariances can be seen as a preprocessing step where we learn transformations that better align clinical time series across patients. In the temporal conditional shift problem setting, we assume that the time series are already well aligned and instead explore how to model the changing relationship between X and Y within a sequence.

Organization. The remainder of the chapter is organized as follows. First, we give a background on temporal conditional shift and existing methods for adapting to this setting. Then, we present methods by which one can relax parameter sharing in LSTMs, including our proposed approach `mixLSTM`. Next, we describe our clinical tasks and baselines. Finally, we examine the relationship between temporal conditional shift and shared weights and the benefits of relaxed weight sharing in this context.

4.2 Background

We focus on developing techniques that can handle temporal conditional shift, a type of data shift that commonly occurs in tasks involving clinical time-series data. There are two main types of data shift: i) covariate shift and ii) conditional shift. Covariate shift is the scenario where $P(X = \boldsymbol{x})$ varies across datasets [56], [57], e.g., the distributions of patient demographics may differ across study populations. In contrast, conditional shift, our main focus, occurs when $P(Y = y|X = \boldsymbol{x})$ changes [15], [58], e.g., two hospitals may have similar patient populations, but different factors could drive patient risk due to differences in clinical protocols. In conditional shift, the relationship between input $\boldsymbol{x}$ and output y has shifted. This can occur independently of a change in population. For some time, the study of data shift has driven research in the fields of domain adaptation, transfer learning, and multitask learning [59]–[62].

Methods for dealing with conditional shift are largely driven by the problem setting. Researchers have explored the use of pre-trained features [63], generalizable representations [64], [65], and applying importance re-weighting techniques [15]. In contrast to these works, we focus on techniques for tackling conditional shift in which the shift is driven by changes in time. In this setting, there is no clear distinction between tasks, because the change occurs gradually. Though related, this differs from 'data drift' (i.e., the setting in which relationships change longitudinally) since we consider time on a local/relative scale as opposed to a global/absolute scale [66], [67]. That is, instead of focusing on differences between 2018 and 2019, we focus on changes within an admission or a patient.

Though such local shift is expected to occur [68], [69], it is often overlooked when modeling patient risk.

In the linear setting, past work has explored the use of multitask learning to model the temporal evolution of risk factors within a patient admission, where each day corresponds to a different model, but models are learned jointly [17]. Related, [69] proposed a variation to Cox regression analysis, studying different time windows separately and using time specific hazard ratios.

Nonlinear methods designed to explicitly deal with temporal conditional shift have more recently been explored, focusing primarily on modifications to RNN architectures [24], [70], especially LSTMs. For example, [24] proposed an extension to LSTMs, Hypernetwork, that relaxes parameter sharing by learning an auxiliary network that sets the primary network's parameters at each time step. Specifically, the auxiliary network can change the primary network's parameters through a scalar multiplier. Similar to Hypernetworks, we also consider a variation on the LSTM, in part because LSTMs are commonly applied to clinical time-series [2], [28], [36]. However, in contrast to previously proposed modifications for handling conditional shift, we impose fewer restrictions on how parameters can be modified at each time step.

Mixture of experts models are commonly used for multitask learning and conditional computation [71]–[76]. By framing conditional shift as a multitask problem, we can exploit the large body of work in mixture of experts. [71] proposed a two-step approach in which first, a hidden Markov model (HMM) learns a segmentation of the time series, so that each segment is assigned to an expert, and second, the learned experts are mixed at the segmentation boundaries. [74] stacked experts to form a deep mixture of experts; [75] mixed the parameters of fully connected layers, stacking them to account for differences in the training and test sets in audio processing tasks; and [72] learned a gating function to mix the output of experts in a multitask learning setting. The methods proposed by [76] are particularly related. The authors learn coefficients for mixing convolution parameters, increasing parameter sharing across layers of a convolutional neural network (CNN). Building on these approaches, we investigate the utility of a mixture of LSTMs. At each time step, we apply the learned mixing coefficients to form a combined LSTM cell. This facilitates end-to-end learning and allows more than two experts to flexibly contribute to any time step's prediction. Our setting differs from [76] as the mixing coefficients are a) constrained to belong to a simplex, b) learned for each time step instead of each layer, and c) applied to LSTM cells instead of CNN filters.

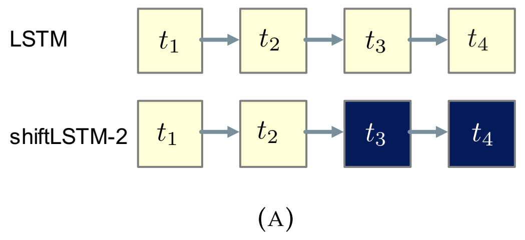
4.3 Methods

In this section, we describe extensions to LSTMs that facilitate learning in the presence of temporal conditional shift. Building off of an LSTM architecture, we present two variations that relax parameter sharing across time: `shiftLSTM` and `mixLSTM`. The first approach, `shiftLSTM`, represents a simple baseline in which different parameters are learned for different time steps (i.e., separate LSTM cells for different time periods). The second approach, `mixLSTM`, addresses the shortcomings of this simple baseline through a mixture approach. But first, we formalize the problem setting of temporal conditional shift and review the architecture of an LSTM.

4.3.1 Problem Setup - Temporal Conditional Shift & LSTMs

Given time-series data representing patient covariates over time, $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T]$ where $\boldsymbol{x}_t \in \mathbb{R}^d$, we consider the task of predicting a sequence of outcomes $\boldsymbol{y} = [y_1, y_2, ..., y_T]$, where $y_t \in \mathbb{R}$ for $t \in [1, 2, 3, ..., T]$. We consider a scenario in which the relationship between $\boldsymbol{x}_{1:t}$ and y_t varies over time, i.e., $y_t = f(\boldsymbol{x}_{1:t}, t; \boldsymbol{\theta}_t)$, where $\boldsymbol{\theta}_t$ represent model parameters at time t. Because t is measured with respect to a patient-specific fiducial marker, we restrict ourselves to conditional shift within a patient-specific time scale (e.g., within an admission).

In the sequence-to-sequence setting described above, LSTMs take as input time-varying patient covariates and output a prediction at each time step. Dynamics are captured in part through a cell state $\boldsymbol{C}_t$ that is maintained over time. A standard LSTM cell is described below, where $*$ represents element-wise multiplication. Here, $\boldsymbol{h}_t$ and $\tilde{\boldsymbol{C}}_t$ represent



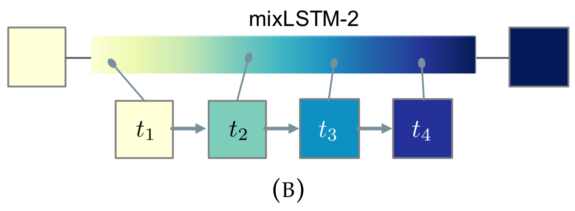


FIGURE 4.1: An illustrative plot comparing `LSTM`, `shfitLSTM`, and `mixLSTM`, with 4 time steps. Each square denotes an LSTM cell. Cells with the same color share the same weights. Arrows denote transitions between time steps. (a) `shiftLSTM-2` is similar to an LSTM, except it uses different cells for the first two time steps compared to the last two. (b) `mixLSTM-2` has two independent underlying cells, and at each time step, it generates a new cell by mixing a convex combination of the underlying cells. For illustrative purposes, the weights at each time step are drawn from sequential locations on the continuum, but in reality, the weight combination is independent of relative position of the time step.

the hidden state and the update to the cell state, respectively.

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_{\boldsymbol{i}}[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_{\boldsymbol{i}}) \tag{4.1}$$
$$\tilde{\boldsymbol{C}}_t = \tanh(\boldsymbol{W}_{\tilde{\boldsymbol{c}}}[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_{\tilde{\boldsymbol{c}}}) \tag{4.2}$$
$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_{\boldsymbol{f}}[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_{\boldsymbol{f}}) \tag{4.3}$$
$$\boldsymbol{C}_t = \boldsymbol{i}_t * \tilde{\boldsymbol{C}}_t + \boldsymbol{f}_t * \boldsymbol{C}_{t-1} \tag{4.4}$$
$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_{\boldsymbol{o}}[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_{\boldsymbol{o}}) \tag{4.5}$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t * \tanh(\boldsymbol{C}_t) \tag{4.6}$$
$$\hat{y}_t = \boldsymbol{W}_y \boldsymbol{h}_t + b_y \tag{4.7}$$

Importantly, each of the learned parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ in equations (1)-(3), (5) and (7) do not vary with time. To capture time-varying dynamics, the hidden and cell states ($\boldsymbol{h}_t$, $\boldsymbol{C}_t$) must indirectly model conditional shift.

4.3.2 Relaxed Parameter Sharing in LSTMs

We hypothesize that in settings where the amount of training data is limited – this is often the case in health applications – an approach that more directly models conditional shift through time-varying parameters will outperform a standard LSTM. To this end, we explore two variations on the LSTM: the `shiftLSTM` and the `mixLSTM`, illustrated in **Figure** 4.1.

shiftLSTM - learning abrupt transitions

As a baseline, we consider an approach that naïvely minimizes parameter sharing across cells, by learning different parameters $\boldsymbol{W}^{(t)}$ and $\boldsymbol{b}^{(t)}$ at each time step t, instead of the time-invariant parameters in equations (1)-(3), (5), and (7). This mimics a feed-forward network, with the hidden state and cell state propagating forward at each time step, but computes the output sequentially. This naïve approach to relaxed parameter sharing assumes no shared relationships across time. As a result, its capacity is significantly greater than that of an LSTM. Given the same hidden state size, the number of parameters scales linearly with the number of time steps. We hypothesize that this naïve approach will result in overfitting and poor generalization, in settings with limited data. To strike a balance between the two extremes, complete sharing and no sharing, we explore a variation of this approach that assumes parameters are shared across a subset of adjacent time steps: `shiftLSTM-`K.

`shiftLSTM-`K. This approach sequentially combines K different LSTM cells over time, resulting in different model parameters every $\lceil T/K \rceil$ time steps (**Figure** 4.1a). $K \in \{1, ..., T\}$ is a hyperparameter, with `shiftLSTM-1` being no different than an LSTM with complete parameter sharing, and `shiftLSTM-`T corresponding to different parameters at each time step. All parameters are learned jointly using backpropagation.

mixLSTM - learning smooth transitions

As described above, the `shiftLSTM` approach is restricted to sharing parameters within a certain number of contiguous time steps. This not only leads to a substantial increase in the number of parameters, but also results in possibly abrupt transitions. We hypothesize that changes in health data, and risk factors specifically, are gradual. To allow for smooth transitions in time, we propose a mixture-based approach: `mixLSTM`-K (**Figure** 4.1b).

`mixLSTM`-K. Given K independent LSTM cells with the same architecture, let $\boldsymbol{W}^{(k)}$ and $\boldsymbol{b}^{(k)}$ represent the k^{th} model's weight parameters from equations (1)-(3), (5) and (7). The parameters of the resulting `mixLSTM` at time step t are

$$\boldsymbol{W}_t = \sum_{k=1}^{K} \lambda_t^{(k)} \boldsymbol{W}^{(k)}, \qquad \boldsymbol{b}_t = \sum_{k=1}^{K} \lambda_t^{(k)} \boldsymbol{b}^{(k)} \tag{4.8}$$

where $\boldsymbol{\lambda} = \{\lambda_t^{(k)} : t = 1 \dots T,\ k = 1 \dots K\}$ are the mixing coefficients and each $\lambda_t^{(k)}$ represents the relevance of the k^{th} model for time step t. The mixing coefficients are learnable parameters (initialized randomly) and are constrained such that $\sum_k \lambda_t^{(k)} = 1$ and $\lambda_t^{(k)} \geq 0$. Similar to above, K is also a hyperparameter, but here it can take on any positive integer value. Note that for every K, all possible `shiftLSTM`-K models can be learned by `mixLSTM`-K.

By mixing models, instead of abruptly transitioning from one model to another, `mixLSTM` can learn to share parameters over time. Moreover, though we do not constrain the mixing coefficients to change smoothly, their continuous nature allows for smooth transitions. We verify these properties in our experiments.

4.4 Experimental Setup

We explore the effects of temporal conditional shift in both synthetic and real data. Here, we describe i) these datasets, ii) several baselines to which we compare our proposed approach, and iii) the details of our experimental setup.

4.4.1 Synthetic Data

We begin by considering a scenario in which we can control the extent of conditional shift in the problem. This allows us to test model performance in a setting where the amount of temporal conditional shift is known. Specifically, we consider a multitask variation of the 'copy memory task' [77], with input sequence $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_T\}$, $\boldsymbol{x}_t \in \mathbb{R}^d$, and output sequence $\{y_{l+1}, \ldots, y_T\}$, $y_t \in \mathbb{R}$ (we start generating output once we have accumulated l values). The output at each time step is some predetermined, weighted combination of inputs from the previous l time steps, described by two probability vectors, $w_t^{(l)} \in \mathbb{R}^l$ and $w_t^{(d)} \in \mathbb{R}^d$, which are used for weighting the l time steps and d feature dimensions respectively. The parameters change gradually at every time step t, such that each time step's weighting (or task) is similar to the task from the previous time step. The parameter δ controls amount of change between temporally adjacent tasks. The generation process of these parameters is described below, followed by the generation process of the datasets. Here, $\left[\boldsymbol{x}_{t-l}, \ldots, \boldsymbol{x}_{t-1}\right]^\intercal \in \mathbb{R}^{l\times d}$ is the concatenation of the previous l inputs at time step t. Inputs are generated to be sparse. `Renormalize`(v) refers to a renormalization process that ensures the weights in every w_t vector are positive and sum to 1. This is done every time step to ensure that the effect of δ does not diminish as t increases.

```
Procedure SampleWeights($T$, $l$, $m$):
    $w_{l+1} \sim \text{Uniform}(0,1) \in \mathbb{R}^m$
    $w_{l+1} = \texttt{Renormalize}(w_{l+1})$
    for $t \in \{l+2, \ldots, T\}$ do
        $\Delta_t \sim \text{Uniform}(-\delta, \delta) \in \mathbb{R}^m$
        $w_t = \texttt{Renormalize}(w_{t-1} + \Delta_t)$
    end
    return $w_{l+1}, \ldots, w_T$

$w_{l+1}^{(d)}, \ldots, w_T^{(d)} = \texttt{SampleWeights}(T, l, d)$
$w_{l+1}^{(l)}, \ldots, w_T^{(l)} = \texttt{SampleWeights}(T, l, l)$
```

```
Procedure SampleData($T$, $w_{l+1:T}^{(l)}$, $w_{l+1:T}^{(d)}$):
    for $t \in \{1, \ldots, T\}$, $i \in \{1, \ldots, d\}$ do
        $z_i \sim \text{Bernoulli}(0.1)$ # for sparse inputs
        $x_i \sim \text{Uniform}(0, 100)$
        $x_t[i] = z_i x_i$
    end
    for $t \in \{l+1, \ldots, T\}$ do
        $y_t = w_t^{(l)\intercal} \left[\boldsymbol{x}_{t-l}, \ldots, \boldsymbol{x}_{t-1}\right]^\intercal w_t^{(d)}$
    end
    return $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$, $\{y_{l+1}, \ldots, y_T\}$
```

Our goal is then to learn to predict $\{y_{l+1}, \ldots, y_T\}$ based on input from the current and all preceding time steps. For each $\delta \in \{0.0, 0.1, 0.2, 0.3, 0.4\}$, we generated five sets of temporal weights, and then used each set to create five different synthetic dataset tasks where $T = 30$, $d = 3$, and $l = 10$. These twenty-five tasks were kept the same throughout experiments involving synthetic data. Train, validation and test sets all had size $N = 1,000$ unless otherwise specified.

4.4.2 Clinical Prediction Tasks

In addition to exploring conditional shift in synthetic data, we sought to test our hypotheses using real clinical data from MIMIC-III [22]. Below we describe the three clinical prediction tasks of interest, in addition to their corresponding study populations, patient covariates, and evaluation criteria.

Outcomes

Throughout the first 48 hours of each ICU visit, we sought to make predictions regarding a patient's risk of experiencing three different outcomes: acute respiratory failure (ARF), shock, and in-hospital mortality, each described in turn below.

ARF. Acute respiratory failure is defined as the need for respiratory support with positive pressure mechanical ventilation [3], [32]. Onset time of ARF was determined by either the documented receipt of invasive mechanical ventilation (`ITEMID: 225792`) or non-invasive mechanical ventilation (`ITEMID: 225794`) as recorded in the `PROCEDURESEVENTS_MV` table, or documentation of positive end-expiratory pressure (PEEP) (`ITEMID: 220339`) in the `CHARTEVENTS` table, whichever occurs earlier. Ventilator records and PEEP settings that are explicitly marked as `ERROR` did not count as an event.

Shock. Shock is defined as inadequate perfusion of blood oxygen to organs or tissues [4], and is characterized by receipt vasopressor therapy. Onset time of shock was determined by the earliest administration of vasopressors [78]. Using the `INPUTEVENTS_MV` table, we considered the following vasopressors:

- norepinephrine (`ITEMID: 221906`),
- epinephrine (`ITEMID: 221289`),
- dopamine (`ITEMID: 221662`),
- vasopressin (`ITEMID: 222315`), and

- phenylephrine (`ITEMID: 221749`).

Drug administration records with the status of `REWRITTEN`, incorrect units, or non-positive amounts/durations did not count towards an event.

In-hospital mortality. As in [2], the time of in-hospital mortality was determined by comparing patient date of death (`DOD` column) from the `PATIENTS` table with hospital admission and discharge times from the `ADMISSIONS` table.

Cohort Selection

We considered adult admissions with a single, unique ICU visit. This excludes patients with transfers between different ICUs. Patients without labels or observations in the ICU were excluded. Since we are interested in how relationships between covariates and outcome change over time, we focused our analysis on patients who remained in the ICU for at least 48 hours. In addition, for ARF and shock prediction tasks, patients who experienced the event of interest before 48 hours were excluded. Using the full 48 hours allows us to focus on temporal trends that are more likely to be present in longer visits. **Table** 4.1 shows the number of admissions and positive labels for the three tasks after applying exclusion criteria.

TABLE 4.1: We considered three clinical prediction tasks. The study population varied in size across tasks, as did the fraction of positive cases (i.e., the portion of patients who experienced the outcome of interest.)

Task	Number of ICU admissions (%positive)
ARF	3,789 (6.01%)
shock	5,481 (5.98%)
in-hospital mortality	21,139 (13.23%)

Data Extraction and Feature Choices

We used the same feature extraction procedure as detailed in [2]. For completeness, we briefly describe the feature extraction process here. For each ICU admission, we extracted

`https://github.com/YerevaNN/mimic3-benchmarks`

17 physiological features (e.g., heart rate, respiratory rate, Glasgow coma scale, see **Table** A.1 in Appendix A.1) from the first 48 hours of their ICU visit. We applied mean normalization for continuous values and mapped categorical values to binary features using one-hot encoding, resulting in 59 features. We resampled the time series with a uniform sampling rate of once per hour with carry-forward imputation. Mask features, indicating if a value had been imputed resulted in 17 additional features. After preprocessing, each example was represented by $d = 76$ time-series (see **Table** A.2 in Appendix A.1 for the complete list of features) of length $T = 48$ and three binary labels indicating whether or not the patient developed ARF, developed shock or died during the remainder of the hospital stay.

Evaluation

Given these data, the goal was to learn a mapping from the features to a sequence of probabilities for each outcome: ARF, shock or in-hospital mortality. We split the data into training, validation, and test as in [2]. We used target replication when training the model [28]. For example, if a patient eventually developed ARF, then every hour of the first 48 hours is labeled as positive (negative otherwise). We used the validation set for hyperparameter tuning, and report model performance as evaluated on the held-out test set. Since we consider a sequence-to-sequence setting, each model makes a prediction for every hour during the first 48 hours. These predictions were evaluated based on whether or not at least one prediction exceeds a given threshold. This threshold was swept across all ranges to generate a receiver operating characteristics curve (ROC) and precision-recall curve (PR). This resembles how the model is likely to be used in practice. With the goal of making early predictions, as soon as the real-time risk score exceeds some specified threshold, clinicians could be alerted to a patient's increased risk of the outcome. It should be noted that this differs from the evaluation used in [2] where a single prediction was made during the 48-hour period. We report performance in terms of the area under the ROC and PR curves (AUROC, AUPR), computing 95% confidence intervals using 1,000 bootstrapped samples of the test set.

4.4.3 Baselines for Comparison

In addition to the approaches with relaxed parameter sharing described in the Section 4.3, we considered a number of baselines, which are described below.

`NN`. A non-recurrent, feed-forward neural network with one hidden layer. The same network is applied *independently* at every time step to generate the prediction for that time step. This model has complete parameter sharing but no recurrent structure to capture temporal dynamics, and thus is a simpler model than LSTMs and less likely to overfit. This model serves as a simple baseline, but highlights the complexity of the tasks in terms of temporal dynamics.

`NN+t`. Similar to `NN` but with an additional input feature $\hat{x}_t = \frac{t}{T} \in \mathbb{R}$ at every time step, representing the relative temporal position. Given time as an input, this model has the capacity to model temporal conditional shift but cannot leverage longitudinal trends.

`LSTM`. We considered a standard LSTM in which parameters are completely shared across time. Synthetic tests used the default Pytorch v0.4.1 implementation (`torch.nn.LSTM()`). In our experiments on the clinical data, we implemented an LSTM that employed orthogonal parameter initialization and layer normalization, in order to match the settings used in the original HyperLSTM implementation (see below).

`LSTM+t`. `shiftLSTM` and `mixLSTM` intrinsically have an additional signal regarding the current time step (captured through the use of time-specific parameters). In order to test whether this was driving differences in performance, we tested `LSTM+t`, an LSTM with an additional input feature $\hat{x}_t = \frac{t}{T} \in \mathbb{R}$ at every time step, representing the relative temporal position.

`LSTM+TE`. Given that positional encoding has recently been shown to provide an advantage over simply providing position [79], we also explored adding a temporal encoding as additional input features. We used a 24-dimensional encoding for each time step. We tested encoding sizes of 12, 24, 36 and 48 on the in-hospital mortality task, and found 24 to result in the best validation performance. We calculated the temporal encoding as:

$TE_{(t,i)} = \sin\left(\frac{t}{10000^{i/24}}\right)$ if i is even, and $TE_{(t,i)} = \cos\left(\frac{t}{10000^{(i-1)/24}}\right)$ if i is odd, where t represents the time step and i the position in the encoding indexed from 0.

`HyperLSTM`. First proposed by [24], this approach uses a smaller, auxiliary LSTM to modify the parameters of a larger, primary LSTM at each time step. Since the parameters at each time step are effectively different, this is a form of relaxed parameter sharing. As in the original implementation, we used orthogonal parameter initialization and layer normalization. The two networks were trained jointly using backpropagation.

4.4.4 Model Training & Implementation Details

The two non-LSTM baselines both used one hidden layer with ReLU activation and a softmax nonlinearity at the output layer. Except in the case of the LSTM applied to synthetic data, LSTM models consisted of single-layer recurrent cells that were orthogonally initialized followed by a fully-connected layer and a softmax nonlinearity. For experiments involving synthetic data, to compensate for the lower capacity of the LSTM compared to `mixLSTM` and `shiftLSTM` which have multiple cells, we allowed it to use an additional layer. Capacity was less of an issue in the experiments involving real data. We tuned the size of the hidden state(s) in all methods based on validation performance.

We trained all models using the Adam optimizer [80] (Pytorch implementation) with the default learning rate of 0.001. On synthetic data we aimed to minimize the mean squared error (MSE) loss, and for the clinical prediction tasks, we aimed to minimize the cross entropy loss with target replication. We used early stopping based on validation performance – MSE loss on synthetic data tasks, AUROC on real data tasks – with a patience of 5 epochs. Models for synthetic datasets were trained with 40 random initializations/hyperparameter settings for a maximum of 30 epochs. We used a batch size of 100 and performed a random search over hidden state sizes of {100, 150, 300, 500, 700, 900, 1100}. For learning models on clinical tasks, we used a batch size of 8, because it was the optimal LSTM batch size setting used in the MIMIC-III benchmark paper on the in-hospital mortality task [2]. When learning models for ARF and shock, we considered 20 random initializations, and trained for a maximum of 30 epochs. For LSTM models, we performed a random search over hidden state and auxiliary hidden states sizes of {25, 50, 75, 100, 125, 150}; for `NN` and `NN+t`, we performed a random search over the number of hidden units in {25, 50, . . . , 1000}. When learning models for in-hospital mortality, we considered 10 random initializations and trained for a maximum of 10 epochs, in part because of the larger training set size. For LSTM models on this task, we performed a random search over hidden state size {100, 150, 300, 500, 700} and the same auxiliary hidden state size search as for ARF and shock; for `NN` and `NN+t`, we considered the same range of hyperparameters as for ARF and shock. To facilitate comparisons, the code for all of our experiments is publicly available online.

4.5 Experiments & Results

We explore the trade-off between temporal conditional shift and shared weights and examine the utility of relaxed weight sharing in this setting. In our experiments, we seek to answer the following questions:

- Question 1: What is the trade off between temporal conditional shift and the performance of shared weight models (i.e., LSTM)?
- Question 2: Does temporal conditional shift exist in clinical tasks?
- Question 3: Does relaxed weight sharing (i.e., `mixLSTM`) improve model performance on tasks with temporal conditional shift?
- Question 4: What time varying relationships are being captured by `mixLSTM`?
- Question 5: How robust is the performance of `mixLSTM` when training data is limited?

4.5.1 Exploring the Effects of Temporal Conditional Shift

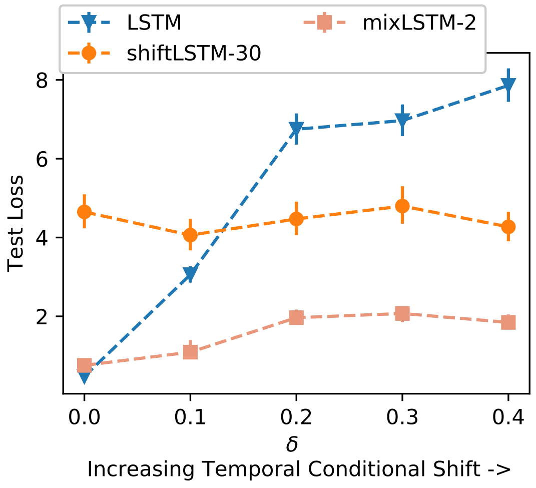
Does parameter sharing hinder the ability of an LSTM to capture time-varying relationships? The data generation process described in Section 4.4.1 allows us to control the amount of temporal conditional shift present in the task. Specifically, by increasing δ, we increase the variability between two temporally adjacent tasks. This allows us to test the

`https://gitlab.eecs.umich.edu/MLD3/MLHC2019_Relaxed_Parameter_Sharing` effects of conditional shift on the performance of an LSTM. We hypothesize that because the LSTM shares parameters over time, it will struggle to adapt to temporal conditional shift. To test our hypothesis, we compare the performance of an LSTM with `shiftLSTM` across a range of δ values (**Figure** 4.2a). Here, the `shiftLSTM` approach learns different sets of parameters for each time step (30 in total). We observe a clear trend: as temporal conditional shift increases, the performance of the LSTM decreases. In contrast, `shiftLSTM` results in steady performance across the range of δ. At low $\delta \in \{0, 0.1\}$, the LSTM outperforms the `shiftLSTM` in terms of MSE on the test set. In this experiment, we limited the amount of training data to 1,000 samples. Theoretically, given enough training data, LSTM should be capable of accurately modeling time-varying relationships. To verify this, we show that the test loss associated with the LSTM models approaches zero as the training set size increases (**Figure** 4.2b). These results support our initial hypothesis that in settings with limited data, temporal conditional shift negatively impacts LSTM performance and that this impact is in part due to the sharing of parameters.

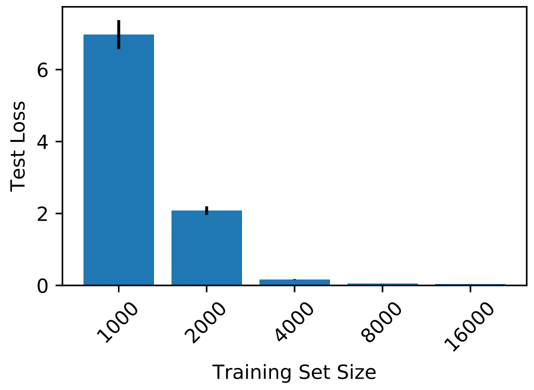
Is there any evidence of time-varying relationships in the three clinical prediction tasks of interest? We tested for the presence of temporal conditional shift in three clinical prediction tasks: ARF, shock, and in-hospital mortality. For these tasks, the underlying parameters that govern the amount of temporal conditional shift (e.g., δ) are unknown. Instead, we indirectly measure temporal conditional shift by applying `shiftLSTM`-K varying K from $\{1, 2, 3, 4, 8, 48\}$, where $K = 1$ is a standard LSTM, and $K = 48$ implies a different set of parameters for each time step. Increasing the number of cells or K reduces sharing. As the difference between sequential tasks increases, we expect the benefit of learning different LSTM cells (less parameter sharing) to increase. Empirically, we observe that less parameter sharing results in better performance (**Figure** 4.3). This supports our hypothesis that architectures for solving clinical prediction tasks could benefit from relaxed parameter sharing.

4.5.2 Comparing the Proposed Approach to Baselines

In this section, we explore the performance of the proposed approach, `mixLSTM`, relative to the other baselines. Again, we hypothesize that it will outperform the other approaches due to a) smooth sharing of parameters and b) the ability to learn which cells to share.

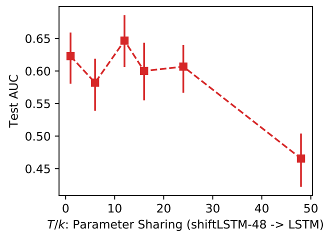


(A)

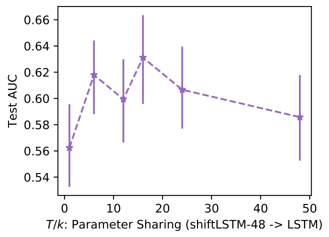


(B)

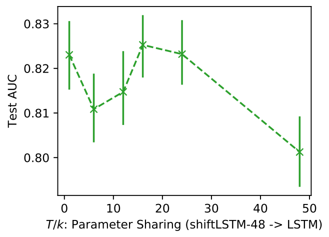
FIGURE 4.2: (A.1): LSTM performance decreases as conditional shift increases. With increasing conditional shift, the time-varying architecture outperforms the LSTM, suggesting that parameter sharing hurts LSTM performance. (A.2) `mixLSTM` bridges the performance tradeoff between LSTM and `shiftLSTM`. As conditional shift increases, `mixLSTM`'s ability to relax parameter sharing helps it increasingly outperform LSTM. By assuming that tasks are unique but related it outperforms `shiftLSTM`. (B) This issue is only apparent when training data are limited; LSTMs can adapt to temporal conditional shift given enough training data. Error bars represent 95% confidence intervals based on bootstrapped samples of the test set. δ was set to 0.3.



(A) ARF



(B) shock



(C) mortality

FIGURE 4.3: As we increase parameter sharing by increasing T/K along the x-axis, there is a drop in performance, supporting our hypothesis that temporal conditional shift is present in our real data tasks. Error bars represent the interquartile ranges based on bootstrapped samples of the test set.

mixLSTM strikes a balance between complete parameter sharing (LSTM) and no parameter sharing (shiftLSTM-48). In addition, compared to shiftLSTM, mixLSTM can share parameters between distant time steps and learns how to accomplish this.

How does the proposed approach perform on synthetic data? mixLSTM has the ability to continuously interpolate between K independent cell parameters. In this instance, mixLSTM-2 has 15 times fewer parameters relative to shiftLSTM-30. On the synthetic data tasks, mixLSTM consistently outperforms shiftLSTM at all levels of temporal shift (**Figure** 4.2a). Moreover, mixLSTM outperforms LSTM at low δ, except when no temporal shift exist. This agrees with our intuition that *smart* sharing is better than no sharing (shiftLSTM) and indiscriminate sharing (LSTM).

TABLE 4.2: Performance on ARF, shock, & mortality with 95% confidence intervals. Though the differences are small, mixLSTM consistently outperforms the other approaches across all tasks in terms of both AUROC and AUPR. The number of test samples for each task is reported in parentheses.

Model	**ARF** (n=549)		**shock** (n=786)		**mortality** (n=3,236)	
	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
NN	0.57 [0.45, 0.67]	0.05 [0.03, 0.10]	0.60 [0.52, 0.68]	0.08 [0.05, 0.12]	0.80 [0.77, 0.82]	0.39 [0.34, 0.44]
NN+t	0.58 [0.46, 0.68]	0.06 [0.03, 0.11]	0.56 [0.47, 0.64]	0.10 [0.05, 0.19]	0.79 [0.77, 0.82]	0.38 [0.33, 0.43]
LSTM	0.47 [0.35, 0.58]	0.04 [0.02, 0.07]	0.59 [0.49, 0.69]	0.09 [0.05, 0.16]	0.80 [0.78, 0.83]	0.39 [0.33, 0.43]
LSTM+t	0.42 [0.30, 0.54]	0.04 [0.02, 0.07]	0.62 [0.53, 0.70]	0.08 [0.05, 0.15]	0.81 [0.79, 0.83]	0.41 [0.36, 0.47]
LSTM+TE	0.48 [0.35, 0.61]	0.05 [0.03, 0.10]	0.60 [0.50, 0.69]	0.10 [0.06, 0.20]	0.82 [0.80, 0.85]	0.43 [0.38, 0.48]
HyperLSTM	0.57 [0.44, 0.68]	0.06 [0.03, 0.10]	0.63 [0.54, 0.72]	0.08 [0.05, 0.12]	0.82 [0.80, 0.84]	0.42 [0.37, 0.47]
shiftLSTM	0.61 [0.49, 0.70]	0.10 [0.03, 0.21]	0.61 [0.52, 0.70]	0.09 [0.05, 0.16]	0.81 [0.79, 0.84]	0.43 [0.37, 0.48]
mixLSTM	**0.72** [0.62, 0.80]	**0.15** [0.06, 0.27]	**0.67** [0.58, 0.76]	**0.10** [0.06, 0.16]	**0.83** [0.81, 0.85]	**0.45** [0.40, 0.50]

How does the proposed approach perform on the clinical prediction tasks? Applied to the three clinical prediction tasks (with varying amounts of training data), mixLSTM consistently performs the best (**Table** 4.2). The NN and NN+t models are simpler architectures that outperform other LSTM-based baselines only under very low data settings (ARF). Compared to the LSTM baseline, LSTM+t and LSTM+TE performed better given sufficient training data, suggesting that having direct access to time either as a feature or a temporal encoding is beneficial. Relaxing parameter sharing further improves performance. As shown earlier, shiftLSTM consistently improves performance over the standard LSTM.

HyperLSTM, similar to `mixLSTM`, bridges the dichotomy of completely shared and completely independent parameters, and outperforms both LSTM and `shiftLSTM` in some cases but not consistently. `mixLSTM` outperforms all other baselines on all three tasks, though the differences are not statistically significant in all cases. Both HyperLSTM and `mixLSTM` achieve high performance and both models relax parameter sharing. This supports our hypothesis that relaxed parameter sharing is beneficial in some settings.

In these experiments, we selected K for each task based on validation performance, testing $K \in \{2, 3, 4, 8, 48\}$ for `shiftLSTM` and sweeping K from 2 to 4 for `mixLSTM`. For `shiftLSTM`, K represents the optimal number of sequential tasks to segment the input sequence into; the best $K = 2, 2, 8$ for ARF, shock, and mortality respectively. For `mixLSTM`, K indicates the optimal number of operational or characteristic modes in the data; the best $K = 4, 4, 2$, respectively. It appears that for `shiftLSTM`, the chosen K is correlated with the amount of training data available. Both ARF and shock have significantly smaller training set sizes compared to mortality. In contrast, `mixLSTM` learns more cells for ARF and shock. This suggests that the structure of `mixLSTM` is better suited to the problem setting than `shiftLSTM`, since it is able to train twice as many cells as `shiftLSTM` and attain a higher test performance. The converse also supports this claim. `mixLSTM` is able to train $\frac{1}{4}$ the number of cells as `shiftLSTM` for mortality and still attain better performance. The optimal K for `mixLSTM` appears to be less indicative of training set size, and more a reflection of the true number of operational or characteristic modes in the data. When we visualize the mixing ratios learned by `mixLSTM-2` in later sections (**Figure** 4.5) we see that while mortality smoothly interpolates between cell1 and cell2 as time passes, ARF and shock both display an initial peak followed by a gradual interpolation. This suggests that the dynamics are more complex for ARF and shock.

4.5.3 Robustness and Sensitivity Analyses

In this section, we further analyze `mixLSTM`, focusing on its robustness in settings when training data are limited and investigate what it has learned in terms of mixing trends and changing feature importance.

Performance with Limited Training Data

Does the proposed approach still perform well when training data are limited? We hypothesized that `mixLSTM` will continue to outperform LSTM, even when training data are limited because `mixLSTMs` are better suited to problem settings exhibiting temporal conditional shift. To test our hypothesis, we compared the performance of `mixLSTM-2` and LSTM trained using different training set sizes for the task of predicting in-hospital mortality. We chose to focus on the task of in-hospital mortality, since it had the most training data (training set size $= 14,681$). We subsampled the training set repeatedly for $N \in \{250, 500, 2000, 5000, 8000, 11000\}$. The test set was held constant across all experiments and $K = 2$ to limit the capacity of the model. `mixLSTM` consistently outperforms LSTM across all ranges of training set sizes (**Figure 4.4**). As one might expect, differences are subtle at smaller training set sizes, where an LSTM with complete parameter sharing is likely more sample efficient and therefore more competitive.

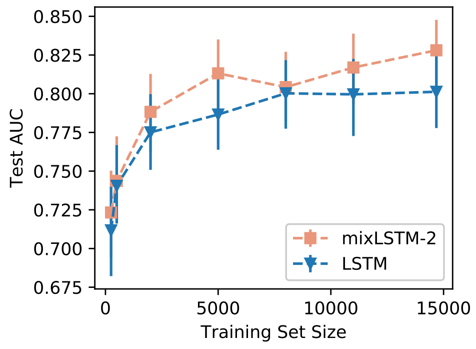


FIGURE 4.4: `mixLSTM-2` is consistently better than LSTM at different training set sizes. Error bars represent 95% confidence intervals bootstrapped from the test set.

What has mixLSTM learned?

To dive deeper into what exactly the `mixLSTM` has learned, we visualize the learned mixing coefficients and the most important features.

Are `mixLSTM`'s learned mixing coefficients smooth? In our learning objective function, `mixLSTM`'s mixing coefficients are not constrained to be smooth. However, we hypothesize that this behavior reflects the underlying dynamics in clinical data. **Figure** 4.5 plots the mixing coefficients ($\lambda^{(1)}$) over time for `mixLSTM-2` on the three clinical prediction tasks. Since there are only two independent cells ($K = 2$), we can infer $\lambda^{(2)} = 1 - \lambda^{(1)}$. The trend indicates that one cell captures the dynamics associated with the beginning of a patient's stay, while the second cell captures the dynamics 48 hours into the stay.

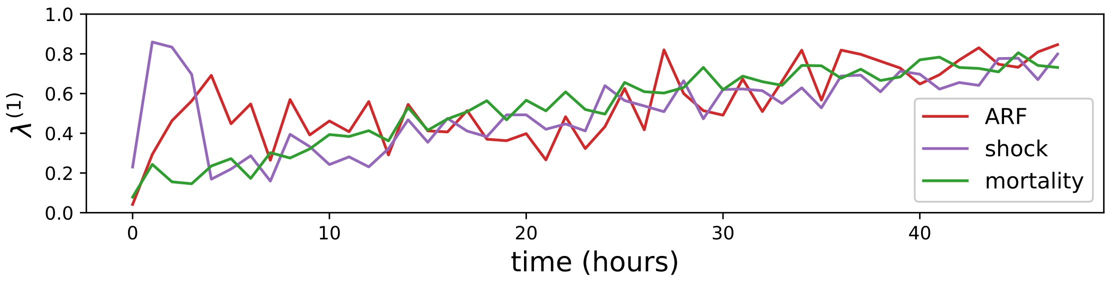


FIGURE 4.5: Visualization of the mixing coefficients learned by `mixLSTM-2`. $\lambda^{(1)}$ is shown on the y-axis, while $\lambda^{(2)}$ can be inferred ($\lambda^{(2)} = 1 - \lambda^{(1)}$). Although not constrained to be smooth, we observe a smooth transition of mixing coefficients between time steps, indicating that one cell is specialized for the beginning of a patient's ICU stay while the other is specialized for 48 hours into the ICU stay.

Does explicitly smoothing the mixing coefficients help? Based on patterns displayed in **Figure** 4.5, we hypothesized that additional smoothing of the mixing coefficients could aid classification performance. To test this hypothesis, we applied regularization based on a similarity measure between models at consecutive time steps. Following [76], we used the normalized cosine similarity as the similarity measure. For consecutive time steps t and $t+1$, this similarity score is $s_t = \frac{\langle \boldsymbol{\lambda}_t, \boldsymbol{\lambda}_{t+1} \rangle}{\|\boldsymbol{\lambda}_t\|_2 \|\boldsymbol{\lambda}_{t+1}\|_2}$ where $\boldsymbol{\lambda}_t := [\lambda_t^{(1)}, \cdots, \lambda_t^{(K)}]$. Denoting $\mathcal{L}$ as the original loss function and $\alpha \in \mathbb{R}^+$ as the regularization strength, we minimize the regularized objective $\mathcal{L}_R := \mathcal{L} - \alpha \sum_{t=1}^{T-1} s_t$ to encourage temporal smoothness.

Figure 4.6 illustrates the effect of temporal smoothness regularization on the model for the mortality task. As expected, larger regularization strength encourages models to share parameters. However, test performance drops monotonically as α increases. Additional regularization likely results in lower model complexity and in some settings underfitting.

Our result aligns with [76] in that while smoothness in patterns naturally arises, explicitly encouraging smoothness through regularization hurts performance.

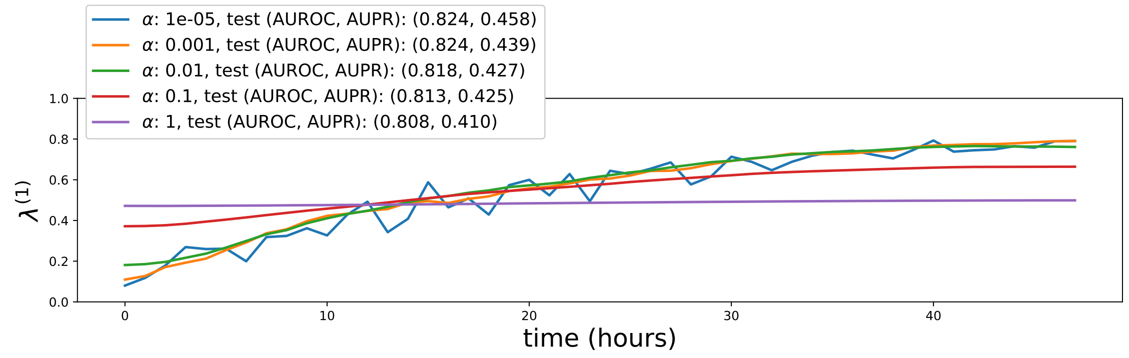


FIGURE 4.6: Visualization of the mixing coefficients learned by `mixLSTM-2` with different regularization strengths for the mortality task. $\lambda^{(1)}$ is shown on the y-axis. $\lambda^{(2)}$ can be inferred ($\lambda^{(2)} = 1 - \lambda^{(1)}$). As regularization strength increase, mixing coefficients become smoother at the cost of lower performance.

What time-varying relationships does the `mixLSTM` learn to recognize? When attempting to understand which features drive a model's predictions, the focus is often put on the importance of certain features. However, because `mixLSTM` was designed to and has been shown to excel in situations with temporal conditional shift, we focus on identifying the features whose influence changes over time. To identify such features, we must first measure the effect of each feature at each time step. Here, we use the input gradient as a proxy for feature importance and visualize importance over time [81]–[83] (**Figure** 4.7). More specifically, we traversed the test set, accumulating the input gradient with respect to the target class. One of the most noticeable patterns is the large amount of variation in feature importance in the first 6 hours of an ICU admission. This pattern is most apparent for the task of predicting shock (**Figure** 4.7b). This may reflect the significant physiological changes a patient may experience at the beginning of their ICU stay as interventions are administered in an effort to stabilize them.

We list the continuous features ranked by importance in **Table** 4.3. Feature importance was calculated by summing importance over time and taking the absolute value. Here positive importance values are associated with increased risk, while negative importance

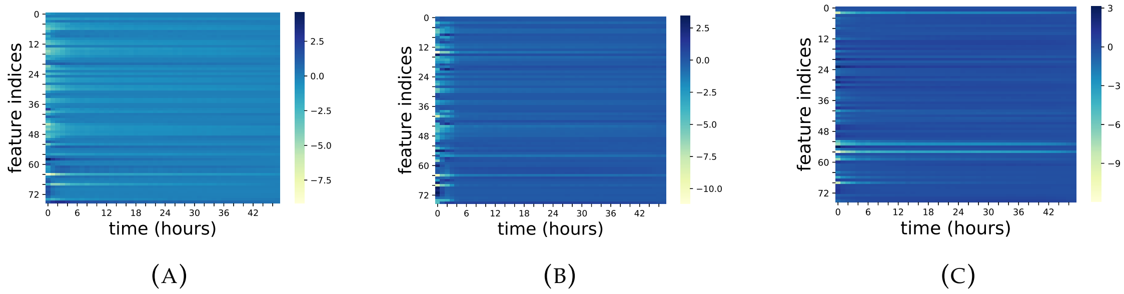


FIGURE 4.7: Input gradient based saliency map of `mixLSTM-2` on three tasks. Each plot shows a proxy of importance of each feature across time steps. Some noticeable temporal patterns include high variability during the first six hours, which may be a reflection of increased physiological change a patient may experience at the beginning of their ICU stay when interventions are more frequent.

values are associated with protection. The color scheme reflects the overall direction of association and the change over time. Dark red and dark green represent 'risk' and 'protective' factors that lead to increased and decreased risk over time, respectively; that is, their effects become amplified over time. Light red and light green represent 'risk' and 'protective' factors that lead to decreased and increased risk over time, respectively; that is, their effects diminish over time. For example, in all three tasks, 'fraction of inspired oxygen' (indicative of whether or not a patient is on supplemental oxygen) is a risk factor initially, and becomes more important over time. This suggests that if a patient is still on high levels of oxygen 48 hours into their ICU admission, their risk is elevated for all three outcomes. For ARF and shock a similar pattern holds for heart rate, where sustained high heart rate is associated with greater risk over time. This suggests that some features, when persistently abnormal, further amplify a patient's risk.

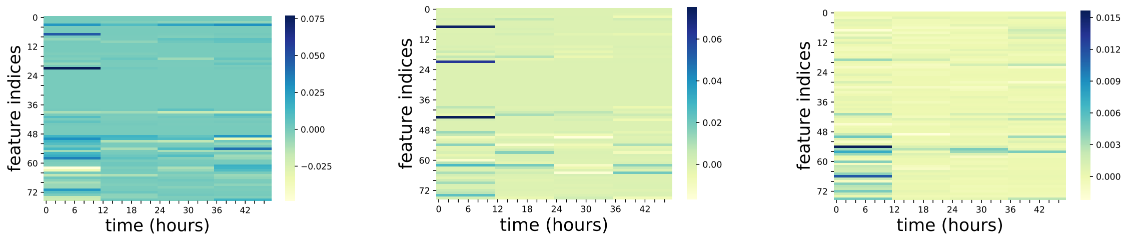
It is important to note that interpreting neural networks, and LSTMs in particular, remains an open challenge. Though the approach considered here is frequently used for interpreting LSTMs, it relies on the local effect of a feature and thus ignores the global trends [83]–[85]. Moreover, these methods merely identify associations and not causation.

Given the limitations of using input gradients to model the importance of discrete features, we also investigated feature importance using a permutation based sensitivity analysis [86], [87]. In the test set, we randomly permuted each covariate at each specific time period and measured predictive performance. By permuting each covariate in

TABLE 4.3: Physiological data ranked by overall importance as identified by `mixLSTM-2` on ARF, shock, and mortality using input gradient. The table is color coded. Light red denotes features that are initially risk factors, where risk *decreases* over time. Dark red denotes features that are initially risk factors, but where risk *increases* over time. Light green denotes features that are initially protective, but become less protective over time. Dark green denotes features that are initially protective, and becomes more protective over time.

ARF	**shock**	**mortality**
pH	Respiratory rate	Respiratory rate
Oxygen saturation	Height	Heart Rate
Weight	Mean blood pressure	Glucose
Respiratory rate	Heart Rate	Fraction inspired oxygen
Fraction inspired oxygen	Fraction inspired oxygen	Height
Heart Rate	pH	Weight
Height	Weight	Systolic blood pressure
Glucose	Glucose	pH
Systolic blood pressure	Oxygen saturation	Mean blood pressure
Mean blood pressure	Systolic blood pressure	Diastolic blood pressure
Temperature	Diastolic blood pressure	Oxygen saturation
Diastolic blood pressure	Temperature	Temperature

turn, we destroy any information that a particular covariate provides. If performance then drops significantly relative to a non-permuted baseline, we conclude the feature was important. To prevent correlated variables from leaking information, we simultaneously permuted variables with a correlation coefficient ≥ 0.95. We permuted grouped features within periods of 12 hours to encourage consistency of perturbation along time. **Figure** 4.8 plots this measure of feature importance over time. Overall, we observe similar trends to the input gradient analysis. In addition to there being greater variability in the first part of the visit, we also observed significant changes in the importance of certain features (measured by sum of importance across time). For example, for the task of predicting in-hospital mortality, respiratory rate is initially the most important feature, but then temperature becomes more important as the patient state evolves. For ARF, a variable pertaining to the Glasgow coma scale is initially most important, before yielding to respiratory rate.



(A) (B) (C)

FIGURE 4.8: Permutation based saliency map of `mixLSTM-2` on three tasks. Each plot shows AUROC degradation for permuting a feature. A larger decrease in AUROC means that the feature is more important with respect to the prediction task. Some noticeable temporal patterns include an increased variability during the first 12 hours which may be a reflection of increased physiological change a patient may experience at the beginning of their ICU stay when interventions are more frequent.

4.6 Summary and Conclusions

In this work, we present and explore the issue of temporal conditional shift in clinical time-series data. In addition, we propose a mixture of LSTM model (`mixLSTM`) and demonstrate that it effectively adapts to scenarios exhibiting temporal conditional shift, consistently outperforming baselines on synthetic and clinical data tasks. We also show that the `mixLSTM` model can adapt to settings with limited training data and learns meaningful, time-varying relationships from the data.

While `mixLSTM` achieves consistently better performance on all tasks considered, we note some important limitations. First, we only considered fixed-length datasets. It would be beneficial to compare LSTM and `mixLSTM`'s ability to generalize to variable length data. Second, our features are largely physiological (e.g., heart rate, temperature). We hypothesize that other types of features such as medications may exhibit stronger time-varying relationships. Third, while it is reasonable to set time zero as the time of ICU admission, patients are admitted to the ICU at different points during the natural history of their illness. Future work should consider the alignment of patient time steps (e.g., learning an optimal alignment prior to applying `mixLSTM`).

Despite these limitations, our results suggest that temporal conditional shift is an important aspect of clinical time-series prediction and future work could benefit from considering this problem setting. Our proposed `mixLSTM` presents a strong starting point from which future work can build off of.

Chapter 5

Exploring Class Imbalance Driven Low Homophily

5.1 Introduction

In modeling the spread of infectious disease, information about interactions among individuals is critical. However, the probability of a transmission event can depend on a complex combination of several factors including: incubation period of the disease, proximity of the interaction, duration of the interaction, among others [89]–[92]. In many cases, it can be difficult to precisely model such complex relationships, especially when aspects of the disease (e.g., incubation period) are unknown or vary. Thus in this paper, we explore a data-driven approach to the problem of predicting transmission events in which we leverage graph neural networks (GNNs). GNNs take as input networks that encode potential pathways for transmission (i.e., interactions) and information about who is known to be infected. From these data GNNs aim to learn patterns of transmission. Specifically, GNNs transform input data in the form of a graph into learned node representations that summarize meaningful aspects of that node's neighborhood [40], [93]. While they have proven useful across a number of domains [19], their application in the context of modeling transmission dynamics in infectious disease has been limited [94]–[96].

Common GNN architectural choices such as using weighted averaging for aggregation, rely on an assumption that the problem exhibits high homophily: connected nodes

The following is an adaptation of a workshop paper [88].

are more likely to share similar outcomes (or labels) than unconnected nodes [19], [20]. When tasks instead exhibit low homophily (e.g., the WebKB dataset, which includes university websites and the hyperlinks between them) GNNs fail to leverage the network [97]. In such settings, GNNs are often outperformed by models that ignore the network and only rely on node features [20]. In light of these limitations, researchers have recently proposed a number of variations to the GNN architecture that specifically address the issue of low homophily [20].

However, in modeling the spread of infectious disease, many infectious diseases tend to have relatively low incidence rates with respect to the general population (e.g., 0.026 per 1000 hospitalized people in the case of hospital associated methicillin resistant *Staphylococcus aureus* (MRSA) [98]). **This leads to an interesting dichotomy where there is high homophily on average, but low homophily with respect to the minority class.** Most nodes are uninfected and are connected to other uninfected nodes (*i.e.,* high homophily). However, even infected nodes are likely to be connected to uninfected nodes due to the sheer number of uninfected nodes and the low probability of transmission (resulting in low homophily with respect to minority infected class). While such tasks exhibit high homophily on average, we hypothesize that applying a generic GNN in such settings will result in the same pitfalls of applying a generic GNN to a low homophily task. Moreover, given the potentially complex relationship among nodes and transmission events, we require a GNN with a wide receptive field. The larger the receptive field, the more neighbors and more interactions that are considered as potential transmission pathways. However, this often translates into increased model depth or additional pooling or aggregation, which can further exacerbate the negative repercussions of violating the homophily assumption.

Our Approach. Given that our task differs from the general low homophily setting, we do not expect recently proposed architecture variations to apply. Thus, we propose an attention-based solution to address the type of low homophily arising from class imbalance. Self-attention can be used to learn the importance of any neighboring pair of nodes with respect to a prediction task. Graph attention networks use this learned importance to inform which neighboring nodes should most influence an ego node's representation [21]. We hypothesize that this will lead to improved performance in the presence of class imbalance by allowing the network to focus on the few but highly important infectious individuals. Beyond addressing the issue of low homophily due to class imbalance, to gain a deeper understanding of the scenarios in which GNNs can learn accurate data-driven definition of transmission events, we evaluate the learned representations across several different tasks based on synthetic networks. Our contributions are as follow:

- We present a graph-based prediction task focused on estimating exposure to and transmission of infectious disease that exhibits high homophily overall, but low homophily in the minority class.
- We demonstrate the failure of generic GNNs in such settings.
- We compare the ability of an attention-based mechanism to address the class imbalance driven low homophily problem to recently proposed solutions.
- We demonstrate that GNNs are able to learn useful representations for predicting transmission events in both real clinical data and across a variety of synthetic networks.

Organization. The remainder of the chapter is organized as follows. First, we give a background on modeling disease transmission and graph neural networks. Then, we present our problem setting and the attention mechanism along with other GNN variants that have been proposed to deal with low homophily. Next, we describe our synthetic and clinical datasets and evaluation details. Finally, we present results on clinical and synthetic data, comparing different methods of handling low homophily as well as comparing a data-driven graph based approach to learning transmission dynamics to an expert-defined notion of colonization pressure.

5.2 Background & Related Work

Here, we review relevant background and related work, highlighting differences with our work. First, we describe how exposure to infectious pathogens is commonly modeled in the literature and second, we describe GNNs and some of the known shortcomings.

5.2.1 Modeling Disease Transmission

Accurately modeling the transmission of infectious diseases in a hospital setting is a complex task. There are many ways an infectious disease can be transmitted in a hospital. Common vectors include patients, healthcare workers and the physical environment [89], [90]. There is evidence that pathogens can linger in the environment for extended periods of time [91], [92]. In addition, not all human spreaders are necessarily symptomatic [99].

To date, researchers commonly model transmission events based on exposure to the pathogen. Exposure is often estimated at a population or aggregate level [100]–[102]. However, by definition, this ignores the specific interactions an individual has with those around them. Another popular estimation method relies on the concept of colonization pressure [17], [42], [103]–[106]. Colonization pressure approximates a patient's exposure to an infectious pathogen as a function of the proportion of known infected patients/patient days within a specific location over a period of time. E.g., one might compute the colonization pressure of a given unit as the proportion of infected patients identified within that unit during the preceding t days, where t can vary. Although it is generally accepted that colonization pressure is correlated with patient risk and has been shown to be a transmission risk factor [104], [105], how long an infected patient contributes to the colonization pressure of a unit can vary [103].

Because information about the incubation period of an infection, prevalence of latent spreaders, etc., may be unknown [107], we consider a data-driven solution that does not rely on such assumptions. Along these lines, prior work has proposed techniques for identifying asymptomatic but colonized spreaders resulting in better estimates of exposure [108]. However, it is limited to capturing transmission pathways that occur via co-location at the same time step. This simplifies the spatiotemporal complexity of the problem and ignores pathways such as transmission by neighbors of neighbors or transmission via lingering pathogens in the hospital environment. Thus, in contrast to prior work that has made restrictive assumptions in approximating exposure and estimating transmission events, we aim to directly estimate the transmission dynamics from a graph of patients and locations using GNNs. However, the application of GNNs is not straightforward because in many cases most of the individuals in a population will not become infected (i.e., healthcare-associated infections are relatively rare). For example, gastrointestinal infections caused by *Clostridioides difficile*, one of the most common healthcare associated infections, are associated with an incidence of approximately 1% or less in large hospitals [42], [109].

5.2.2 Graph Neural Networks (GNNs)

GNNs are a type of neural network that takes graphs as inputs and outputs a low-dimensional, learned representation that captures relevant network structures at the graph, node or edge level [21], [40], [93], [110]. When broken down to its fundamental mechanics, a GNN consists of two basic steps: i) representing the relationship between neighboring nodes and ii) aggregating these representations (e.g., averaging) to update each node's representations [111]. This is repeated at each layer of the network resulting in increasing levels of abstraction and larger receptive fields. Unlike other popular node embedding techniques that incorporate random walks [112]–[114], GNNs do not rely on random sampling, thus eliminating the possibility of missing an influential neighbor due to chance.

However, to date, GNNs have largely been evaluated on tasks that exhibit high homophily. Furthermore, many benchmark tasks tend to be well balanced in terms of class and GNNs can achieve decent performance using shallow models (3-4 layers). E.g., in the Open Graph Benchmark [115] 75% of their binary node classification tasks (ogbn-proteins) have a class balance greater than 7% and the benchmark analysis GNNs have default depths of 2-3. This is not representative of the infectious disease setting where we have higher class imbalance and potentially require large receptive fields to capture all transmission pathways.

GNN performance is known to degrade in the presence of high class imbalance or when increasing depth [20], [116], [117]. In general, class imbalance can bias the loss function towards the majority class [118]. This is also true of GNNs. However, class imbalance can also bias a GNN's learned embeddings to overfit to the majority class nodes due to architectural choices that assume high homophily tasks. Furthermore, as GNN model depth increases, model training is known to suffer from vanishing gradient and learned representations from oversmoothing. Oversmoothing is the phenomena where with increasing model depth, the learned representations of connected nodes will converge to similar values [117]. Class imbalance can also affect prediction performance through oversmoothing, as the converged representation will be more representative of the majority class.

Recently, researchers have proposed approaches that address problems exhibiting low homophily [20]. Simple modifications such as skip connections, residual connections and using higher order neighborhoods can outperform existing popular models on tasks with low homophily. Moreover, they conclude that such approaches outperform attention-based mechanisms since they specifically deal with low homophily. In contrast, we hypothesize that such approaches will not address low homophily when it arises in only the minority class but propose a simple attention-based mechanism as a solution. We formalize our problem setting and test this hypothesis in the sections that follow.

5.3 Methods

We consider the problem of estimating transmission events given information about individuals and their interactions. This corresponds to a task with high homophily overall, but low homophily with respect to the minority class. We formalize our problem setup and notation and then present details of a simple attention mechanism to address the issue of class imbalance driven low homophily.

5.3.1 Problem Set Up & Notation

Given date time-stamped information pertaining to where patients are located in the hospital and who is currently infected, we aim to estimate patient exposure and predict transmission events (*i.e.*, if and when a patient is exposed to enough of an infectious pathogen that they themselves will develop an infection). We represent interactions among patients and locations in a hospital using a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, composed of a set of nodes $\mathcal{V}$ and undirected edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The graph is heterogeneous in that its nodes can correspond to either patients or locations in the hospital, and its edges can represent either temporal or spatial connections. We describe both $\mathcal{V}$ and $\mathcal{E}$ in the paragraphs that follow.

Nodes ($\mathcal{V}$): Each node $v \in \mathcal{V}$ represents a patient or location on a specific day: $\mathcal{V} = \{\mathcal{P} \cup \mathcal{L}\}$ where $\mathcal{P}$ is the set of nodes pertaining to patients indexed by $i = 1, ..., P$ and $t = 1, ..., T$ and $\mathcal{L}$ is the set of nodes pertaining to locations indexed by j and t where $j = 1, ..., L$ and $t = 1, ..., T$. Here P and L are the number of patients and locations, respectively and T represents the total number of time steps represented by the graph. $\mathcal{L}$ includes location nodes repeated at each time point t. In contrast, $\mathcal{P}$ includes only patients corresponding to the days in which the patient was in the hospital. Thus, $|\mathcal{L}| = LT$ whereas $|\mathcal{P}| \leq PT$. We refer to the total number of nodes $|\mathcal{V}| = n$.

For each patient node $p \in \mathcal{P}$ we have a corresponding feature vector, $\boldsymbol{x}_p \in \mathbb{R}^{d_{patient}}$. And for each location node $l \in \mathcal{L}$, we have a corresponding feature vector $\boldsymbol{x}_l \in \mathbb{R}^{d_{location}}$. These feature vectors encode information such as the node type (*e.g.*, patient or location), the time step, and current known infection status. To simplify implementation, we represent the feature vectors in the same d-dimensional space, in which redundant features are removed and features that are specific to patients are set to zero for locations and vice versa.

Edges ($\mathcal{E}$): Edges connect pairs of nodes. Depending on the type of node, edges correspond to either 'spatial' connections or 'temporal' connections. Spatial edges exist between patient and location nodes and denote the presence of patients in a location and (*i.e.*, $(p_t, l_t) \in \mathcal{E}$) if the patient associated with node p_t is in the location associated with node l_t at time t. Temporal edges connect nodes associated with the same entity (be it a patient or location), over consecutive time steps: $\forall p$ and $l, (p_t, p_{t+1}), (l_t, l_{t+1}) \in \mathcal{E}$ provided that $p_{t+1}, l_{t+1} \in \mathcal{V}$.

Neighborhoods ($N(v)$): Let $u - v$ denote a sequences of nodes $u = u_1, u_2, ..., u_k = v$ such that for every $i \in \{1, ..., k-1\}, (u_i, u_{i+1}) \in \mathcal{E}$. We define a neighborhood centered at ego node v as the subset of nodes for which there exists a path to the ego node: $N(v) = \{u | u - v \in \mathcal{E}\}$. More specifically, we denote neighborhoods of varying locality ($N_m(v)$) as subsets of $N(v)$ where there exists a path of length m between member node u and ego node v. We represent neighborhoods using a neighborhood matrix A_m, where entry u, v, of the matrix is equal to 1 if $u \in N_m(v)$ and equals zero otherwise. In the case $m = 1$, A_1 is simply the adjacency matrix.

Outcome: We are interested in estimating if and when a patient is exposed to an infectious pathogen *such that* transmission occurs. We frame this as a binary prediction task in which each patient is either exposed and experiences a transmission event or does not, at each time step ($y_p \in \{0,1\}$). Once infected, a patient stays infected, therefore only one label is necessary per patient. In observational data, we often have to use the proxy of infection to determine if exposure resulted in transmission. In the case of synthetic data, we have ground truth information pertaining to transmission and know exactly when exposure resulted in transmission. We consider a dataset in which we observe patients for T time steps, during which a subset of patients are exposed and subsequently become infected. A patient node p_t is labeled $y = 1$ if the patient becomes infected at that time step t and labeled zero otherwise. We assume that y is a function of the features associated with the nodes of that individual's neighbors (which include neighbor infectious status) for some definition of neighborhood.

Prediction Task: Given a graph $\mathcal{G}$, we learn a mapping $f : \mathcal{G} \rightarrow \{0,1\}^{|\mathcal{P}|}$. f takes as input the design matrix $X \in \mathbb{R}^{n \times d}$ where each row corresponds to a feature vector associated with a different node and the neighborhood matrix $A_m \in \{0,1\}^n \times \{0,1\}^n$. Where m, the locality of the neighborhood is determined by the user but is most typically set to 1. Given these inputs, f outputs a P-dimensional vector of binary predictions, $\hat{\boldsymbol{y}}$ associated with each patient node in the graph.

5.3.2 Graph Attention Networks (GAT) for Class Imbalance Driven Low Homophily

We hypothesize that the basic GNN architecture will fail to accurately model transmission events when class imbalance is high. This leads to a scenario with high homophily overall: most nodes remain uninfected, and the uninfected ego is similar to its uninfected neighbors. However, for infected nodes, few neighbors are also infected. Thus, the sparse signal from their few infected neighbors is potentially overwhelmed by the uninfected nodes when using GNN architectures that assume all neighbors are relevant. Due to this limitation, we explore a simple attention mechanism as a potential solution.

We consider a Graph Attention Network (GAT) with K layers [21]. Such an approach assumes that there exists some neighborhood of size K around ego node v, such that the outcome y_v can be derived from the feature representations of the ego node v and its neighborhood of nodes. Each layer $k = 1, ..., K$ of the GAT updates the learned representation of node v ($\boldsymbol{r}_v^{(k)} \in \mathbb{R}^{d_k}$) via function $f^{(k)}$:

$$f^{(k)} = ReLU(ATTN(A_1, R^{(k-1)}; W^{(k)})R^{(k-1)}\theta^{(k)}), \tag{5.1}$$

Here, $R^{(k)}$ is the ($n \times d_k$) matrix of learned representations, and $W^{(k)}$ and $\theta^{(k)}$ are the parameters of a learned linear transformation. At the first layer, $k = 1$, the input representation $\boldsymbol{r}_v^{(0)} = \boldsymbol{x}_v$. Each layer of the GAT is parameterized by $\theta^{(k)}$, each self attention layer of the GAT is parameterized by $W^{(k)}$. These parameters are learned in an end-to-end manner by minimizing the cross entropy loss between y_v and $\hat{y}_v = \arg\max(\log \text{softmax}(\boldsymbol{r}_v^{(K)}\theta))$. Note that our input is n dimensional since we have both location and patient nodes. However, we only make predictions for patient nodes, such that $\hat{\boldsymbol{y}}$ is P dimensional.

A self attention mechanism (ATTN) is applied to the input representations in order to learn the pairwise importance of neighbors [21], [79]. This helps identify which neighbors the model should pay more attention to. This is illustrated in **Figure** 5.1 (a). The GAT updates each node's representation by: (1) "ATTENTION" - calculating self attention on $R^{(k-1)}$ ($R^{(0)} = X$) in order to re-weight A_1, (2) "AGGREGATE" - taking an attention weighted average of their ego and neighbor representations ($ATTN(A_1, R^{(k-1)}; W^{(k)})R^{(k-1)}$, and (3) "TRANSFORM" - applying a linear transformation followed by a nonlinear ReLU. This process is applied multiple times via multiple attention blocks whose outputs are then averaged ("AGG: MEAN"). We hypothesize that this will lead to improved performance in the presence of class imbalance driven low homophily by allowing the network to focus on the few but highly important, infected individuals that affect a patient's outcome. In our implementation, we use a scaled dot-product attention [79] in our multi-headed attention, using the adjacency matrix as a mask.

5.4 Experimental Setup

Applied to both synthetic and clinical datasets, we evaluate GNNs equipped with a simple attention mechanism in terms of accuracy in estimating transmission of infectious disease. We compare to several recently proposed approaches described below.

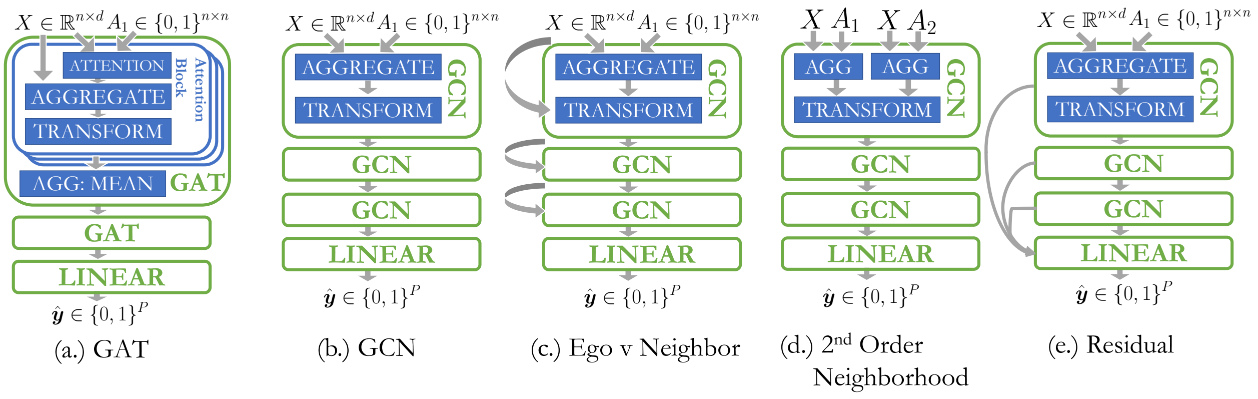


FIGURE 5.1: Visualizations of the different GNN architectures and variants described in Section 5.3. Unlike the basic GCN (b.), (c.) and (e.) contain additional connections, (d.) has additional inputs and (a.) has additional attention functions and attention blocks. These are to aid in understanding the differences between the multiple architectures described and should not be taken as literal representations of final GNN architectures described in experimental results. Note, "AGGREGATE" is sometimes shortened to "AGG" due to space limitations.

5.4.1 Comparisons

We compare the performance of an attention-based approach to a number of variations on the GNN that were recently proposed to address low homophily [20].

Graph Convolutional Networks (GCN). A commonly used form of GNN, Graph Convolutional Networks (GCN) can be formulated as [40]:

$$f^{(k)} = ReLU(D^{-\frac{1}{2}}A_1D^{-\frac{1}{2}}R^{(k-1)}\theta^{(k)}) \tag{5.2}$$

where D is the diagonal node degree matrix of A_1 and $D^{-\frac{1}{2}}A_1D^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix and we will refer to it as $\tilde{A}_1$ henceforth. This is further illustrated in **Figure** 5.1 (b).

Separating ego vs neighbor representations (Ego vs. Neighbor). Often the ego represented is processed no differently than the representation of its neighbors. In this variation, the ego representation is passed in separately from the neighbor representation.

More formally, Eqn. 5.2 is replaced with with Eqn. 5.3:

$$f^{(k)} = ReLU([R^{(k-1)}||\tilde{A}R^{(k-1)}]\theta^{(k)}). \tag{5.3}$$

Here the double vertical lines indicate a concatenation along the feature dimension. This essentially introduces a skip-connection between the input and output of the aggregation step of the k^{th} layer of the GNN (**Figure** 5.1 (c)). Such skip connections improve performance in settings of low homophily by allowing the model to learn ego specific parameters and thereby process the representation of the ego separately from its dissimilar neighbors.

Higher order neighborhoods (2^{nd} Order Neighborhood). In this variation, we explore the impact of incorporating higher order neighborhoods in addition to $N_1(v)$. In this setting, the update in Eqn 5.2 is replaced with the update in Eqn. 5.4:

$$R^{(k)} = ReLU([\tilde{A}_1 R^{(k-1)}||\tilde{A}_2 R^{(k-1)}||...||\tilde{A}_M R^{(k-1)}]\theta^{(k)}) \tag{5.4}$$

where $\tilde{A}_m$ refers to the symmetrically normalized neighborhood matrix associated with the m^{th} order neighborhood. An example network for $m = 2$ can be seen in **Figure** 5.1 (d). This allows the model to process representations of different order neighborhoods separately and can help in scenarios of low homophily when different order neighborhoods have higher levels of homophily [20].

Residual Connections (Residual). Residual connections provide a pathway for intermediate level representations to directly impact the final representation (**Figure** 5.1 (e)). This variant affects the final learned representation in the following manner:

$$\hat{\boldsymbol{y}} = \arg\max(\log \text{softmax}([R^{(0)}||R^{(1)}||...||R^{(K)}]\theta)) \tag{5.5}$$

Each successive layer of a GNN increases the receptive field. Therefore, the output of each layer represents a different level of locality. Residual connections separate out representations at different levels of locality, thereby increasing the representative power of models. Under scenarios of low homophily, this representation allows the model take advantage of differences in label distribution at different localities.

5.4.2 Datasets

We evaluate the proposed approach in the context of estimating pathogen exposure and subsequent infection using both clinical and synthetic datasets.

Clinical Dataset. In our clinical data experiments, we utilize electronic health record data from Michigan Medicine to predict hospital onset *Clostridioides difficile* infection (CDI). One of the most common healthcare-associated infections, CDI is known to persist on hospital surfaces for weeks [119]. Our study cohort includes all adult inpatient admissions from 2016-2019 with a higher risk for hospital acquired CDI (i.e., a length of stay ≥ 7 days). This work was approved of by our institutional review board (IRB).

Our study population consists of 44,393 unique patients, in which 397 or 0.9% developed CDI (determined through a positive laboratory test [42]). In addition to these patient data, we extract information pertaining to 135 unit locations. Although we consider both patients and locations, it is the number of days that is the main driver of network size. Given the potential size of the graph, for computational efficiency, we represent these data using multiple graphs spanning 3 day time periods ($T = 3$). This results in graphs with an average $|\mathcal{V}| = 1371$ and an average $|\mathcal{E}| = 2097$. As described in the problem setup, each node in the graph is associated with a feature vector $\boldsymbol{x}_v$, encoding: (1) if the node is a patient or a location, (2) the time step t, and (3) an indicator value for a positive CDI lab result that linearly decays from 1 to 0 over the course of 30 days.

Although three days may seem limiting, current methods such as colonization pressure rely solely on co-location (i.e., a graph of $T = 1$). Note that although the considered interactions are limited to the current time step, patient history used in the calculation of colonization pressure is not. By increasing T, we can capture more complex interactions beyond co-location, such as exposure from yesterday's neighbor's neighbor or previous occupants of a location.

When using synthetic data, we know exactly who became exposed to the pathogen and when. For clinical data, this information in unavailable. Thus, we rely on a proxy: infection status. We know that a patient must be both susceptible and exposed to become infected. Relying on infection status will underestimate exposure, but is clinically relevant nonetheless. Moreover, if exposed, we do not have ground truth on precisely *when* a patient was exposed to the pathogen. Thus, we aim to make predictions at the patient level, not the patient time-step level. Specifically, we aim to estimate if a patient was exposed in the first T time steps and as a result will develop an infection after time step T. Every node associated with a patient during their hospital visit is labeled 1, and zero otherwise.

Synthetic Network Dataset Experiments with synthetic data allow us to control the transmission dynamics and access ground truth exposure. Using synthetic data, we evaluate to what extent GNNs can accurately learn useful representations of exposure as we increase the complexity of the transmission dynamics. Synthetic hospital networks are generated using a stochastic block model [120]. Networks consist of 10 sub-communities of 5-20 patient nodes and a network represents a single hospital day ($T = 1$). In this scenario, we simplify the problem setting and only consider patient nodes. Within a sub-community, patient nodes share an edge with probability 0.5; across sub-communities patient nodes share and edge with probability 0.01. These parameters were selected to mimic how patients in similar units are more likely to interact with each other than patients of different units.

Each node is labelled according to $y_p \sim Bernoulli(\phi_p)$, where Φ is an aggregation function and $\mathbf{c}$ (contribution to colonization pressure) is a function of their neighbors infection status $\mathbf{x}$ and time of infection $\mathbf{t}$, $\phi_p = \Phi(\{\mathbf{c}(\mathbf{x}_u, \mathbf{t}_u) : u \in N_m(p) \land u \neq p\})$. In our experiments, we explore different functions Φ and $\mathbf{c}$ (e.g., the mean of a linear decay).

20% of patients are randomly selected to have a positive infection status ($\mathbf{x}_i = 1$) which results in a mean class imbalance of 5.2% and an edge homophily ratio of 0.13 when only looking at edges involving the minority class. Each positive infection status is randomly assigned an infection time $\mathbf{t} \in \{T - 30, T - 29, ..., T\}$. This time step determines the pressure an infected patient exerts on its neighbors. A patient whose positive status is associated with 30 days prior to T is less influential than a patient whose status is associated with 1 day prior to T. An infected patient's influence decreases as they begin to be treated and contact precautions are implemented. Patients who are known to be positive at the time of prediction are excluded from evaluation. We generate 1000 networks for training and 20 networks for validation and testing for each problem setting. This resulted in an average of 118.5 nodes and 434.3 edges per graph. Additional details on data generation can be found in **Appendix B.2**.

5.4.3 Training & Evaluation Details

Most node classification benchmarks are set up as semi-supervised learning problems on a single large graph $\mathcal{G}$, whose nodes are split into train, validation and test. Instead, we have multiple graphs that are split into training, validation and test. This prevents information leakage from training to test as each graph covers a different time frame.

All models were trained using ADAM with a weighted cross entropy loss. Loss is weighted based on the average class imbalance found in the training and validation sets so that on average, each class equally influences the loss. Loss is regularized using a L2 loss with a 0.001 weight and dropout was applied during training with a rate of 0.01.

During training, we used a patience of 5 epochs for early stopping and for the learning rate schedule. The first time 5 epochs pass without a decrease in the validation loss, the learning rate is stepped down from the default 0.001 to 0.0001. After 5 epochs passes for the second time without a decrease in the validation loss, training is stopped. We selected the best model based on validation area under the receiver operating characteristics curve (AUROC).

GNN hyperparameters were tuned using validation performance optimizing for AUROC. For all GNNs, we searched over depths of $\{1, 2\}$ for the synthetic task and $\{1, 2, 3, 4, 5\}$ for the clinical task. Twenty-four models were trained for each model architecture for a maximum of 160 epochs for synthetic tests and 80 epochs for clinical tests. For the GCN, the size of the hidden layers was set to the same value ($d_1 = d_2 = ... = d_K$); we searched over values of $\{100, 200, 400, 800\}$. For the GAT, the alpha parameter was set to 0.2 (the setting used in the original paper [21]), while the number of heads (swept over values $\{1, 3, 5\}$) and hidden layer size (swept over values $\{50, 100, 200, 400\}$) was chosen via validation performance. To reduce oversmoothing the learned node embeddings we employed PairNorm [117].

Evaluation on Real-World Data. We split the data into consecutive three day periods represented by separate graphs. For each graph, time granularity is at the day level, location granularity is at the unit level and $T = 3$. During implementation, although n estimates are generated, only those pertaining to a subset of our study population are used in training and evaluation. Unused values are masked. Our study population consists of two groups. The first group is used for training and evaluation and consists of a subset of patient nodes who have been in the hospital for at least 7 days. These patients are more likely to be impacted by within hospital transmissions. The second group consists of patients we think are likely to be spreaders: patients who are known to be positive before the time of prediction, and patients with a recent history of CDI. We do not train or evaluate on this second group. Given that we only have patient level labels, we also evaluate at the patient level, such that no patient is represented more than once during evaluation. We calculate the patient level AUROC by taking the maximum score for a patient during their hospital stay: $\hat{y}_i = \max(\{\hat{y_{i,t}} : t = 1, ..., T\})$ [121]. This mimics a realistic deployment where multiple scores are generated for each individual but once an individual crosses a threshold they are deemed high risk and receive an intervention.

Evaluation Synthetic Networks. Similar to the clinical dataset evaluation, patients known to be infected prior to T were included in the network but did not contribute to the loss function during training or evaluation. Compared to the clinical dataset evaluation, here we had access to ground truth for each time step. Thus, labels were evaluated at each time step t.

5.5 Experiments & Results

In the context of modeling transmission events, we compare an attention-based approach to several recently proposed variations on the GNN designed to handle tasks with low homophily. We apply this attention-based approach to both clinical and synthetic datasets in which the transmission dynamics vary, characterizing the settings in which a GNN can accurately model transmission. In doing so, we aim to answer the following questions:

- Question 1: When applying GNNs, does an attention-based solution address the issue of class imbalance-driven low homophily? **(Section 5.5.1)**
- Question 2: How does an attention-based solution compare or complement recently proposed variations on the GNN in this setting? **(Section 5.5.1)**
- Question 3: Does a data-driven graph-based approach to learning transmission dynamics outperform an approach based on an expert-defined notion of colonization pressure? **(Section 5.5.2)**
- Question 4: What types of transmission dynamics are readily learned by a graph-based approach? **(Section 5.5.2)**

5.5.1 GAT versus GCN (and other Variations)

Applied to held-out data from the clinical dataset, consisting of 12,343 patients in which 0.7% of the population is infected, the attention-based approaches significantly outperformed the base GCN: AUROC=0.715 (95% Confidence Interval [CI] 0.687-0.744) versus 0.684 (95% CI 0.659-0.710). Allowing the model to weight the importance of different neighbors in a variable manner improved performance on this task in which only a small fraction of an individual's neighbors influence their outcome. This provides empirical evidence that supports our hypothesis that an attention-based solution can help mitigate class imbalance driven low homophily in GNNs (**Q1**).

TABLE 5.1: Performance of GNN architectural variants on predicting infectious disease exposure in a clinical dataset. We find that GAT has the best test AUROC, while a GCN+++ which incorporates all variations except attention has the second best performance out of the tested GCN architectures. The tested GCN architectures are known to improve performance under low homophily. However, an architecture like GAT that has a modified aggregation function may be better suited for scenarios with class imbalance driven low homophily. We see that GAT significantly outperform the GCN.

Model	AUROC (95% Confidence Interval)
GAT	0.715 (0.688,0.742)
GCN (basic)	0.684 (0.659,0.710)
GCN + (Ego v. Neighbor)	0.696 (0.670,0.726)
GCN + (2^{nd} Order Neighborhood)	0.681 (0.652,0.713)
GCN + (Residual)	0.706 (0.679,0.735)
GCN +++	0.709 (0.682,0.737)

With respect to all of the variations we considered, performance ranged from 0.681 to 0.715 (**Table** 5.1). While differences are small, a few trends emerge. First, GAT statistically significantly outperforms the GCN and outperforms all other variations. Although this second difference is not always statistically significant, it is consistent (**Q2**).

Second, the augmented architectures (*e.g.*, Ego v. Neighbor, 2^{nd} Order Neighborhood) generally improve predictive performance over the basic GCN performance except for adding higher (2^{nd}) order neighbors. In contrast to the other two variants (Ego v. Neighbor (skip connection) and Residual), adding higher order neighbors does not necessarily aid gradient flow during training [122]. The most beneficial single augmentation to the

GCN was the addition of residual connections (e.g., 0.022 increase in AUROC). In addition to aiding gradient flow, residual connections provide more granularity with respect to neighborhood locality. Output from different model depths are provided to the final layer, and each layer's output corresponds to different neighborhood sizes. This may be especially beneficial to estimating exposure as likelihood of transmission decreases the further away two nodes are from each other.

5.5.2 Learning Transmission Dynamics

Here, we further test the ability of the GAT to accurately model complex transmission dynamics by comparing performance against hand-coded definitions of exposure in both real-world and synthetic networks. Using the GAT, we compare the predictive utility of a learned (or data-driven) definition of transmission to one based on an expert defined (or hand-coded definition) of exposure. We also explored the other GNN variants, however based on validation performance we proceed with GAT in all subsequent experiments (see **Appendix B.1** for validation performance).

Our hand-coded definition of exposure is informed by the infectious disease literature and based on the notion of *colonization pressure* described earlier. More formally,

$$\hat{\phi}_p = \sum_{\{u : u \in N_2(p) \wedge u \neq p\}} \mathbf{c}(\mathbf{x}_u, \mathbf{t}_u), \tag{5.6}$$

where $N_2(p)$ are neighbors of patient p who are known to be positive prior to or on day t and $\mathbf{c} : \mathbb{Z}_+^2 \rightarrow [0, 1]$, node u's contribution to the colonization pressure, which linearly decays over 14 days from the day of the positive test. We consider two neighborhoods based on co-locations, at the unit and hospital level. This results in two estimates of exposure. These estimates are then used as features in training a linear model to predict infection. For a fair comparison, the logistic regression is trained and tested using a patient representation that concatenates data from all available time steps prior to the time of prediction: $[\boldsymbol{x}_{i,1}||, ..., ||\boldsymbol{x}_{i,t}]$ for prediction time t.

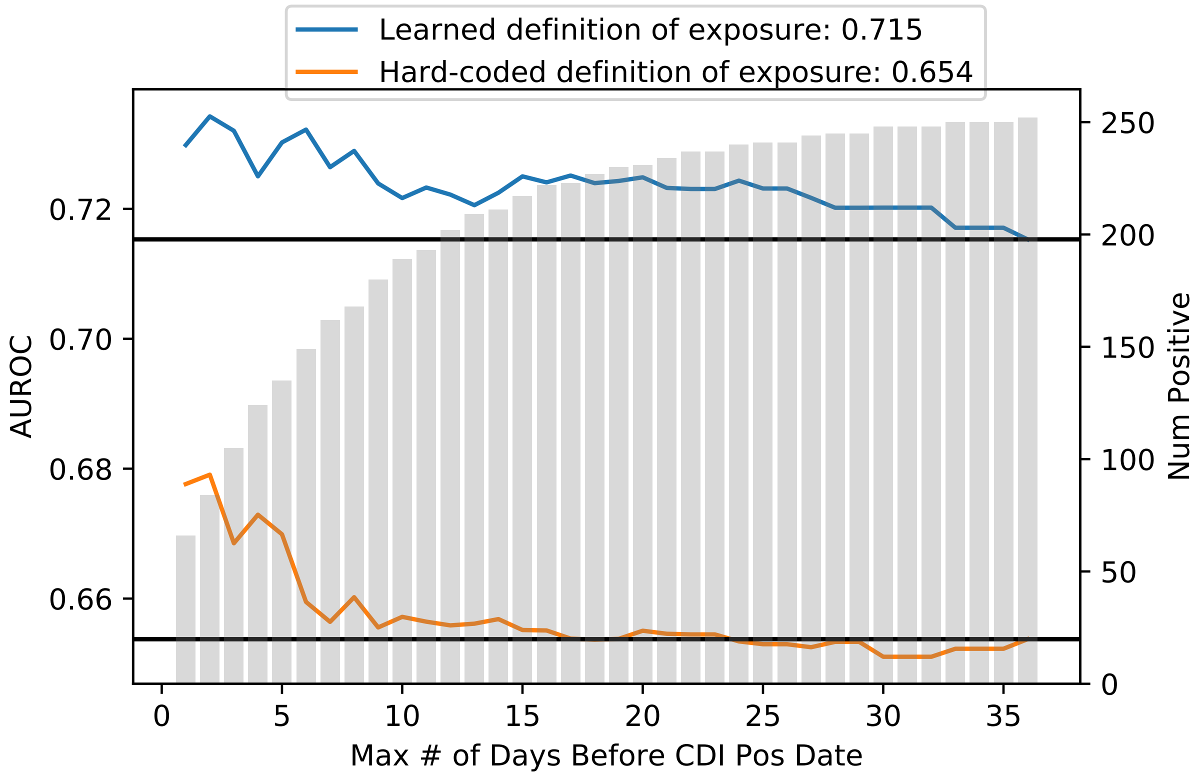


FIGURE 5.2: AUROC of a learned definition of exposure (0.715) vs hand-coded definition of exposure (0.654) at predicting risk of CDI. Results are further broken down by the number of days before the CDI diagnosis. Both models have better performance when diagnosis is made soon after the day of prediction.

Results on Clinical Dataset

Applied to the held-out test set of 12,343 patients in which 84 tested positive for CDI, the learned definition of exposure (GAT) achieves significantly higher discriminative performance compared to the hand-coded definition 0.715 (95% CI: 0.688,0.742) vs 0.654 (95% CI: 0.625,0.680). This provides strong empirical evidence in support of our hypothesis that a data-driven graph-based approach can lead to improved estimates of exposure compared to an expert-defined hand-coded estimate (**Q3**).

In our problem setting, we observed each patient for T time steps and then try to predict whether the patient will develop an infection after T. In a follow-up analysis, we

TABLE 5.2: We compare the performance a learned-definition of exposure (GAT) to exact (ϕ) and inexact hand-coded definitions of exposure ($\hat{\phi}$). When the misspecified hard-coded definition either includes additional irrelevant pathways or excludes relevant pathways (e.g., considering all known positive neighbors instead of just symptomatic neighbors or ignoring 2^{nd} order neighbors) a learned definition can perform significantly better than a hand-coded definition of exposure.

Scenario	Class Imbalance	Homophily Homophily$_+$	AUROC (95% Confidence Interval)		
			True ϕ	$\hat{\phi}$	GAT
ϕ: 1^{st} & 2^{nd} Order Neighbors $\hat{\phi}$: 1^{st} Order Neighbors	5%	0.765 0.092	0.751 (0.745, 0.756)	0.609 (0.601, 0.617)	**0.694** (0.634,0.757)
ϕ: Mean $\hat{\phi}$: Max	5%	0.774 0.117	0.851 (0.848, 0.854)	**0.827** (0.824, 0.831)	0.739 (0.695,0.784)
ϕ: Decay Period 14 Days $\hat{\phi}$: Decay Period 3 Days	5%	0.774 0.117	0.851 (0.848, 0.854)	0.663 (0.657, 0.669)	**0.739** (0.695,0.784)
ϕ: Exponential Decay $\hat{\phi}$: Linear Decay	3%	0.791 0.097	0.886 (0.883, 0.890)	**0.881** (0.878, 0.885)	0.835 (0.786,0.887)
ϕ: Symptomatic Neighbors $\hat{\phi}$: All known positives Neighbors	8%	0.869 0.249	0.986 (0.985, 0.986)	0.855 (0.852, 0.857)	**0.927** (0.904,0.950)

measured predictive performance on subsets of the population based on when they developed the infection (e.g., soon after T or much later). In **Figure** 5.2, we sweep the a cutoff corresponding to the maximum number of days after T before the patient becomes infected. Intuitively, as we increase the temporal gap between observation and outcome, we'd expect the task to become more difficult. I.e., patients who develop infection sooner are likely easier to identify/predict because they are more likely to have been exposed during the period used to predict the outcome: $[1, T]$. We see this evidenced by both the GAT and logistic regression's improved performance on patients with earlier test dates. Estimating transmission, and in turn predicting infection, is only possible when the transmission event(s) occurs in the observation window.

Still, the GAT-based definition of exposure performs well at predicting transmission and subsequent infection, even without any additional information pertaining to patient susceptibility. We are particularly encouraged by how it performs relative to the hand-coded definition of exposure in estimating CDI cases in a 5-10 days horizon. Performance of the hand-coded definition of exposure quickly degrades for longer horizons, whereas the learned definition can accurately predict more difficult cases.

Results on Synthetic Networks

In the experiments above, we based the hand-coded definition of exposure on definitions commonly used in the infectious diseases literature [103]. However, the true underlying transmission dynamics of CDI (and many other infectious diseases) are often unknown. I.e., the hand-coded definition depends on assumptions which may or may not hold; it is only an approximation, and perhaps a poor approximation given the associated predictive performance.

Thus, in our final set of experiments, we explore the ability of the GAT to accurately learn a definition of exposure when the transmission dynamics vary in complexity. We use synthetic data because it allows us to control the true underlying transmission dynamics and explore an arbitrary degree of complexity. As a comparison, we also measure the performance a perfect hand-coded definition exposure and an imperfect hand-coded definition based on an incorrect assumption. For example, in one setting we assume a transmission event is driven by maximum exposure, when in fact it is driven by average exposure. Since in practice definitions of transmission dynamics are often approximations, this experiment helps us understand under what circumstances learning a definition of exposure could more accurate than an imperfect hand-coded definition.

We vary the complexity of the transmission dynamics and the accuracy of the predefined exposure formulation as follows. Given our formulation of a patient's exposure, $\phi_p = \Phi(\{\mathbf{c}(\mathbf{x}_u, \mathbf{t}_u) : u \in N_1(p) \wedge u \neq p\})$, we modify the aggregation function Φ, the calculation of a neighbor's contribution $\mathbf{c}$ and the definition of neighborhood $N(p)$. We use the true ϕ to perfectly estimate the underlying transmission dynamics and compare to an inexact hand-coded formulation of exposure ($\hat{\phi}$). A positive patient is symptomatic with a probability of 0.25, and the amount each patient contributes to the overall exposure of others (c) is based on whether the individual is symptomatic. For the neighborhood definition we consider $N_1(p)$ vs $N_2(p)$. We quantify the effect of different types of (potentially common) misspecifications against ground truth (ϕ vs $\hat{\phi}$): mean vs max aggregation (Φ), and for the contribution calculation, we vary the decay over 14 vs 3 days (i.e., gradual versus quick decay), exponential vs linear decay, and contribution conditioned on neighbor characteristics (symptomatic vs all positive patients).

For each scenario in Table 5.2, we measured class imbalance, edge homophily ratio (the fraction of intra-class edges in a network), as well as a modified edge homophily ratio that only looks at edges involving the minority class (Homophily$_+$). When calculating homophily, we considered exposed patients and infected/spreading patients to belong to the same class. Each scenario has high overall homophily, while the minority class specific homophily ratio is low.

Compared to the performance of the hand-coded definitions, the performance of a GAT approximates our ability to automatically learn an accurate data-driven definition of exposure using only information about the patient network (i.e., who's connected to whom and patient characteristics). In the majority of scenarios, the GNN performs as well as or better than the approximate hand-coded definition of exposure (**Table** 5.2). In particular, in scenarios in which the hand-coded definition of exposure $\hat{\phi}$ has too many pathways (e.g., including asymptomatic patients) or is missing pathways (e.g., those who were positive more than 3 days ago and those who are only second order neighbors) the GAT achieves a significantly higher AUROC (**Q4**). By simply incorporating all potential pathways into the network, one can learn an accurate definition of exposure directly from the data. This is especially important because both missing pathways and extraneous pathways are detrimental. In cases in which all and only relevant pathways are included, the hand-coded definition of exposure is closer to the true definition (e.g., max vs. mean and linear vs. exponential) and the relative performance of the hand-coded definition improves.

5.6 Summary and Conclusion

We investigate the application of GNNs to modeling the transmission of infectious disease. We highlight the challenge of class imbalance driven low homophily that arises in such tasks and present a simple attention mechanism as a solution. While previous work concluded that an attention based method would fail in scenarios with low homophily, we showed that when low homophily is driven by class imbalance, attention can help performance and outperform previously proposed solutions. Moreover, such an approach can effectively learn complex transmission dynamics. Overall, when estimating disease transmission data-driven definitions of exposure present a promising alternative to hand-coded definitions that rely on potentially inaccurate assumptions about which transmission pathways to include vs exclude.

Chapter 6

Conclusion

This dissertation addressed the challenges of small sample sizes when adapting machine learning methods to tasks utilizing EHR data. There are vast amounts of data being collected in the EHR. With the help of machine learning, these data have the potential to provide great insight into patient specific risks and conditions. Deep learning has been popular in the recent decade due to its ability to learn useful representations with minimal prior knowledge. However, this is predicated on the availability of a large amount of training examples. The EHR poses an interesting dichotomy between small and big data: where within thousands of admissions, there may only be a handful of examples for the outcome of interest. This poses issues for deep learning methods that rely on large sample sizes to learn complex relationships without overfitting and in some cases may rely on balanced data assumptions.

This dissertation builds on work spanning several fields, including time-series analysis and inference on graphs. While there exists methods for adapting to known temporal invariances [11], [12], this can be difficult when the potential invariances are not known beforehand. Furthermore, while there exists methods that handle conditional shift, these tend to focus on the multitask setting [15], [63]–[65]. When conditional shift is driven by time, the distinction between tasks is less clear as the change in task happens gradually over time. In addition, popular methods for handling time-series data, such as RNNs, rely on shared weights across time steps. This can lead to poor adaptation to time varying dynamics driven by temporal conditional shift [16]. Finally, graph neural networks are generally evaluated on tasks with high homophily and often make architectural assumptions based on this underlying characteristic [20]. However, clinical tasks often exhibit high class imbalance which results in an asymmetrical case of low homophily with respect to the minority class.

Building on this past work, our dissertation has contributed new deep learning approaches for learning useful representations from clinical time-series and graph data by addressing task specific structure. In order to leverage temporal invariances, **we proposed Sequence Transformer Network**, an end-to-end trainable network that learns and applies patient and task specific transformations to reduce class independent signal variations (**Chapter 3**). In order to adapt to temporal conditional shift, **we proposed a mixture of LSTMs** that relaxes weight sharing in LSTMs. This allows the direct modeling of time varying signal dynamics, improving performance especially under constrained data settings (**Chapter 4**). Finally, to address class imbalance driven low homophily using graph neural networks, **we evaluated the ability of attention-based aggregation** against recently proposed low homophily solutions (**Chapter 5**). Throughout this dissertation we explored clinically relevant problem settings and proposed methods to best adapt popular deep learning methods to those settings.

There are several areas touched upon in this dissertation that could be interesting for further examination. Here we outline a few possibilities.

First, our work on capturing invariances via the Sequence Transformer had limitations. Namely it did not consider feature-specific transformations or local warping. Since publication, more recent work has looked at capturing local warping [23]. However, feature specific transformations still presents the challenge of incorporating the benefits of increased transformation flexibility while preserving valuable temporal relations between events. Only applying certain transformations at the feature level (*i.e.*, magnitude) or providing both original and transformed representations present potential routes for further investigation.

Second, on our work estimating exposure using graph neural networks had some areas for potential future work. We used a network that captured both space and time relationships by representing each patient and location at each time step using a separate node. While this allowed us to capture transmission pathways that cut across space and time, each additional time step considered significantly increased the network size and therefore the required computational memory.

Second, whenever applying deep learning to critical tasks such as those involving patient care, there is always the question of accountability and assurance. Models are only as good as the data they are trained on and are liable to make decisions that perpetuate inequitable social norms [123] or identify patterns that are not robust or generalizable [124]. This is especially true when applying black box methods such as deep learning models where it's difficult to spot these issues. There is an ongoing field in model interpretability that aims to help mitigate this problem as well as ever evolving best practices to help researchers pose the right questions and obtain robust results [31], [125], [126]. However, there is still much to be done in this area to ensure models are safe for deployment.

Finally, one major limitation of this dissertation is that we did not touch upon real time evaluations or deployment efforts. The development of risk estimators provides benefits such as insights into the data (i.e., patterns and dynamics present) and the identification of useful cohorts and risk factors. However, it is through deployment that risk estimators are able to aid real time clinical decision making. Post-development there are still many considerations before deployment. One of the most important aspects is vetting the model through prospective or real time evaluation. Retrospective evaluation is not guaranteed to reflect performance during deployment due to factors such as data imputation, data collation, retrospective additions, data leakage, etc. We are currently in the process of prospectively evaluating one of our risk scores [42] at Michigan Medicine and hope to publish results in the near future.

The main contributions of this dissertation are: 1) presenting and formalizing problem settings commonly found in clinical tasks and 2) addressing these problem settings through the development of machine learning algorithms. We emphasized an overarching problem in clinical tasks, namely limited training data. Limited data can lead to reduced generalizability and robustness of deep learning models. We addressed this through the development of novel methods that leveraged specific task structures. Highly imbalanced data can lead to pitfalls when using generic deep learning models such as GCNs. We addressed this issue by evaluating attention based and other architectural variants in this problem setting. Together, these contributions lead to improved predictive performance when training with limited data; once incorporated into clinical decision-making workflows such models could ultimately lead to improved patient care.

Appendices

Appendix A

Appendix for Chapter 4: Temporal Conditional Shift

A.1 Details of Data & Features

TABLE A.1: The 17 physiological features extracted from MIMIC-III database, the source tables, and the corresponding ITEMIDs

Index	Variable Name	Table(s)	ITEMID(s)
1	Capillary refill rate	CHARTEVENTS	3348, 115, 8377
2	Diastolic blood pressure	CHARTEVENTS	8368, 220051, 225310, 8555, 8441, 220180, 8502, 8440, 8503, 8504, 8507, 8506, 224643
3	Fraction inspired oxygen	CHARTEVENTS	3420, 223835, 3422, 189, 727
4	Glascow coma scale eye opening	CHARTEVENTS	184, 220739
5	Glascow coma scale motor response	CHARTEVENTS	454, 223901
6	Glascow coma scale total	CHARTEVENTS	198,
7	Glascow coma scale verbal response	CHARTEVENTS	723, 223900
8	Glucose	CHARTEVENTS + LABEVENTS	50931, 807, 811, 1529, 50809, 51478, 3745, 225664, 220621, 226537
9	Heart Rate	CHARTEVENTS	221, 220045
10	Height	CHARTEVENTS	226707, 226730, 1394
11	Mean blood pressure	CHARTEVENTS	52, 220052, 225312, 224, 6702, 224322, 456, 220181, 3312, 3314, 3316, 3322, 3320
12	Oxygen saturation	CHARTEVENTS + LABEVENTS	834, 50817, 8498, 220227, 646, 220277
13	Respiratory rate	CHARTEVENTS	618, 220210, 3603, 224689, 614, 651, 224422, 615, 224690
14	Systolic blood pressure	CHARTEVENTS	51, 220050, 225309, 6701, 455, 220179, 3313, 3315, 442, 3317, 3323, 3321, 224167, 227243
15	Temperature	CHARTEVENTS	3655, 677, 676, 223762, 3654, 678, 223761, 679
16	Weight	CHARTEVENTS	763, 224639, 226512, 3580, 3693, 3581, 226531, 3582
17	pH	CHARTEVENTS + LABEVENTS	50820, 51491, 3839, 1673, 50831, 51094, 780, 1126, 223830, 4753, 4202, 860, 220274

TABLE A.2: The 76 time-series features used as input to all the models.

Index	Feature Name	Type
0	Capillary refill rate->0.0	Binary
1	Capillary refill rate->1.0	Binary
2	Diastolic blood pressure	Numeric
3	Fraction inspired oxygen	Numeric
4	Glascow coma scale eye opening->To Pain	Binary
5	Glascow coma scale eye opening->3 To speech	Binary
6	Glascow coma scale eye opening->1 No Response	Binary
7	Glascow coma scale eye opening->4 Spontaneously	Binary
8	Glascow coma scale eye opening->None	Binary
9	Glascow coma scale eye opening->To Speech	Binary
10	Glascow coma scale eye opening->Spontaneously	Binary
11	Glascow coma scale eye opening->2 To pain	Binary
12	Glascow coma scale motor response->1 No Response	Binary
13	Glascow coma scale motor response->3 Abnorm flexion	Binary
14	Glascow coma scale motor response->Abnormal extension	Binary
15	Glascow coma scale motor response->No response	Binary
16	Glascow coma scale motor response->4 Flex-withdraws	Binary
17	Glascow coma scale motor response->Localizes Pain	Binary
18	Glascow coma scale motor response->Flex-withdraws	Binary
19	Glascow coma scale motor response->Obeys Commands	Binary
20	Glascow coma scale motor response->Abnormal Flexion	Binary
21	Glascow coma scale motor response->6 Obeys Commands	Binary
22	Glascow coma scale motor response->5 Localizes Pain	Binary
23	Glascow coma scale motor response->2 Abnorm extensn	Binary
24	Glascow coma scale total->11	Binary
25	Glascow coma scale total->10	Binary
26	Glascow coma scale total->13	Binary
27	Glascow coma scale total->12	Binary
28	Glascow coma scale total->15	Binary
29	Glascow coma scale total->14	Binary
30	Glascow coma scale total->3	Binary
31	Glascow coma scale total->5	Binary
32	Glascow coma scale total->4	Binary
33	Glascow coma scale total->7	Binary
34	Glascow coma scale total->6	Binary

35	Glascow coma scale total->9	Binary
36	Glascow coma scale total->8	Binary
37	Glascow coma scale verbal response->1 No Response	Binary
38	Glascow coma scale verbal response->No Response	Binary
39	Glascow coma scale verbal response->Confused	Binary
40	Glascow coma scale verbal response->Inappropriate Words	Binary
41	Glascow coma scale verbal response->Oriented	Binary
42	Glascow coma scale verbal response->No Response-ETT	Binary
43	Glascow coma scale verbal response->5 Oriented	Binary
44	Glascow coma scale verbal response->Incomprehensible sounds	Binary
45	Glascow coma scale verbal response->1.0 ET/Trach	Binary
46	Glascow coma scale verbal response->4 Confused	Binary
47	Glascow coma scale verbal response->2 Incomp sounds	Binary
48	Glascow coma scale verbal response->3 Inapprop words	Binary
49	Glucose	Numeric
50	Heart Rate	Numeric
51	Height	Numeric
52	Mean blood pressure	Numeric
53	Oxygen saturation	Numeric
54	Respiratory rate	Numeric
55	Systolic blood pressure	Numeric
56	Temperature	Numeric
57	Weight	Numeric
58	pH	Numeric
59	mask->Capillary refill rate	Binary
60	mask->Diastolic blood pressure	Binary
61	mask->Fraction inspired oxygen	Binary
62	mask->Glascow coma scale eye opening	Binary
63	mask->Glascow coma scale motor response	Binary
64	mask->Glascow coma scale total	Binary
65	mask->Glascow coma scale verbal response	Binary
66	mask->Glucose	Binary
67	mask->Heart Rate	Binary
68	mask->Height	Binary
69	mask->Mean blood pressure	Binary
70	mask->Oxygen saturation	Binary
71	mask->Respiratory rate	Binary
72	mask->Systolic blood pressure	Binary
73	mask->Temperature	Binary

74	mask->Weight	Binary
75	mask->pH	Binary

Appendix B

Appendix for Chapter 5: Exploring Class Imbalance Driven Low Homophily

B.1 Validation Results for Real-World Data

Table B.1 presents validation performance for GNN architectural variants on the task of prediction infectious disease exposure using clinical data.

TABLE B.1: Validation performance of GNN architectural variants on predicting infectious disease exposure in a clinical dataset. Rows are ordered to mimic the ordering found in **Table 5.1**. We find that GAT has the best validation AUROC.

Model	AUROC (95% Confidence Interval)
GAT	0.6624 (0.6377,0.6869)
GCN	0.6525 (0.6295,0.6771)
GCN Ego v. Neighbor	0.6505 (0.6290,0.6739)
GCN 2^{nd} Order Neighborhood	0.6412 (0.6183,0.6630)
GCN Residual	0.6503 (0.6244,0.6743)
GCN All Variants	0.6516 (0.6295,0.6740)

B.2 Synthetic Data Generation

Graph Settings

- Number of blocks: 10

- Size of blocks: a random variable following the discrete uniform distribution over the set $\{5, 6, 7, ..., 20\}$.
- Intra-block probability of an edge: 0.5
- Inter-block probability of an edge: 0.01
- Probability of positive infection status: 0.2
- Maximum number of days prior to T for infection time: 30
- Decay Period: 14

Probability of Exposure for 1st and 2nd vs 1st Order Neighbors Experiment: Each patient has a probability of exposure defined as $\phi_p = \Phi(\{\mathbf{c}(\mathbf{x}_u, \mathbf{t}_u) : u \in N_m(p) \wedge u \neq p\})$, where:

- $\Phi = \text{mean}$
- $\mathbf{c}(\mathbf{x}_u, \mathbf{t}_u) = \mathbb{1}_{\mathbf{x}_u} \cdot \left(1 - min(T - \mathbf{t}_u, d)\frac{1}{d}\right)$
- d = Decay Period
- $N_m(p) =$ First and second order neighborhood of p

Probability of Exposure for Mean vs Max, Decay Period 14 vs 3 Days Experiments: Each patient has a probability of exposure defined as $\phi_p = \Phi(\{\mathbf{c}(\mathbf{x}_u, \mathbf{t}_u) : u \in N_m(p) \wedge u \neq p\})$, where:

- $\Phi = \text{mean}$
- $\mathbf{c}(\mathbf{x}_u, \mathbf{t}_u) = \mathbb{1}_{\mathbf{x}_u} \cdot \left(1 - min(T - \mathbf{t}_u, d)\frac{1}{d}\right)$
- d = Decay Period
- $N_m(p) =$ First order neighborhood of p

Probability of Exposure for Exponential vs Linear Decay Experiment: Each patient has a probability of exposure defined as $\phi_p = \Phi(\{\mathbf{c}(\mathbf{x}_u, \mathbf{t}_u) : u \in N_m(p) \wedge u \neq p\})$, where:

- $\Phi = \text{mean}$

- $\mathbf{c}(\mathbf{x}_u, \mathbf{t}_u) = \mathbb{1}_{\mathbf{x}_u} \cdot max\left(0, 1 - \frac{log(T - \mathbf{t}_u + 1)}{log(d)}\right)$
- d = Decay Period
- $N_m(p)$ = First order neighborhood of p

Probability of Exposure for Symptomatic vs All Known Positive Neighbors Experiment: Each patient has a probability of exposure defined as $\phi_p = \Phi(\{\mathbf{c}(\mathbf{x}_u, \mathbf{t}_u) : u \in N_m(p) \wedge u \neq p\})$, where:

- Φ = mean
- $\mathbf{c}(\mathbf{x}_u, \mathbf{t}_u) = \mathbb{1}_{\mathbf{x}_u \wedge \mathbf{s}_u} \cdot \left(1 - min(T - \mathbf{t}_u, d)\frac{1}{d}\right)$
- d = Decay Period
- $\mathbf{s}_u$ = Indicator for Symptomatic Cases
- $\mathbf{s}_u \sim Binomial(p = 0.25)$
- $N_m(p)$ = First order neighborhood of p

Bibliography

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

[2] H. Harutyunyan, H. Khachatrian, D. C. Kale, and A. Galstyan, "Multitask learning and benchmarking with clinical time series data," *arXiv preprint arXiv:1703.07771*, 2017.

[3] M. S. Stefan, M.-S. Shieh, P. S. Pekow, M. B. Rothberg, J. S. Steingrub, T. Lagu, and P. K. Lindenauer, "Epidemiology and outcomes of acute respiratory failure in the united states, 2001 to 2009: A national survey," *Journal of hospital medicine*, vol. 8, no. 2, pp. 76–82, 2013.

[4] D. F. Gaieski and M. Mikkelsen, "Definition, classification, etiology, and pathophysiology of shock in adults," *UpToDate, Waltham, MA. Accesed*, vol. 8, p. 17, 2016.

[5] T. Chopra and B. Navalkele, *Clostridium difficile in the hospital: Infection prevention considerations*, Jan. 2019. [Online]. Available: `https://www.infectiousdiseaseadvisor.com/home/decision-support-in-medicine/hospital-infection-control/clostridium-difficile-in-the-hospital-infection-prevention-considerations/`.

[6] T. Claro, S. Daniels, and H. Humphreys, "Detecting clostridium difficile spores from inanimate surfaces of the hospital environment: Which method is best?" *Journal of Clinical Microbiology*, vol. 52, no. 9, pp. 3426–3428, 2014. DOI: `10.1128/jcm.01011-14`.

[7] P. J. Olver and P. J. Olver, *Classical invariant theory*, 44. Cambridge University Press, 1999.

[8] J. E. Wipf, B. A. Lipsky, J. V. Hirschmann, E. J. Boyko, J. Takasugi, R. L. Peugeot, and C. L. Davis, "Diagnosing pneumonia by physical examination: Relevant or relic?" *Archives of Internal Medicine*, vol. 159, no. 10, pp. 1082–1087, 1999.

[9] *Arrhythmia*. [Online]. Available: `https://www.nhlbi.nih.gov/health-topics/arrhythmia`.

[10] J. Wiens and J. V. Guttag, "Active learning applied to patient-adaptive heartbeat classification," in *Advances in Neural Information Processing Systems (NIPS)*, 2010, pp. 2442–2450.

[11] N. Razavian and D. Sontag, "Temporal convolutional neural networks for diagnosis from lab tests," *arXiv preprint arXiv:1511.07938*, 2015.

[12] N. Razavian, J. Marcus, and D. Sontag, "Multi-task prediction of disease onsets from longitudinal laboratory tests," in *Machine Learning for Healthcare Conference (MLHC)*, 2016, pp. 73–100.

[13] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," *arXiv preprint arXiv:1603.06995*, 2016.

[14] J. Oh, J. Wang, and J. Wiens, "Learning to exploit invariances in clinical time-series data using sequence transformer networks," in *Proceedings of the Machine Learning for Healthcare Conference, MLHC 2018*, 2018.

[15] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang, "Domain adaptation under target and conditional shift," in *International Conference on Machine Learning*, 2013, pp. 819–827.

[16] J. Oh, J. Wang, S. Tang, and J. Wiens, "Relaxed weight sharing: Effectively modeling time-varying relationships in clinical time-series," in *In Submission to Machine Learning for Healthcare Conference, MLHC 2019*, 2019.

[17] J. Wiens, J. Guttag, and E. Horvitz, "Patient risk stratification with time-varying parameters: A multitask learning approach," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2797–2819, 2016.

[18] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[19] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020. DOI: `10.1109/TKDE.2020.2981333`.

[20] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[21] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations*, 2018, accepted as poster. [Online]. Available: `https://openreview.net/forum?id=rJXMpikCZ`.

[22] 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, p. 160 035, 2016.

[23] S. Lohit, Q. Wang, and P. Turaga, "Temporal transformer networks: Joint learning of invariant and discriminative time warping," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 426–12 435.

[24] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *International Conference on Learning Representations*, 2017.

[25] J. Oh and J. Wiens, "A data-driven approach to estimating infectious disease transmission from graphs: A case of class imbalance driven low homophily," in *In Submission to ACM Conference on Health, Inference, and Learning, ACM-CHIL 2021*, 2021.

[26] R. Evans, “Electronic health records: Then, now, and in the future,” *Yearbook of medical informatics*, vol. 25, no. S 01, S48–S61, 2016.

[27] K. S. Little, *The electronic medical records (emr) mandate*, Jan. 2013. [Online]. Available: `https://www.healthcarelaw-blog.com/the-electronic-medical-records-emr-mandate/`.

[28] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel, “Learning to diagnose with lstm recurrent neural networks,” *International Conference on Learning Representations (ICLR)*, 2016.

[29] D. Agniel, I. S. Kohane, and G. M. Weber, “Biases in electronic health record data due to processes within the healthcare system: Retrospective observational study,” *Bmj*, vol. 361, 2018.

[30] P. Yadav, M. Steinbach, V. Kumar, and G. Simon, “Mining electronic health records (ehrs) a survey,” *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–40, 2018.

[31] J. Wiens, S. Saria, M. Sendak, M. Ghassemi, V. X. Liu, F. Doshi-Velez, K. Jung, K. Heller, D. Kale, M. Saeed, *et al.*, “Do no harm: A roadmap for responsible machine learning for health care,” *Nature medicine*, vol. 25, no. 9, pp. 1337–1340, 2019.

[32] G. U. Meduri, R. E. Turner, N. Abou-Shala, R. Wunderink, and E. Tolley, “Noninvasive positive pressure ventilation via face mask: First-line intervention in patients with acute hypercapnic and hypoxemic respiratory failure,” *Chest*, vol. 109, no. 1, pp. 179–193, 1996.

[33] F. C. Lessa, Y. Mu, W. M. Bamberg, Z. G. Beldavs, G. K. Dumyati, J. R. Dunn, M. M. Farley, S. M. Holzbauer, J. I. Meek, E. C. Phipps, *et al.*, “Burden of clostridium difficile infection in the united states,” *New England Journal of Medicine*, vol. 372, no. 9, pp. 825–834, 2015.

[34] E. Dubberke, “Strategies for prevention of clostridium difficile infection,” *Journal of Hospital Medicine*, vol. 7, no. S3, Mar. 2012.

[35] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[36] M. Fiterau, S. Bhooshan, J. Fries, C. Bournhonesque, J. Hicks, E. Halilaj, C. Ré, and S. Delp, "Shortfuse: Biomedical time series representations in the presence of structured information," *Machine Learning for Healthcare Conference (MLHC)*, 2017.

[37] C. Olah, *Understanding lstm networks*, Aug. 2015. [Online]. Available: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[38] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of cnn and rnn for natural language processing," *arXiv preprint arXiv:1702.01923*, 2017.

[39] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[40] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations (ICLR)*, 2017.

[41] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific Reports*, vol. 8, no. 1, p. 6085, 2018.

[42] J. Oh, M. Makar, C. Fusco, R. McCaffrey, K. Rao, E. E. Ryan, L. Washer, L. R. West, V. B. Young, J. Guttag, *et al.*, "A generalizable, data-driven approach to predict daily risk of clostridium difficile infection at two large academic health centers," *infection control & hospital epidemiology*, vol. 39, no. 4, pp. 425–433, 2018.

[43] Y. Liu, Z. Syed, B. M. Scirica, D. A. Morrow, J. V. Guttag, and C. M. Stultz, "ECG morphological variability in beat space for risk stratification after acute coronary syndrome," *Journal of the American Heart Association (JAHA)*, vol. 3, no. 3, e000981, 2014.

[44] J. J. G. Ortiz, C. P. Phoo, and J. Wiens, "Heart sound classification based on temporal alignment techniques," in *Computing in Cardiology Conference (CinC), 2016*, IEEE, 2016, pp. 589–592.

[45] G. E. Batista, X. Wang, and E. J. Keogh, "A complexity-invariant distance measure for time series," in *Proceedings of the 2011 SIAM International Conference on Data Mining (SDM)*, SIAM, 2011, pp. 699–710.

[46] Z. Wang and T. Oates, "Imaging time-series to improve classification and imputation," *Proceedings of the 24th International Join Conference on Artificial Intelligence (IJCAI)*, 2015.

[47] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *International Conference on Machine Learning (ICML)*, 2017.

[48] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," *International Conference on Machine Learning (ICML)*, 2017.

[49] H. Suresh, N. Hunt, A. Johnson, L. A. Celi, P. Szolovits, and M. Ghassemi, "Clinical intervention prediction and understanding with deep neural networks," in *Machine Learning for Healthcare Conference (MLHC)*, 2017, pp. 322–337.

[50] P. Bashivan, I. Rish, M. Yeasin, and N. Codella, "Learning representations from EEG with deep recurrent-convolutional neural networks," *International Conference on Learning Representations (ICLR)*, 2016.

[51] T. T. Um, F. M. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić, "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks," in *Proceedings of the 19th ACM International Conference on Multimodal Interaction (ICMI)*, ACM, 2017, pp. 216–220.

[52] F. Wang, N. Lee, J. Hu, J. Sun, and S. Ebadollahi, "Towards heterogeneous temporal clinical event pattern discovery: A convolutional approach," in *Proceedings*

of the 18th International Conference on Knowledge Discovery and Data Mining (KDD), ACM, 2012, pp. 453–461.

[53] G. Forestier, F. Petitjean, H. A. Dau, G. I. Webb, and E. Keogh, "Generating synthetic time series to augment sparse datasets," in *IEEE International Conference on Data Mining (ICDM)*, IEEE, 2017, pp. 865–870.

[54] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 2017–2025.

[55] A. Rajkomar, E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun, *et al.*, "Scalable and accurate deep learning with electronic health records," *NPJ Digital Medicine*, vol. 1, no. 1, p. 18, 2018.

[56] S. J. Reddi, B. Poczos, and A. Smola, "Doubly robust covariate shift correction," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[57] M. Sugiyama, M. Krauledat, and K.-R. MÃžller, "Covariate shift adaptation by importance weighted cross validation," *Journal of Machine Learning Research*, vol. 8, no. May, pp. 985–1005, 2007.

[58] M. Gong, K. Zhang, T. Liu, D. Tao, C. Glymour, and B. Schölkopf, "Domain adaptation with conditional transferable components," in *International conference on machine learning*, 2016, pp. 2839–2848.

[59] H. Daumé III, "Frustratingly easy domain adaptation," *Proc. 45th Ann. Meeting of the Assoc. Computational Linguistics*, 2007.

[60] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[61] Y. Ding, J. Yu, and J. Jiang, "Recurrent neural networks with auxiliary labels for cross-domain opinion target extraction," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[62] J. J. Thiagarajan, D. Rajan, and P. Sattigeri, "Can deep clinical models handle real-world domain shifts?" *arXiv preprint arXiv:1809.07806*, 2018.

[63] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: An astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806–813.

[64] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 513–520.

[65] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He, "Supervised representation learning: Transfer learning with deep autoencoders," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[66] D. M. dos Reis, P. Flach, S. Matwin, and G. Batista, "Fast unsupervised online drift detection using incremental kolmogorov-smirnov test," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 1545–1554.

[67] D. J. Soemers, T. Brys, K. Driessens, M. H. Winands, and A. Nowé, "Adapting to concept drift in credit card transaction data streams using contextual bandits and decision trees," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[68] C. A. Bellera, G. MacGrogan, M. Debled, C. T. de Lara, V. Brouste, and S. Mathoulin-Pélissier, "Variables with time-varying effects and the Cox model: Some statistical concepts illustrated with a prognostic factor study in breast cancer," *BMC medical research methodology*, vol. 10, no. 1, p. 20, 2010.

[69] F. W. Dekker, R. De Mutsert, P. C. Van Dijk, C. Zoccali, and K. J. Jager, "Survival analysis: Time-dependent effects and time-varying risk factors," *Kidney international*, vol. 74, no. 8, pp. 994–997, 2008.

[70] H. Park and C. D. Yoo, "Early improving recurrent elastic highway network," *arXiv preprint arXiv:1708.04116*, 2017.

[71] J. Kohlmorgen, K.-R. Müller, and K. Pawelzik, “Analysis of drifting dynamics with neural network hidden markov models,” in *Advances in Neural Information Processing Systems*, 1998, pp. 735–741.

[72] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi, “Modeling task relationships in multi-task learning with multi-gate mixture-of-experts,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, 2018, pp. 1930–1939.

[73] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, “Skipnet: Learning dynamic routing in convolutional networks,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 409–424.

[74] D. Eigen, M. Ranzato, and I. Sutskever, “Learning factored representations in a deep mixture of experts,” *International Conference on Learning Representations workshop*, 2014.

[75] T. Tan, Y. Qian, and K. Yu, “Cluster adaptive training for deep neural network based acoustic model,” *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 24, no. 3, pp. 459–468, 2016.

[76] P. Savarese and M. Maire, “Learning implicitly recurrent CNNs through parameter sharing,” in *International Conference on Learning Representations*, 2019. [Online]. Available: `https://openreview.net/forum?id=rJgYxn09Fm`.

[77] M. Arjovsky, A. Shah, and Y. Bengio, “Unitary evolution recurrent neural networks,” in *International Conference on Machine Learning*, 2016, pp. 1120–1128.

[78] T. Avni, A. Lador, S. Lev, L. Leibovici, M. Paul, and A. Grossman, “Vasopressors for the treatment of septic shock: Systematic review and meta-analysis,” *PloS one*, vol. 10, no. 8, e0129305, 2015.

[79] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[80] D. Kingma and J. Ba, "Adam: A method for stochastic optimization (2014)," *International Conference on Learning Representations*, vol. 15, 2015.

[81] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[82] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, *et al.*, *Grad-cam: Visual explanations from deep networks via gradient-based localization., in 'iccv'*, 2016.

[83] A. Graves, "Supervised sequence labelling," in *Supervised sequence labelling with recurrent neural networks*, Springer, 2012, pp. 5–13.

[84] A. S. Ross, M. C. Hughes, and F. Doshi-Velez, "Right for the right reasons: Training differentiable models by constraining their explanations," *International Joint Conferences on Artificial Intelligence Organization*, 2017.

[85] A. Ghorbani, A. Abid, and J. Zou, "Interpretation of neural networks is fragile," *AAAI*, 2019.

[86] A. Fisher, C. Rudin, and F. Dominici, "All models are wrong but many are useful: Variable importance for black-box, proprietary, or misspecified prediction models, using model class reliance," *arXiv preprint arXiv:1801.01489*, 2018.

[87] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[88] J. Oh and J. Wiens, "Graph attention neural networks for estimating infectious disease transmission: Addressing class imbalance driven low homophily," *The Conference on Health, Inference, and Learning (ACM CHIL) Workshop*, 2021.

[89] J. Bintz, S. Lenhart, and C. Lanzas, "Antimicrobial stewardship and environmental decontamination for the control of clostridium difficile transmission in healthcare settings," *Bulletin of mathematical biology*, vol. 79, no. 1, pp. 36–62, 2017.

[90] N. Blanco, L. M. O'Hara, and A. D. Harris, "Transmission pathways of multidrug-resistant organisms in the hospital setting: A scoping review," *Infection Control & Hospital Epidemiology*, vol. 40, no. 4, pp. 447–456, 2019.

[91] K. R. Chng, C. Li, D. Bertrand, A. H. Q. Ng, J. S. Kwah, H. M. Low, C. Tong, M. Natrajan, M. H. Zhang, L. Xu, *et al.*, "Cartography of opportunistic pathogens and antibiotic resistance genes in a tertiary hospital environment," *Nature Medicine*, pp. 1–11, 2020.

[92] D. E. Freedberg, H. Salmasian, B. Cohen, J. A. Abrams, and E. L. Larson, "Receipt of antibiotics in hospitalized patients and risk for clostridium difficile infection in subsequent patients who occupy the same bed," *JAMA internal medicine*, vol. 176, no. 12, pp. 1801–1808, 2016.

[93] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[94] A. Kapoor, X. Ben, L. Liu, B. Perozzi, M. Barnes, M. Blais, and S. O'Banion, "Examining covid-19 forecasting using spatio-temporal graph neural networks," *arXiv preprint arXiv:2007.03113*, 2020.

[95] V. La Gatta, V. Moscato, M. Postiglione, and G. Sperli, "An epidemiological neural network exploiting dynamic graph structured data applied to the covid-19 outbreak," *IEEE Transactions on Big Data*, vol. 7, no. 1, pp. 45–55, 2020.

[96] S.-H. Wang, V. V. Govindaraj, J. M. Górriz, X. Zhang, and Y.-D. Zhang, "Covid-19 classification by fgcnet with deep feature fusion from graph convolutional network and convolutional neural network," *Information Fusion*, vol. 67, pp. 208–229, 2020.

[97] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[98] B. T. Fukunaga, W. K. Sumida, D. A. Taira, J. W. Davis, and T. B. Seto, “Hospital-acquired methicillin-resistant staphylococcus aureus bacteremia related to medicare antibiotic prescriptions: A state-level analysis,” *Hawai'i Journal of Medicine & Public Health*, vol. 75, no. 10, p. 303, 2016.

[99] D. W. Eyre, D. Griffiths, A. Vaughan, T. Golubchik, M. Acharya, L. O'Connor, D. W. Crook, A. S. Walker, and T. E. Peto, “Asymptomatic clostridium difficile colonisation and onward transmission,” *PloS one*, vol. 8, no. 11, e78445, 2013.

[100] E. van Kleef, J. V. Robotham, M. Jit, S. R. Deeny, and W. J. Edmunds, “Modelling the transmission of healthcare associated infections: A systematic review,” *BMC infectious diseases*, vol. 13, no. 1, p. 294, 2013.

[101] Z. Xing, B. Nicholson, M. Jimenez, T. Veldman, L. Hudson, J. Lucas, D. Dunson, A. K. Zaas, C. W. Woods, G. S. Ginsburg, *et al.*, “Bayesian modeling of temporal properties of infectious disease in a college student population,” *Journal of Applied Statistics*, vol. 41, no. 6, pp. 1358–1382, 2014.

[102] S. Chae, S. Kwon, and D. Lee, “Predicting infectious disease using deep learning and big data,” *International Journal of Environmental Research and Public Health*, vol. 15, no. 8, p. 1596, Jul. 2018, ISSN: 1660-4601. DOI: 10.3390/ijerph15081596. [Online]. Available: http://dx.doi.org/10.3390/ijerph15081596.

[103] A. O. Ajao, A. D. Harris, M.-C. Roghmann, J. K. Johnson, M. Zhan, J. C. McGregor, and J. P. Furuno, “Systematic review of measurement and adjustment for colonization pressure in studies of methicillin-resistant staphylococcus aureus, vancomycin-resistant enterococci, and clostridium difficile acquisition,” *Infection Control & Hospital Epidemiology*, vol. 32, no. 5, pp. 481–489, 2011.

[104] M. J. Bonten, S. Slaughter, A. W. Ambergen, M. K. Hayden, J. van Voorhis, C. Nathan, and R. A. Weinstein, “The role of colonization pressure in the spread of vancomycin-resistant enterococci: An important infection control variable,” *Archives of internal medicine*, vol. 158, no. 10, pp. 1127–1132, 1998.

[105] S. J. Lawrence, L. A. Puzniak, B. N. Shadel, K. N. Gillespie, M. H. Kollef, and L. M. Mundy, “Clostridium difficile in the intensive care unit: Epidemiology, costs,

and colonization pressure," *Infection control and hospital epidemiology*, vol. 28, no. 2, pp. 123–130, 2007. DOI: 10.1086/511793.

[106] L. Puzniak, J. Mayfield, T. Leet, M. Kollef, and L. Mundy, "Acquisition of vancomycin-resistant enterococci during scheduled antimicrobial rotation in an intensive care unit," *Clinical Infectious Diseases*, vol. 33, no. 2, pp. 151–157, 2001.

[107] J. M. Mylotte, S. Russell, B. Sackett, M. Vallone, and M. Antalek, "Surveillance for clostridium difficile infection in nursing homes," *Journal of the American Geriatrics Society*, vol. 61, no. 1, pp. 122–125, 2013.

[108] M. Makar, J. Guttag, and J. Wiens, "Learning the probability of activation in the presence of latent spreaders," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[109] S. S. Magill, E. O'Leary, S. J. Janelle, D. L. Thompson, G. Dumyati, J. Nadle, L. E. Wilson, M. A. Kainer, R. Lynfield, S. Greissman, *et al.*, "Changes in prevalence of health care–associated infections in us hospitals," *New England Journal of Medicine*, vol. 379, no. 18, pp. 1732–1744, 2018.

[110] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" In *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=ryGs6iA5Km.

[111] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 9240–9251.

[112] K. Agarwal, T. Eftimov, R. Addanki, S. Choudhury, S. Tamang, and R. Rallo, "Snomed2vec: Random walk and poincar\'e embeddings of a clinical knowledge base for healthcare analytics," *arXiv preprint arXiv:1907.08650*, 2019.

[113] Y. Dong, N. V. Chawla, and A. Swami, "Metapath2vec: Scalable representation learning for heterogeneous networks," in *KDD '17*, ACM, 2017, pp. 135–144.

[114] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2016, pp. 855–864.

[115] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *Advances in Neural Information Processing Systems*, 2020.

[116] M. Shi, Y. Tang, X. Zhu, D. Wilson, and J. Liu, "Multi-class imbalanced graph convolutional network learning," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, 2020.

[117] L. Zhao and L. Akoglu, "Pairnorm: Tackling oversmoothing in gnns," in *International Conference on Learning Representations*, 2020. [Online]. Available: `https://openreview.net/forum?id=rkecl1rtwB`.

[118] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, no. 1, pp. 1–54, 2019.

[119] K. Roberts, C. F. Smith, A. M. Snelling, K. G. Kerr, K. R. Banfield, P. A. Sleigh, and C. B. Beggs, "Aerial dissemination of clostridium difficilespores," *BMC infectious diseases*, vol. 8, no. 1, p. 7, 2008.

[120] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[121] K. E. Henry, D. N. Hager, P. J. Pronovost, and S. Saria, "A targeted real-time early warning score (trewscore) for septic shock," *Science translational medicine*, vol. 7, no. 299, 299ra122–299ra122, 2015.

[122] G. Philipp, D. Song, and J. G. Carbonell, "Gradients explode-deep networks are shallow-resnet explained," 2018.

[123] Z. Obermeyer, B. Powers, C. Vogeli, and S. Mullainathan, “Dissecting racial bias in an algorithm used to manage the health of populations,” *Science*, vol. 366, no. 6464, pp. 447–453, 2019.

[124] S. Jabbour, D. Fouhey, E. Kazerooni, M. W. Sjoding, and J. Wiens, “Deep learning applied to chest x-rays: Exploiting and preventing shortcuts,” in *Machine Learning for Healthcare Conference*, PMLR, 2020, pp. 750–782.

[125] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[126] J. Wang, J. Wiens, and S. Lundberg, “Shapley flow: A graph-based approach to interpreting model predictions,” 2021.