Learning Deep Patient Representations for the TeleICU

by

Ini Oguntola

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2019

Author .....
Department of Electrical Engineering and Computer Science
August 23, 2019

Certified by .....
Amar Gupta
Research Scientist
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Learning Deep Patient Representations for the TeleICU

by

Ini Oguntola

Submitted to the Department of Electrical Engineering and Computer Science
on August 23, 2019, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis presents a method of extracting deep robust representations of teleICU clinical data using Transformer networks, inspired by recent machine learning literature in language modeling. The utility of these representations is evaluated in various prediction outcome tasks, in which they were able to outperform linear and neural baselines. Also examined are the probability distributions of various patient characteristics across the learned patient representation space; where corresponding high-level spatial structure suggests potential for use as a similarity metric or in combination with other patient similarity metrics. Finally, the code for the models developed is publicly provided as a starting point for further research.

Thesis Supervisor: Amar Gupta
Title: Research Scientist

Acknowledgments

This is the acknowledgements section. You should replace this with your own acknowledgements.

Contents

Contents **7**

List of Figures **11**

List of Tables **13**

1 Introduction **15**

1.1 Telemedicine and the TeleICU 15

1.2 Goals 16

2 Deep Learning Overview **17**

2.1 Convolutional Neural Networks 17

2.2 Recurrent Neural Networks 18

2.3 Autoencoders 18

2.4 Attention 19

3 Deep Learning Techniques for EHR Analysis **21**

3.1 Outcome Prediction 22

3.1.1 Interpretability 23

3.2 Representation Learning 24

3.2.1 Unsupervised Approaches 24

3.2.2 Supervised Approaches 25

3.2.3 Patient Similarity 25

4 Vision **27**

5 Learning Deep Representations for Clinical Time-Series Data 29

5.1 Approach . . . 29

5.2 Data . . . 30

5.2.1 Dataset . . . 30

5.2.2 Data Preprocessing . . . 30

5.3 Models . . . 31

5.3.1 General Architecture . . . 32

5.3.2 Multitask Learning . . . 32

5.3.3 Autoregressive Generative Sequence Modeling . . . 34

5.4 Training . . . 36

6 Outcome Prediction with Learned Representations 39

6.1 Supervised Fine-Tuning . . . 39

6.2 Experiments . . . 40

6.2.1 Data . . . 40

6.2.2 Outcome Prediction Tasks . . . 40

6.2.3 Baselines . . . 41

6.3 Results . . . 41

6.4 Discussion . . . 43

6.4.1 Calibration . . . 44

6.4.2 K-Fold Cross Validation . . . 46

6.4.3 Finetuning Convergence . . . 51

7 Patient Similarity with Learned Representations 53

7.1 Visualizations . . . 53

7.1.1 Interpreting the Visualizations . . . 54

7.2 Principal Component Analysis . . . 55

7.3 T-Distributed Stochastic Neighbouring Entities . . . 56

7.4 Results . . . 56

7.5 Visualizations of Sepsis Patients . . . 57

8 Caduceus **81**

8.1 PatientFinder 81

8.2 Methods: 82

9 Conclusions **85**

9.1 Contributions 85

9.1.1 Patient Outcome Prediction 85

9.1.2 Patient Similarity Analysis 86

9.1.3 Open-Sourced Code 86

9.2 Future Work 86

10 References **89**

List of Figures

6-1 ROC Curves for Mortality Prediction 44
6-2 ROC Curves for Decompensation Prediction 45
6-3 Mortality Calibration 47
6-4 Decompensation Calibration 48

7-1 Mortality Visualization with PCA 58
7-2 Mortality Visualization with t-SNE 59
7-3 Decompensation Visualization with PCA 60
7-4 Decompensation Visualization with t-SNE 61
7-5 Length of Stay Visualization with PCA (< 24 hours) 62
7-6 Length of Stay Visualization with PCA (1-2 days) 63
7-7 Length of Stay Visualization with PCA (2-3 days) 64
7-8 Length of Stay Visualization with PCA (≥ 3 days) 65
7-9 Length of Stay Visualization with t-SNE (< 24 hours) 66
7-10 Length of Stay Visualization with t-SNE (1-2 days) 67
7-11 Length of Stay Visualization with t-SNE (2-3 days) 68
7-12 Length of Stay Visualization with t-SNE (≥ 3 days) 69
7-13 Phenotype Visualization with PCA (Acute and unspecified renal failure) 70
7-14 Phenotype Visualization with PCA (Chronic obstructive pulmonary disease and bronchiectasis) 71
7-15 Phenotype Visualization with PCA (Essential hypertension) 72
7-16 Phenotype Visualization with PCA (Pneumonia) 73

7-17 Phenotype Visualization with PCA (Respiratory failure) 74
7-18 Phenotype Visualization with t-SNE (Acute and unspecified renal failure) 75
7-19 Phenotype Visualization with t-SNE (Chronic obstructive pulmonary disease and bronchiectasis) 76
7-20 Phenotype Visualization with t-SNE (Essential hypertension) 77
7-21 Phenotype Visualization with t-SNE (Pneumonia) 78
7-22 Phenotype Visualization with t-SNE (Respiratory failure) 79
7-23 Sepsis Visualization with PCA 80

List of Tables

5.1 Selected Clinical Variables and Impute Values 31
5.2 Selected ICU Phenotypes 35

6.1 Outcome Prediction Results 42
6.2 K-Fold Cross Validation Results (Autoregressive Transformer) 49

Chapter 1

Introduction

There is more data in healthcare than ever before. Recent years have seen widespread integration of information technology in the healthcare industry, which has resulted in an unprecedented amount of digitally-accessible clinical data; one 2018 report estimates the size of the healthcare datasphere at 1.2 zettabytes (10^{21} bytes, over a trillion gigabytes), and projects it to grow at a rate of 36% annually through 2025 [1]. Increased access to such data suggests growing potential for the use of data analytics in healthcare to support clinical decision making and improve patient outcomes.

Critical care is one area where data analytics can be particularly useful. Patients in ICUs and in critical care are high-risk, and thus are closely monitored and produce more EHR data than non-ICU patients [2]. Data analytics has tremendous potential to help improve critical care for patients by helping to manage the sheer volume of data produced in ICUs and through insights gleaned from this data.

1.1 Telemedicine and the TeleICU

In recent years many hospitals have adapted teleICU models, a model of intensive care in which patients are monitored by a remote team of clinicians who guide bedside providers via planned consults and reacting to a continuous stream of data and alerts

[3].

TeleICUs have been particularly attractive due to a recent shortage of intensivists that is predicted to only worsen in the next decade [4], contributing to their growing adoption. However, teleICU systems do not necessarily have to be implemented as a replacement for traditional ICUs; they can also augment traditional ICUs and help better optimize how medical personnel attend to patients [4].

With the recent releases of the MIMIC-III [5] and Philips eICU datasets [3], much of recent research in applying analytics based in deep learning approaches to patient data has been focused on the ICU [2,6]. In the teleICU, however, few have attempted to use the power of modern deep learning techniques for data analytics. In this thesis we propose systems that analyze and utilize clinical data from the past through deep learning approaches and representations to aid clinical decision making in teleICU environments.

1.2 Goals

The central question of this thesis is: is it possible to use modern deep learning techniques and a way of leveraging historical patient data to improve clinical care in teleICU environments? Specifically, the goals of this work are as follows:

1. To create a method for extracting clinically meaningful vector deep representations of teleICU data for individual patients
2. To use these representations to predict patient outcomes
3. To use these representations to derive a patient similarity metric which allows clinicians to identify similar patients for any individual patient
4. To build a framework that leverages derived metrics and past patient data to facilitate clinical operations in the teleICU

Chapter 2

Deep Learning Overview

Deep learning generally refers to a collection of models known as *neural networks* that use multiple layers to successively extract representations of input data that are better suited for a given task. The prototypical neural network is the *multilayer perceptron* (MLP), also called a *feedforward network* (FFN), which transforms its inputs via composition and nonlinear activations from the neurons in each layer [7]. Deep models in general have shown to be especially effective when trained on very large amounts of data, able to recognize complex and even ill-defined patterns in the input.

There are many variations in neural network architecture developed for processing different types of data and for different domains. Some of the standard neural network architectures are described throughout the rest of this section. These architectures are often combined and/or used as subnetworks in larger networks to create more complex architectures.

2.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a class of neural networks developed for handling grid-like data, most commonly used for processing images in the field of

computer vision. A convolutional layer is similar to a hidden layer in a feedforward network, but uses convolution instead of ordinary matrix multiplication. Convolutional layers usually have three stages: the first layer convolves the input with a learned convolution kernel to create an activation map, the second passes the activation map through some non-linear activation function, and the final stage modifies each value in the input data based on locally nearby values. CNNs are characterized by containing at least one of these convolutional layers [7].

2.2 Recurrent Neural Networks

Deep learning techniques are also effective in dealing with sequential or temporal data. In particular, a class of deep models called *recurrent neural networks* (RNNs) have shown impressive results when dealing with sequences and/or time-series data. At a basic level RNNs work by processing the input sequentially, and at each timestep using the corresponding input along with learned parameters to update some hidden internal state. The most commonly used recurrent architecture is the Long-Short-Term-Memory network (LSTM) [8].

2.3 Autoencoders

Autoencoders (AEs) are a type of neural network architecture that use an encoder decoder structure to learn (usually compressed) representations of the input data. The encoder takes in the input and encodes it, usually as a lower-dimensional vector. The decoder takes in the encoded input and attempts to reconstruct the original input. Autoencoders are trained by learning to minimize some reconstruction error or loss. They can be thought of as a deep form of dimensionality reduction; like techniques such as singular value decomposition (SVD) or principle component analysis (PCA) they obtain more efficient representations of data.

If the dimension of the hidden code is larger than that of the input, the risk is that the autoencoder simply learns the identity function. To combat this, a variant called *denoising autoencoders* (DAEs) corrupt the input with some noise before feeding it into the encoder, but the decoder tries to reconstruct the hidden code to match the original input, not the corrupted input; this causes the model to learn to undo the noise and prevents it from simply copying the input. Other common variants include variational autoencoders (VAEs) and sparse autoencoders (SAEs) [7].

2.4 Attention

Attention mechanisms in neural networks are a recent development in deep processing sequential data. Inspired by the way the human brain filters visual information to retain useful details, attention mechanisms learn to select and prioritize the most important information from a stream of data. Attention mechanisms are also interpretable, as they allow us to determine which parts of the input data were most important or influential in generating the output, which can be quite useful for many deep learning applications [9, 10].

Chapter 3

Deep Learning Techniques for EHR Analysis

Until recently most approaches to EHR analysis and general data analysis in healthcare were based on traditional machine learning techniques; however, the use of deep learning techniques is becoming increasingly widespread in healthcare applications. Most deep learning approaches for EHR analysis fall into one of the following categories: *information extraction*, *clinical data de-identification*, *computational phenotyping*, *representation learning*, and *outcome prediction* [11].

Information extraction approaches aim to obtain relevant information from textual clinical notes, and clinical data de-identification approaches aim to automate the process of removing sensitive patient information, including names, dates, and geographic locations. Both of these heavily utilize deep learning techniques and architectures from natural language processing (NLP). Computational phenotyping uses unsupervised learning approaches to refine or to derive new data-driven descriptions of diseases and diagnoses [11].

Outcome prediction approaches utilize deep learning techniques to predict specific patient outcomes or future events, such as mortality, length of stay, readmission, or diagnosis of a particular disease. Representation learning approaches to analysis focus

on transforming discrete medical codes from EHR data into vectors in a continuous space to facilitate analysis and for use in other predictive tasks.

As the focus of this thesis is outcome prediction and representation learning in the teleICU, here we also focus on existing deep approaches to outcome prediction and representation learning with EHR data, which are described in more detail throughout the rest of this section.

3.1 Outcome Prediction

Deep approaches to outcome prediction vary in two main ways: in the architecture of the model being used and in the outcome metric being predicted. In the context of critical care the most common metrics are mortality and length of stay [12]; prediction of readmission or prediction of diagnosis are also common as well [2]. An important starting question to address is: does deep learning provide any advantage over traditional methods for clinical prediction of risk or outcomes?

Purushotham et. al. performed a detailed comparison on the MIMIC-III dataset for predicting ICU mortality and length of stay [12]. Comparison was done across a suite of traditional prediction methods including logistic regression, regression trees, additive models, shallow neural networks, and random forests. Three deep learning models were tested. The first was a feed-forward network (FFN) with static ICU data as input, and the second was a type of recurrent neural network called the Gated Recurrent Unit (GRU) that took in temporal ICU data as input. The final deep model was a multimodal model that took in both static and temporal input data, used FFN and GRU sub-networks to process the static and temporal data respectively, and then used them to learn shared latent representations to predict both ICU mortality and length of stay. The study was able to show that the deep models consistently outperform all the other approaches, especially when a large number of raw clinical time series data was used as input.

Rajkomar et. al. created a similar pipeline for outcome prediction from EHR data represented in the open-source Fast Healthcare Interoperability Resources (FHIR) format [13]. They tested three model architectures: an LSTM, an attention-based time-aware neural network (TANN), and a feedforward network with boosted time-series embeddings. These models were trained to separately predict mortality, length of stay, unplanned readmission, and diagnosis; in all cases they outperformed predictive models traditionally used by clinicians.

Other less commonly used deep architectures include CNNs, autoencoders, RBMs, and DBNs [14–17], all of which perform better than traditional non-deep techniques.

3.1.1 Interpretability

One of the issues that often arises in deep learning applications, including in healthcare applications, is the lack of interpretability for deep models; acting as essentially black boxes, it is often difficult to explain their behavior, no matter how well they perform. Che et. al. tackle this problem through what is known as mimic or imitation learning, where one model is trained to reproduce the output of another [18]. They first train a multimodal deep models consisting of a FFN and GRU to predict ICU mortality and Ventilator Free Days (VFD) given both static and temporal input data. They then model the probability distribution predicted by the deep models with gradient boosting trees (GBT), achieving similar levels of accuracy. As gradient boosting trees are interpretable linear models, one can actually see a set of intelligible criteria that approximate the results of the deep model.

Others tackle the issue of interpretability in temporal data with attention-mechanisms [15, 19, 20].

3.2 Representation Learning

EHR data is very high-dimensional, noisy, sparse, often incomplete, erroneous, or biased [15], all of which present nontrivial challenges for data analysis. Representation learning based approaches to focuses on transforming EHR patient data into more usable formats that are also clinically meaningful in some way. These representations are often used for tasks such as outcome prediction, and many of the works mentioned in the previous first perform some form of representation learning, then feed the learned representations as input into some other prediction algorithm.

EHR data records contain large numbers of discrete medical codes corresponding to individual patient encounters (e.g. lab tests or administration of medication). Representation learning approaches either learn continuous vector representations for each of these codes, or use an aggregation of these codes to learn a continuous vector representation for each patient [2]. Methods that do the former tend to be unsupervised, learning the structure of the data without any labels, whereas the latter is commonly done via learning to perform some relevant supervised task [11]. Assessment for both of these categories of approaches usually involves evaluation on supervised prediction task using the learned representations as pre-processed input.

3.2.1 Unsupervised Approaches

Unsupervised approaches to representation learning with EHR data typically use an autoencoder architecture. One approach by Miotto et. al. is called DeepPatient, which uses a three-layer stack of denoising autoencoders to project EHR data into a continuous vector space [15]. The autoencoders (each consisting of a two-layer feedforward network) are independently trained, and the final layer is taken as the latent representation for each individual patient. The learned representations were shown to result in superior prediction of diseases when compared to the raw EHR data. Other autoencoder-based approaches use variants of the stacked denoising autoencoders (SDA) used in DeepPatient, or use alternative subnetworks such as

GRUs or other recurrent architectures in their autoencoders [21, 22].

3.2.2 Supervised Approaches

Most supervised approaches to EHR analysis are influenced by approaches *word embeddings*. The prototypical notable word embedding example is Word2Vec [23]. Word2Vec is a technique for learning vector representations of words that uses a neural network to predict surrounding contextual words given some input word, and takes the penultimate layer as the word's "embedding" or vector representation; this is known as a *skip-gram model*. In similar fashion it is possible to train a deep neural network on patient input data for some relevant predictive task, and to take one of the intermediate learned representations a "patient vector".

Zhang et. al. did exactly this with longitudinal EHR data, using it to predict future risk of hospitalization and taking the final intermediate layer as the vector representation for the patient in a system they call *Patient2Vec* [24]. Their approach primarily relies on hierarchical self-attention mechanisms. Other supervised approaches also utilize attention mechanisms on top of recurrent networks [13, 19, 25], and result in state-of-the-art performance on prediction tasks. Others have also achieved comparable or superior results by using non-recurrent attention-only models [20, 26].

3.2.3 Patient Similarity

Rather than focusing on obtaining representations that can be used for predictive tasks, others instead focus on representations that can be used to provide meaningful measures of patient similarity. Patient similarity metrics can be leveraged for personalized treatment recommendation by comparing present patients with past patients; this allows for prediction of future diseases and complications as well as options for course of treatment [27]. In the teleICU context this can be especially valuable in failure cases when the bedside provider loses contact with the remote center. Such a patient similarity method could also enable the clustering of patients into

like groups or cohorts that can be further analyzed to facilitate personalized "precision medicine" [28]. Additionally, using patient similarity to aid clinical-decision making is interpretable by nature, and thus avoids the interpretability problem that many other deep approaches face.

Zhu et. al. take two approaches for determining patient similarity: one unsupervised and one supervised [29]. Both methods rely on the skip-gram model to derive fixed-length embeddings for each medical code in the EHR data and aggregate the embeddings for each patient into an embedding matrix. The unsupervised approach uses the RV coefficient and $dCov$ coefficient on the embedding matrix to measure linear and non-linear relations between pairs of patients. The supervised approach feeds the embedding matrix into a CNN and uses the intermediate feature-maps along with a learned matching matrix to compute a similarity score. The similarity scores in the supervised approach are dependent on which similarity criteria the training labels reflect.

Suo et. al. build on this approach by using a modified "time-fusion" CNN network architecture designed to also use temporal context for determining patient similarity [30]. Suo et. al. extend this even further by using the above architecture in a deep metric learning context with a triplet loss [31]. By clustering patients through the learned similarity metric, experimental results showed improvement over both raw EHR data with cosine or Euclidean metrics and classical distance learning techniques.

Also note that any representation learning approach can produce a patient similarity metric by using the learned representation transformation substituted into a traditional distance metric learning setup with either Euclidean or cosine norms. Lei et. al. used t-SNE to cluster patient representations learned via autoencoders, and found that clusters of patients corresponded to groups with similar mortality levels [22]. However, most of the work with representation learning on EHR data does not focus directly on patient similarity.

Chapter 4

Vision

Our intention is to develop a single class of flexible deep representations that can be used for multiple purposes: both for use as a patient similarity metric and for prediction of various outcome metrics. Not only are these useful applications in the teleICU environment, but they also serve as a way to more comprehensively evaluate the utility of learned representations.

To do this we plan utilize attention-based recurrent models similar to that used by Zhang et. al. [24], but one that is trained to simultaneously predict multiple patient outcome metrics; this is known as *multitask learning*. It has been shown in many cases that jointly learning to solve multiple related tasks at the same time can improve performance for each task individually [32]. The idea is to obtain a single class of learned representations capable of generalizing for use in a variety of clinical prediction tasks; flexible and versatile patient representations should also provide a good basis for patient similarity as well.

We also wish to further utilize the potential of patient similarity metrics derived from these representations by using them to leverage past data to help treat new patients. This can help in predicting future medical complications and can aid in care by showing past courses of treatment for similar cases. This can be particularly valuable for bedside providers in teleICU systems if the system fails or is disconnected

for a period of time and contact is lost with the remote center. The aim is to develop a framework in which aggregated data for a single teleICU patient can be queried against such a database to produce cases that are similar based on derived patient similarity metrics.

Chapter 5

Learning Deep Representations for Clinical Time-Series Data

5.1 Approach

Recurrent neural networks (RNNs) have become the standard for sequence modeling in deep learning, particularly the long short-term memory network (LSTM) [8] and the gated recurrent unit (GRU) [33]. Vaswani et al. introduced the Transformer [10], a sequence-to-sequence model that forgoes recurrence completely and relies only on attention mechanisms. Recently the Transformer has become the state of the art for many sequence modeling tasks. Applications of the Transformer have notably made breakthrough progress in natural language processing [34–37].

For deep modeling of clinical time-series data, most existing work either uses RNNs [14, 38] or uses a combination of RNNs with attention [24, 25]. However, recent work has shown improved performance in clinical modeling with Transformer-based models [26]. We utilize the Transformer in the context of transfer learning, to develop more robust patient representations for use in downstream clinical tasks such as patient outcome prediction or patient similarity comparison.

5.2 Data

5.2.1 Dataset

To train our models we used the Philips eICU Collaborative Research Database [3], a freely accessible multi-center database made available by Philips Healthcare and the MIT Laboratory for Computational Physiology. The dataset contains 200,859 ICU stays for 139,367 unique patients between 2014 and 2015, sourced from 335 ICUs at 208 different United States hospitals.

5.2.2 Data Preprocessing

Cohort Selection Our cohort selection process closely follows that of Harutyunyan et al. [39]. Firstly, we only considered data from adult patients age 18 or above. In addition, we removed any hospital admission with multiple ICU stays. Finally, we removed all hospital admissions missing in-hospital mortality records. After applying these exclusion criteria, the remaining data contained 136,695 unit stays from 117,333 unique patients.

Feature Extraction To facilitate comparison across different datasets we restricted our feature selection to variables available in both the eICU and MIMIC-III databases. For each patient unit stay we extracted 17 physiological variables sampled at regularly spaced intervals. Static variables (e.g. height) were replicated for each timestep. Temporal variables (e.g. vitals and labs) were rounded up to the nearest timestep; for multiple measurements of a variable in the same timestep, only the last measurement was used. Missing values were imputed with the last known measurement; if no previous measurement was available, we imputed a prespecified "normal" value for each variable.

Discrete variables were represented via one-hot encoding, and continuous variables were mean-centered and normalized, resulting in 48 features. We also included

Table 5.1: Selected Clinical Variables and Impute Values

Variable	Impute Value	Modeled as
Capillary refill rate	0	categorical
Diastolic blood pressure chart	59.0	continuous
Fraction inspired oxygen	0.21	continuous
Glascow coma scale eye opening	4	categorical
Glascow coma scale motor response	6	categorical
Glascow coma scale total	15	categorical
Glascow coma scale verbal response	5	categorical
Glucose	128.0	continuous
Heart Rate	86	continuous
Height	170.0	continuous
Mean blood pressure	77.0	continuous
Oxygen saturation	98.0	continuous
Respiratory rate	19	continuous
Systolic blood pressure	118.0	continuous
Temperature	36.6	continuous
Weight	81.0	continuous
pH	7.4	continuous

a binary mask for each of the 17 variables to indicate whether the corresponding measurement was truly observed or was imputed. This mask was concatenated with the rest of the features to produce a vector of length 65.

After removing samples not containing any measurements for the 17 selected variables, the remaining data contained 126,344 patient stays from 112,836 unique patients. 10% of these patients were randomly sampled to form a test set consisting of 12,647 stays for 11,283 patients; the rest was used as training data.

5.3 Models

The models introduced in this section are both based on the Transformer architecture [10]. Two deep models are presented here: the first takes a multitask learning approach and learns to jointly predict multiple clinical outcomes, while the second is an autoregressive generative sequence model. Both of these models are described in more detail below.

5.3.1 General Architecture

We are given a series of clinical measurement values $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_t}$, with $\mathbf{x_i} \in \mathbb{R}^k$ and k being the dimension of the input. Our goal is to extract some representation z of the sequence that can be used for other purposes. In the multitask learning case we wish to use the same representation z for multiple tasks, and in the autoregressive case we want to predict the value of $\mathbf{x_{t+1}}$.

The general architecture used for our models is primarily inspired by the Transformer-based network from Radford et. al. [34], developed as an autoregressive approach to language modeling. We direct the reader to that paper for a detailed background description for both Transformer and the GPT architecture.

Input Most Transformer-based models take in discrete inputs, and transform them to continuous representations via some token embedding. In our case our inputs are already continuous, so we omit any embedding layers. We instead pass our input through a linear layer to project our input from $\mathbb{R}^k$ to $\mathbb{R}^d$, where k is the dimension of the input and d is the dimension of our model.

Transformer Block For our models, each transformer block consists of two sub-layers: a multi-head masked self-attention layer, and a feedforward layer. Each of the sub-layers are followed by a residual connection [40], a dropout layer [41], then layer normalization [42]. Our overall model stacks 5 of these transformer blocks. The remaining part of the architecture is different for each model.

5.3.2 Multitask Learning

It has been shown in many cases that jointly learning to solve multiple related tasks at the same time with a single deep model can improve model generalization, and in some cases even improve performance for each task individually [32]. By training concurrently on multiple related tasks with the same input, the goal is to force

the model to develop flexible representations of the data that generalize to all of the tasks. In our case we take the four tasks set by Harutyunyan et al. [39] for clinical benchmarking: *in-hospital mortality*, *physiologic decompensation*, *length of stay* (LOS), and *phenotype classification*. To perform these 4 prediction tasks, the output of the last transformer block is passed into 4 separate linear layers in parallel. A softmax activation layer is applied to each of the 3 layers corresponding to the single-label prediction tasks; we apply a sigmoid layer on the layer corresponding to the multi-label phenotype classification task.

In-Hospital Mortality In-hospital mortality prediction is framed as a binary classification task to determine whether a given patient died within the same hospital admission as the ICU stay. Given a sequence of clinical measurements, we predict the probability of in-hospital mortality at each timestep, conditioned on all previous clinical measurements. In this case the output value of patient mortality is replicated across all timesteps for each individual unit stay.

Physiologic Decompensation Physiologic decompensation is framed as the task of predicting mortality within the next 24 hours. Given a sequence of clinical measurements, we predict the probability of in-hospital mortality within the next 24 hours at each timestep, conditioned on all previous clinical measurements. In this case the output value can potentially vary depending on how far a given time step is from time of death.

Length of Stay Length of stay is framed as a multi-class classification task, where the amount of remaining time in the ICU was classified into one of 10 classes: one for less than 24 hours, seven classes corresponding to each day of the first week, one class for between 1-2 weeks, and an additional class for above 2 weeks. Given a sequence of clinical measurements, we learn a probability distribution for the remaining length of stay across the aforementioned classes, conditioned on all previous clinical measurements. In this case the output value varies based on how close a given timestep

is to time of unit discharge.

Phenotyping Phenotype classification is framed as a multi-label multi-class classification task with 25 labeled phenotypes, each corresponding to ICD-10 groupings defined by Health Cost and Utilization (HCUP) Clinical Classification Software (CCS). These 25 phenotypes are the same ones used for benchmarking by Harutyunyan et al. [39], and are listed in Table 5.2. To accommodate the multi-label nature of the problem, we use replace the softmax activation on the final linear layer with a sigmoid activation, using a threshold value of 0.5. At each timestep we aim to predict all relevant diagnoses, both past and future; this results in a phenotype label value that is fixed across all timesteps for each individual unit stay.

5.3.3 Autoregressive Generative Sequence Modeling

Given a sequence of values $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, autoregressive generative models aim to model the joint probability distribution $p(\mathbf{X})$. This can be factored as the product of conditional probabilities as follows:

$$p(\mathbf{X}) = \prod_{i=t}^{n} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \ldots, \mathbf{x}_1) \tag{5.1}$$

We model the conditional probability p with the Transformer-based neural network described earlier, which is trained to minimize the negative log-likelihood $\log p(\mathbf{X})$, written as follows:

$$\mathcal{L}(\mathbf{X}) = -\sum_{t=1}^{n} \log p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \ldots, \mathbf{x}_1) \tag{5.2}$$

Although modeling the conditional probabilities with a continuous distribution appears to be the most straightforward approach, it has been shown that using discrete distributions often result in improved performance, regardless of any implicit continuous nature of the output [43, 44].

Table 5.2: Selected ICU Phenotypes

Variable	**Type**
Acute and unspecifed renal failure	acute
Acute cerebrovascular disease	acute
Acute myocardial infarction	acute
Cardiac dysrhythmias	mixed
Chronic kidney disease	chronic
Chronic obstructive pulmonary disease	chronic
Complications of surgical/medical care	acute
Conduction disorders	mixed
Congestive heart failure; nonhypertensive	mixed
Coronary atherosclerosis and related	chronic
Diabetes mellitus with complications	mixed
Diabetes mellitus without complication	chronic
Disorders of lipid metabolism	chronic
Essential hypertension	chronic
Fluid and electrolyte disorders	acute
Gastrointestinal hemorrhage	acute
Hypertension with complications	chronic
Other liver diseases	mixed
Other lower respiratory disease	acute
Other upper respiratory disease	acute
Pleurisy; pneumothorax; pulmonary collapse	acute
Pneumonia	acute
Respiratory failure; insufciency; arrest	acute
Septicemia (except in labor)	acute
Shock	acute

Specifically, we use a discretized logistic mixture likelihood to model the conditional probabilities, similar to that used by Salimans et al. [44] for image pixel generation. However, our likelihood model differs in that the probability output space is unbounded; the modified likelihood is described as follows.

We assume that each variable in $\mathbf{x}_t$ has an independent latent continuous distribution that can be modeled as a mixture of logistic distributions; we then round it to a discrete value. In our case all of our inputs $\mathbf{x}_t$ are normalized to have unit standard deviation and zero mean, so we round each value to the nearest $\frac{1}{10}$ as part of pre-processing. Noting that the CDF of the logistic distribution is simply the logistic sigmoid function, we then calculate the log-likelihood of any x_t observed from the data as follows:

$$\lambda_{t,i}(z) = \sum_{j=1}^{M} \pi_{i,j} \left[\sigma \left(\frac{z + r/2 - \mu_{i,j}}{s_{i,j}} \right) - \sigma \left(\frac{z - r/2 - \mu_{i,j}}{s_{i,j}} \right) \right] \tag{5.3}$$

$$\log p(\mathbf{x}_t \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \mathbf{s}) = \sum_{i=1}^{k} \log(\lambda_{t,i}(x_{t,i})) \tag{5.4}$$

where M is the number of mixture components, k is the dimension of input $\mathbf{x}_t$ σ is the logistic sigmoid function. For our experiments we set $r = 0.1$.

To predict the output distribution of $\mathbf{x}_{t+1}$ given $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_t$, we pass the output of the last transformer block through 3 linear layers in parellel; the outputs of these layers are then reshaped into the $k \times M$ parameter matrices $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, and $\mathbf{s}$.

We used this loss in addition to the loss from the four tasks used by the multitask model; the hope is that the added autoregressive loss helps the model to generalize better across various tasks.

5.4 Training

Training procedure was identical for both models. Of the 119,282 sequences in the training set, 10% of these were randomly sampled to be held-out and used as a

validation set. We used the Adam optimizer [45] with hyperparameters $\beta_1 = 0.9, \beta_2 = 0.999$, and a learning rate of 2e$-$4. Parameter gradients were clipped to a maximum norm of 5. All transformer blocks used the Gaussian Error Linear Unit (GELU) [46] as an activation function, and used a dropout rate of 0.1. We performed hyperparameter optimization via random search. Our models were each trained for 100 epochs on minibatches of 32 randomly sampled training sequences.

Chapter 6

Outcome Prediction with Learned Representations

Recall that one of the major motivations for improved vector patient representations is to facilitate the prediction of patient outcomes. In this section we evaluate the efficacy of the methods introduced in the previous section when used to predict patient outcomes on unseen data.

6.1 Supervised Fine-Tuning

Given a series of clinical measurement values $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t]$, the general architecture of our transformer-based architectures produces vectors $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_t$ in the output of the last transformer block; we take the last output in the sequence $\mathbf{h}_t$ as the learned representation of the sequence.

In general, we can perform both classification and regression by passing $\mathbf{h}_t$ obtained from a pretrained model through a linear layer with the correct activation. In the case of classification, if the input sequence has label y, we can predict a distribu-

tion across label classes as follows:

$$P(y \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t) = \text{softmax}(\mathbf{h}_t \mathbf{W}_y) \tag{6.1}$$

where $\mathbf{W}_y$ is a matrix of trainable weights.

Similar to [34], we also use the loss for pretrained model's task as an auxiliary loss to accelerate convergence in training and improve generalization performance for the supervised model.

6.2 Experiments

6.2.1 Data

The experiments and visualizations in this section are done using the held-out test set of the Philips eICU Collaborative Research Database; this consists of 12,647 ICU stays for 10,875 unique patients. Note that this is not the full dataset; just the portion that was not used for the initial training in chapter 5. We further partition our test-set for the purposes of supervised finetuning; the data from 6,525 randomly selected patients is used for additional training, and we evaluate on the remaining 4,350 patients. We consider each subsequence of patient measurements that starts from ICU admission as a unique data point, which results in a training set of size 435,504 and a test set of size 292,902.

6.2.2 Outcome Prediction Tasks

We perform outcome prediction using the four tasks set by Harutyunyan et al. [39] for clinical benchmarking: in-hospital mortality, physiologic decompensation, length of stay, and phenotype classification. The predictions are generated in the same way described in chapter 5.3.2.

6.2.3 Baselines

Logistic Regression Our logistic regression baseline model uses hand-engineered features based on the baseline model from Harutyunyan et al. [39]. Our raw input sequence is of the form $\mathbf{X}^{(i)} = [\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \ldots, \mathbf{x}_t^{(i)}]$, where each $\mathbf{x}_j^{(i)}$ is a vector of length 17, corresponding to the values of the 17 variables specified in Table 5.1. We consider the following 7 subsequences: the first 10%, 25% and 50% of the sequence, the last 10%, 25% and 50% of the sequence, and the full sequence. For each subsequence we calculate the following 6 statistics per variable: minimum, maximum, mean, standard deviation, skew, and number of measurements. Features for missing measurements are substituted with predefined "normal" values. This results in a total of $17 \times 7 \times 6 = 714$ features, which are then mean-centered and normalized.

LSTM We also test against deep supervision LSTM model baselines as described in [39]. The input to the LSTM is formatted in the same way as the input to the Transformer-based models defined in chapter 5, where each timestep of clinical observations is represented as vector of length 65. Since the LSTM processes input sequentially, we can impose additional constraints on the hidden state output to obtain an output value for each timestep; it has been shown that this type of deep supervision can improve performance [38, 39].

Multitask LSTM We also train an LSTM model baseline that simultaneously predicts all 4 outcome metrics as described in [39], similarly to the case of the multitask Transformer model introduced earlier.

6.3 Results

In the binary classification tasks for mortality and decompensation we used the area under the receiver operator characteristic curve (AUC-ROC) as our primary metric,

Table 6.1: Outcome Prediction Results

(a) In-Hospital Mortality

Model	**AUC-ROC**	**AUC-PR**
Logistic Regression	0.698	0.285
LSTM	0.866	0.528
Multitask LSTM	0.883	0.573
Multitask Transformer	0.901	0.621
Autoregressive Transformer	**0.905**	**0.650**

(b) Physiologic Decompensation

Model	**AUC-ROC**	**AUC-PR**
Logistic Regression	0.629	0.058
LSTM	0.878	0.595
Multitask LSTM	0.905	0.634
Multitask Transformer	0.918	0.695
Autoregressive Transformer	**0.927**	**0.725**

(c) Length of Stay

Model	**Cohen's Kappa**
Logistic Regression	0.180
LSTM	0.230
Multitask LSTM	0.262
Multitask Transformer	**0.332**
Autoregressive Transformer	0.324

(d) Phenotyping

Model	**Macro AUC-ROC**	**Micro AUC-ROC**
Logistic Regression	0.541	0.865
LSTM	0.662	**0.906**
Multitask LSTM	0.619	0.895
Multitask Transformer	0.632	0.902
Autoregressive Transformer	**0.681**	0.905

the most commonly used statistic for mortality prediction. We additionally report the area under the precision-recall curve (AUC-PR), which can be a more informative metric when dealing with imbalanced classes. Cohen's linear weighted kappa [47] is used as a metric for length of stay classification. For a multilabel classification problem, Cohen's kappa statistic measures the level of agreement between two label assignments (i.e. predicted LOS vs actual LOS) with a score between -1 and 1, where a value of 1 indicates complete agreement and a value of -1 indicates complete disagreement. For multilabel phenotyping we take both the macro-averaged and micro-averaged AUC-ROC; any classes that are missing entirely from the test set are omitted in this calculation. The results for each of the mortality, decompensation, LOS, and phenotyping tasks are reported in Table 6.1.

6.4 Discussion

Similarly to other clinical ICU benchmarks [12, 39], we found that logistic regression on raw or hand-engineered features is consistently outperformed by deep models. In our experiments, logistic regression had the lowest performance across the board, for all tasks and in all metrics.

The representations learned by the multitask LSTM outperformed the task-specific LSTM for mortality, decompensation, and length of stay in the relevant metrics, but performed worse that the task specific LSTM for phenotype classification.

Our fine-tuned Transformer models generally outperformed all of the LSTM models on all tasks; we found that the fine-tuned representations learned by Transformer-based architectures were comparable for phenotyping and significantly superior for the other three tasks. The model with the added autoregressive loss was the best model overall. It increased performance over the multitask-only Transformer model for all tasks except length of stay, where it performed only slightly worse.

We also include the ROC curves for the four deep models on both mortality and

decompensation prediction in Figure 6-1 and Figure 6-2 respectively, to visualize the tradeoff between true positive rate and false positive rate for each model.

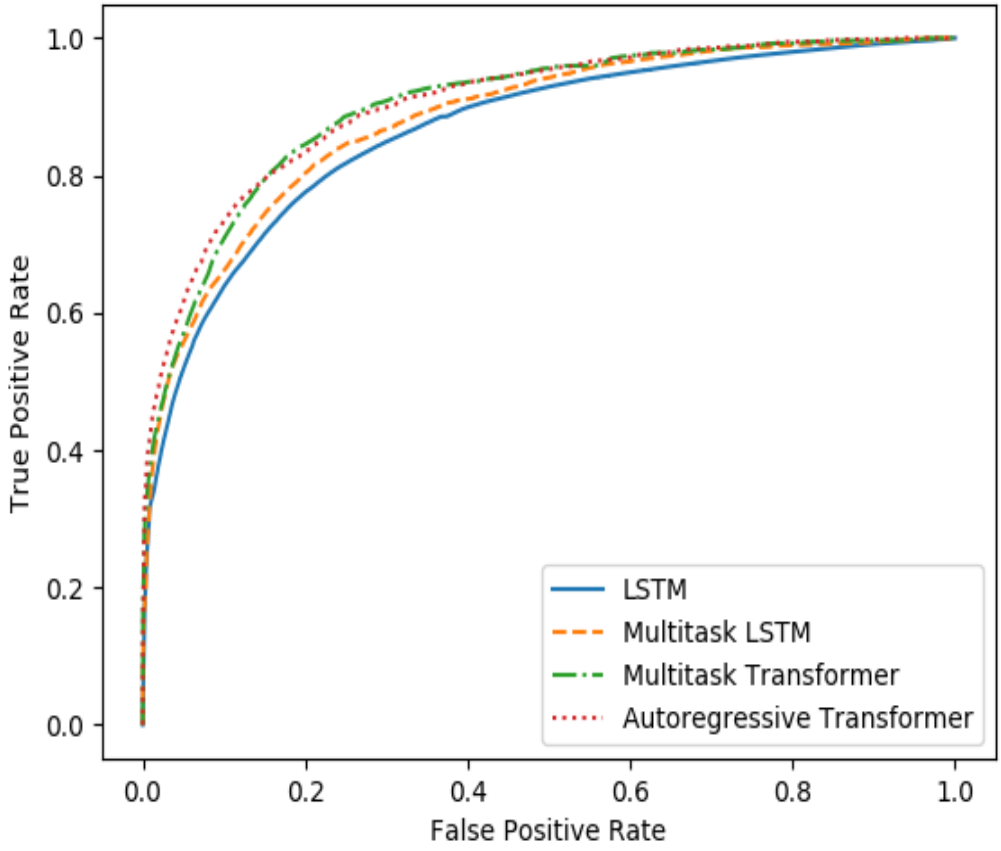

Figure 6-1: ROC Curves for Mortality Prediction

6.4.1 Calibration

For binary classification tasks such as mortality and decompensation can also be framed as risk prediction. It is informative to know how reliable our predicted probabilities are to understand how accurately we can use them as a measure of risk. Such reliability is measured via what is called *calibration*, a method which is often used for evaluating predictive clinical models [39].

In a perfectly calibrated model, given a group of patients with predicted mortality

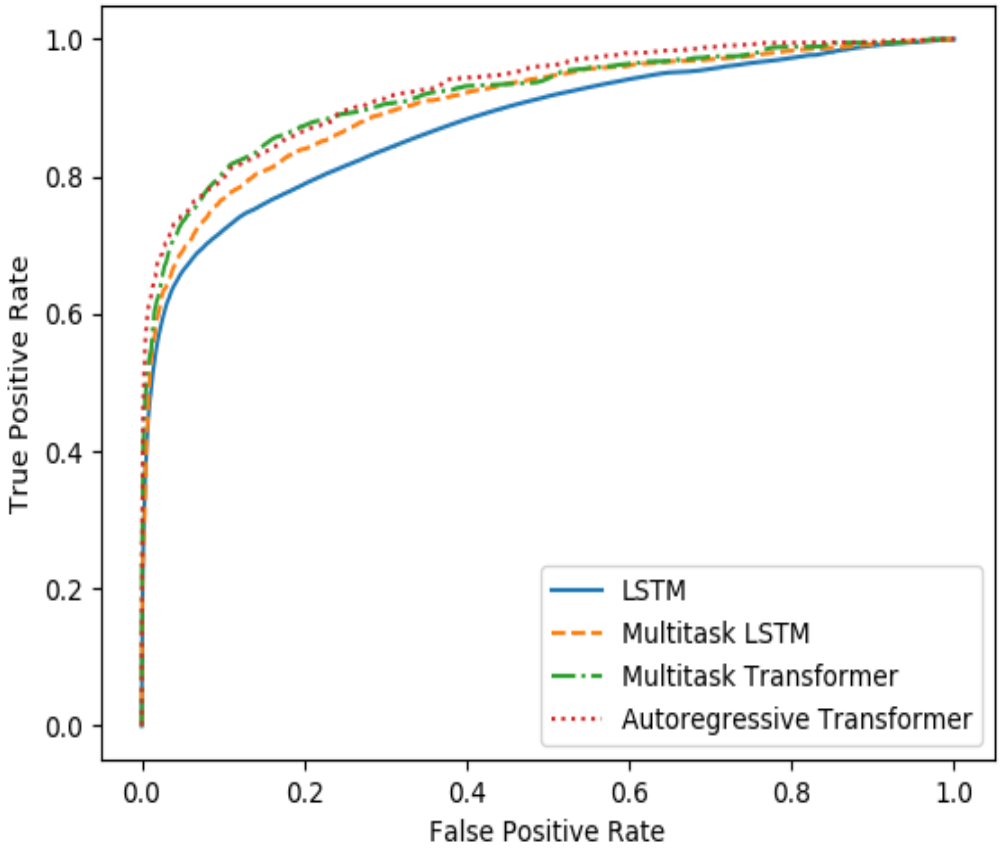

Figure 6-2: ROC Curves for Decompensation Prediction

of 0.2, 20% of these patients actually do decease. We visualize calibration on the evaluation portion of the held-out test set for each of the 4 deep models, with both mortality and decompensation tasks; these can be found in Figures 6-3 and 6-4. We found that the LSTM and autoregressive Transformer models were reasonably calibrated. However, the multitask LSTM model tended to overestimate risk, and the multitask Transformer model tended to underestimate risk. The addition of the autoregressive loss appears to correct for the multitask model's underestimations.

In general, the models were calibrated worse for decompensation than for mortality, which is unsurprising due to the greater class imbalance for decompensation prediction.

6.4.2 K-Fold Cross Validation

Another important factor is the ability of outcome prediction models to generalize across hospitals. To facilitate this, we performed a k-fold cross validation with $k = 7$ for our best-performing autoregrssive transformer model, where partitions were split by hospital. The data was split into 7 parts, sourced from disjoint sets of hospitals. For each of the 7 data partitions we retrained the autoregressive transformer on the other 6, and evaluated on the held-out partition. The results can be found in Table 6.2.

We were able to achieve comparable results with k-fold cross validation for the autoregressive Transformer model, with generally small variance across folds, albeit a relatively small sample size. The worst performing fold for phenotyping actually outperformed all of the other fully-trained models on Macro AUC-ROC; similarly we found that the worst folds for mortality and decompensation outperformed all of the other fully-trained models except the multitask Transformer.

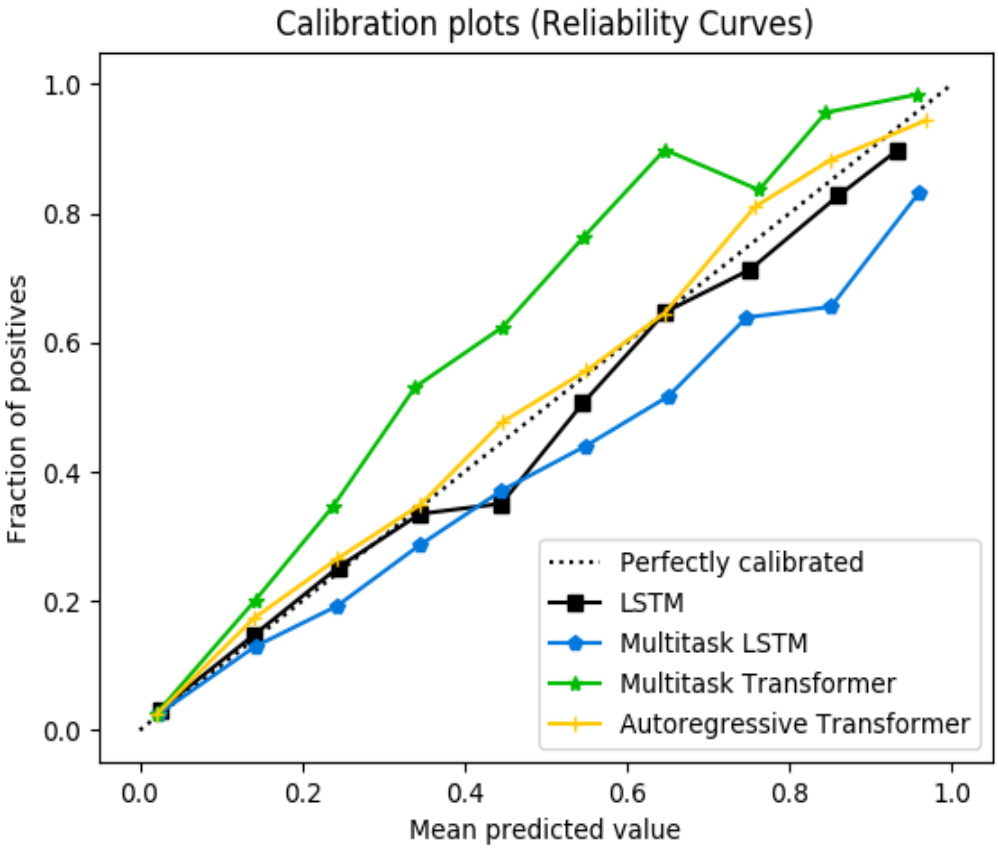

Figure 6-3: Mortality Calibration

Better calibrated predictions will fall closer to the diagonal.

We see that the LSTM model is reasonably calibrated, only slightly overestimating the actual probability of mortality.

The multitask LSTM model, while outperforming the LSTM model on metrics such as AUC-ROC and AUC-PR, is noticeably worse calibrated. This model consistently overestimates the actual probability of mortality.

The multitask transformer model consistently underestimates the actual probability of mortality. Again, while outperforming the LSTM model on our metrics, it is noticeably worse calibrated.

Our autoregressive transformer model has the best calibration out of all four deep models, in addition to performing the best on the AUC-ROC and AUC-PR metrics.

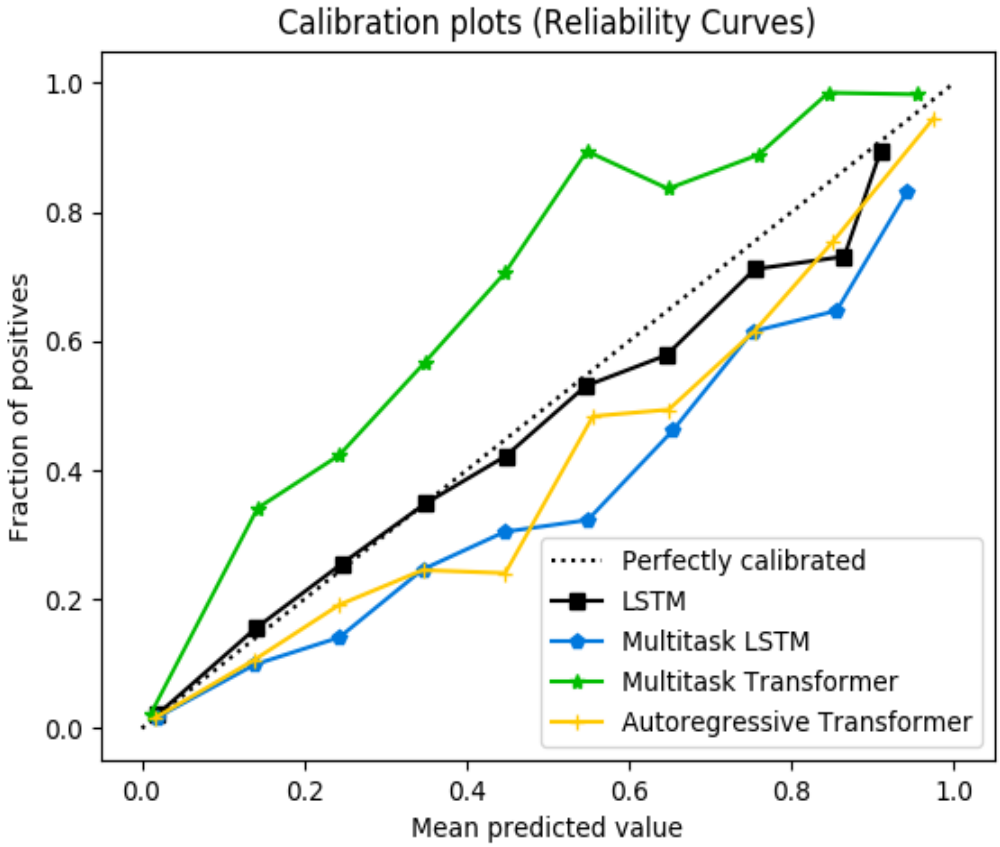

Figure 6-4: Decompensation Calibration

Better calibrated predictions will fall closer to the diagonal.

Similarly to the mortality task, we see that the LSTM model is calibrated quite well.

The multitask LSTM model slightly overestimates the actual probability of decompensation.

The multitask transformer model is similarly calibrated for decompensation as it was for the mortality task, consistently underestimating the actual probability of decompensation.

Our autoregressive transformer model slightly overestimates the actual probability of decompensation. For the decompensation task, this model is calibrated comparably to if not slightly better than the multitask LSTM.

Table 6.2: K-Fold Cross Validation Results (Autoregressive Transformer)

(a) In-Hospital Mortality

Fold #	**AUC-ROC**	**AUC-PR**
1	0.908	0.655
2	0.914	0.689
3	0.897	0.612
4	0.884	0.574
5	0.919	0.696
6	0.914	0.678
7	0.898	0.659
Average	**0.904**	**0.652**
Standard Deviation	**0.011**	**0.041**

(b) Physiologic Decompensation

Fold #	**AUC-ROC**	**AUC-PR**
1	0.924	0.737
2	0.944	0.778
3	0.934	0.708
4	0.915	0.652
5	0.915	0.702
6	0.936	0.766
7	0.915	0.725
Average	**0.926**	**0.724**
Standard Deviation	**0.011**	**0.039**

(c) Length of Stay

Fold #	**Cohen's Kappa**
1	0.343
2	0.300
3	0.282
4	0.259
5	0.309
6	0.417
7	0.284
Average	**0.313**
Standard Deviation	**0.049**

(d) Phenotyping

Model	**Macro AUC-ROC**	**Micro AUC-ROC**
1	0.619	0.875
2	0.596	0.896
3	0.659	0.869
4	0.647	0.889
5	0.715	0.887
6	0.692	0.839
7	0.576	0.891
Average	**0.643**	**0.878**
Standard Deviation	**0.046**	**0.012**

6.4.3 Finetuning Convergence

The autoregressive model converged fairly quickly in the fine-tuning phase, always in 3 epochs or less. In fact, often the learned representations were so robust that fine-tuning had no effect on performance; this was the case for the tasks of in-hospital mortality and physiologic decompensation prediction. One general disadvantage of deep models compared to traditional machine learning and statistical technqiues is the computational intensiveness of training; the relatively quick convergence of the pretrained autoregressive model during finetuning mitigates this significantly.

Chapter 7

Patient Similarity with Learned Representations

Beyond just predicting outcomes, learned patient representations present a data-driven heuristic tool for comparing patients. It is important to note that there is no single objective definition for patient similarity, and different similarity metrics may be desired for different use cases. In one case one might desire to group patients by diagnosis; in another case mortality may be preferable. Here we evaluate general similarity in the output space for our learned multitask representations, but any custom similarity metric can be used on top of the learned representations.

7.1 Visualizations

For any similarity metric, the goal is to have patients with similar characteristics located close to each other in the patient representation space. The visualizations in this section map our learned patient representations to 2D space; we then analyze the distributions of specific classes of patients within this space, looking for any structure or clustering. For each patient class or characteristic, the presence of high-level structure in 2D space like dense clusters indicates that similar structure exists

in the higher dimensional patient representation space, which in turn suggests that even basic Euclidean distance in this higher dimensional space can serve as a useful similarity metric for that particular characteristic.

The idea here is not to replace existing methods of patient similarity, but to augment them by feeding in better input representations as opposed to raw clinical data. Intuitively, it is easier to learn a similarity metric in a space that already has a relatively simple high-level structure; if we can show evidence that this high-level structure exists for our learned patient representation space across a wide range of patient classes and characteristics, we demonstrate the general potential our patient representations have either for use alone with a simple metric or in combination with other more complex metrics for patient similarity.

7.1.1 Interpreting the Visualizations

Essentially, the Transformer models take in sequences of patient clinical data and map them to vectors of length 128. The dimensionality reduction techniques presented in this chapter take these vectors of length 128 and map them to vectors of length 2, which can be interpreted as (x, y) coordinates in 2D space; these techniques create a mapping in way such that points that are close in the patient representation space (i.e. 128-dimensional space) are mapped to nearby points in 2D space, and points that are distant in the patient representation space are mapped to distant points in 2D space. The idea is that if we can discover structure in the 2D space, this indicates the existence of corresponding structure in the patient representation space.

The visualizations in this chapter are in the form of heatmaps, where each heatmap shows the distribution of a particular patient characteristic (e.g. mortality) across all test patients. First, the 2D output space is split into separate blocks, then patients that are mapped to (x, y) coordinates that fall within the same block are grouped together. Each block is then assigned a color indicating the density or prevalance of the given patient characteristic within that particular block; darker blocks indicate

a higher concentration of the given patient characteristic. For example, when considering mortality, a dark block indicates a higher relative mortality rate for patients mapped to that block as opposed to a lighter colored block.

We are looking for evidence that our learned representations impose higher level structure on the patient space; particularly, the hope is to see patients of the same patient characteristic grouped together spatially (i.e. high density patient blocks clustered together). Since the representations are meant to be general purpose, patients for a single characteristic being broken into more than one cluster is not unexpected, as the separate clusters likely correspond to another separate patient characteristic varying (e.g. within a given phenotype, perhaps patients with short LOS are in one cluster, and patients with longer LOS are within another).

For all the 2D visualizations pictured, the axes do not have any inherent meaning or correspond to any units; they are simply axes with scale defined by the given transformation to 2D space. Also note that the projections shown by PCA and t-SNE are different, and so (x, y) coordinates for the PCA graphs do not necessarily correspond to the same (x, y) coordinates in the t-SNE plots.

7.2 Principal Component Analysis

We perform dimensionality reduction with principal component analysis (PCA), and projecting the patient matrix of learned representations into two dimensions by selecting the first two principal components.

We split this space into $30 \times 30 = 900$ discrete blocks, and calculate label density at each location i, j as follows:

$$H_{i,j} = \frac{L_{i,j}}{T_{i,j} + c} \tag{7.1}$$

where $L_{i,j}$ is the label frequency in block (i, j), $T_{i,j}$ is the total count for (i, j), and $c = 5$ is an empirically set constant smoothing factor. We then create the resulting

heatmap from matrix H for each label class.

7.3 T-Distributed Stochastic Neighbouring Entities

We also explore an alternative dimensionality reduction technique in t-distributed stochastic neighbouring entities (t-SNE) [48]. T-SNE works by modeling the relationship between neighboring points as a probability distribution, then recreates the original data in lower dimensional space to match the distribution. It can be quite computationally intensive as the number of samples grows large, so our visualizations use a randomly selected subset containing 100,000 samples. Heatmaps for t-SNE are constructed in the same way described in the previous section for PCA.

7.4 Results

In the figures presented throughout the rest of this section, we observe that each class of patient is clustered into particular block locations, and many of these blocks have high densities of 80% or above. Examining the phenotype visualizations we see that in many cases most of the relevant patients are clustered within very few blocks; however, PCA was much better than t-SNE at clustering patients of a phenotype from different blocks together.

One interesting result is that the PCA visualizations tended to have increasing mortality along the y-axis and increasing length of stay along the x-axis, in addition to various phenotypes clustered in spots. Examining these different characteristics together, the PCA transformation of our learned representations indicates a high level of structure in the patient representation space. This representation space learned by our models shows significantly more structure for instance than that from Lei et. al [22], who use representations from a deep recurrent LSTM variant for patient

similarity analysis.

Our observations indicate that our learned representations can be useful for comparing patients across a variety of clinical aspects. In general, patients with similar characteristics tend to cluster together spatially, which indicates that Euclidean distance on its own in PCA or t-SNE transformed space can serve as an informative metric of patient similarity.

It is important to note that such a similarity metric is not necessarily intended to calculate probability of any specific patient outcome; we have already shown that our learned representations can be used to do so with models designed for that very purpose. However, the fact that our metric tends to group patients of the same type near each other allows us to use it as a heuristic for general patient similarity across multiple clinical aspects.

7.5 Visualizations of Sepsis Patients

We also use PCA to create a visualization of patients with a teleICU admission diagnosis of sepsis, which represent an important class of ICU patients. This is particularly interesting considering the fact our learned representations were not trained in any way on ICU admission diagnoses or any pre-ICU diagnoses at all.

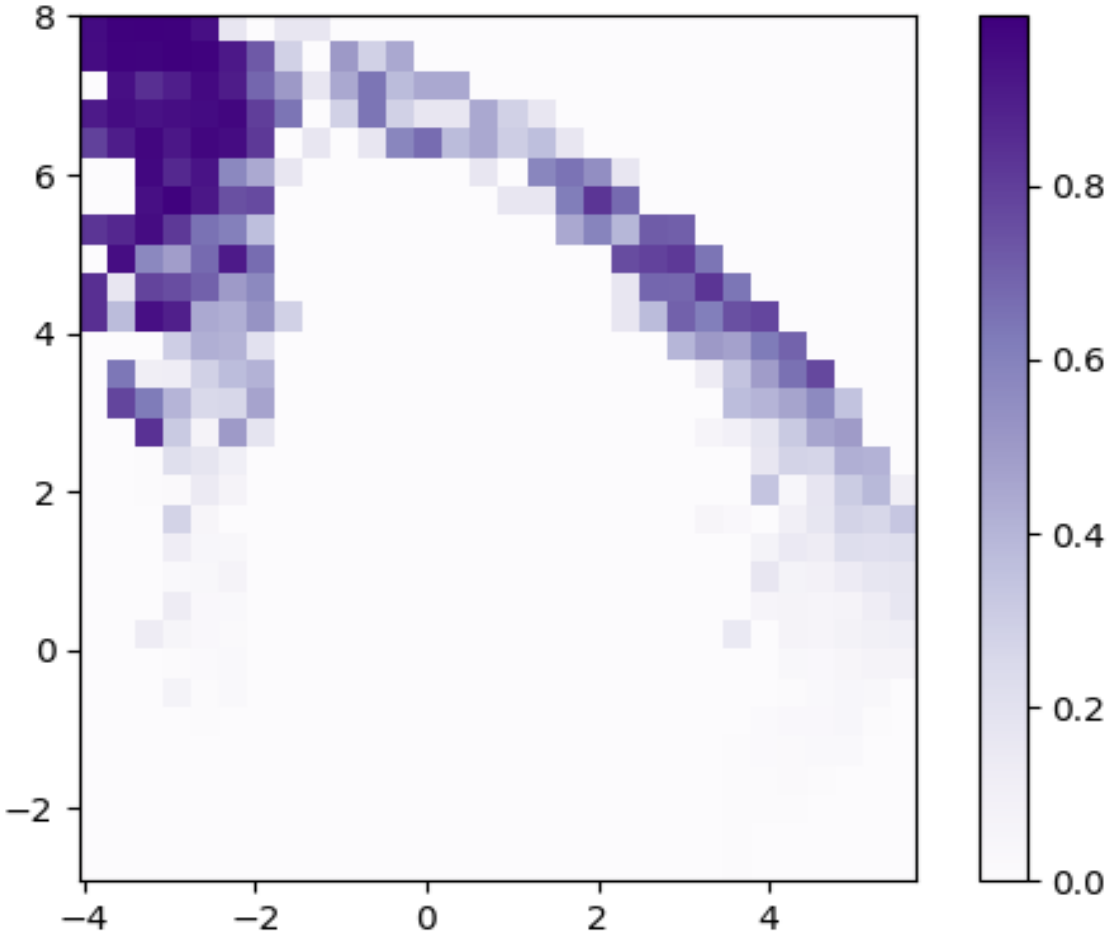

Figure 7-1: Mortality Visualization with PCA

Darker colors correspond to higher mortality rates.

We see that deceased patients are generally clustered into neighboring blocks in the top-left or along a curve in the top-right portion of the graph. Most of these blocks have high density, with mortality rates of 80% or higher. In general, it appears mortality rates increase along the y-axis.

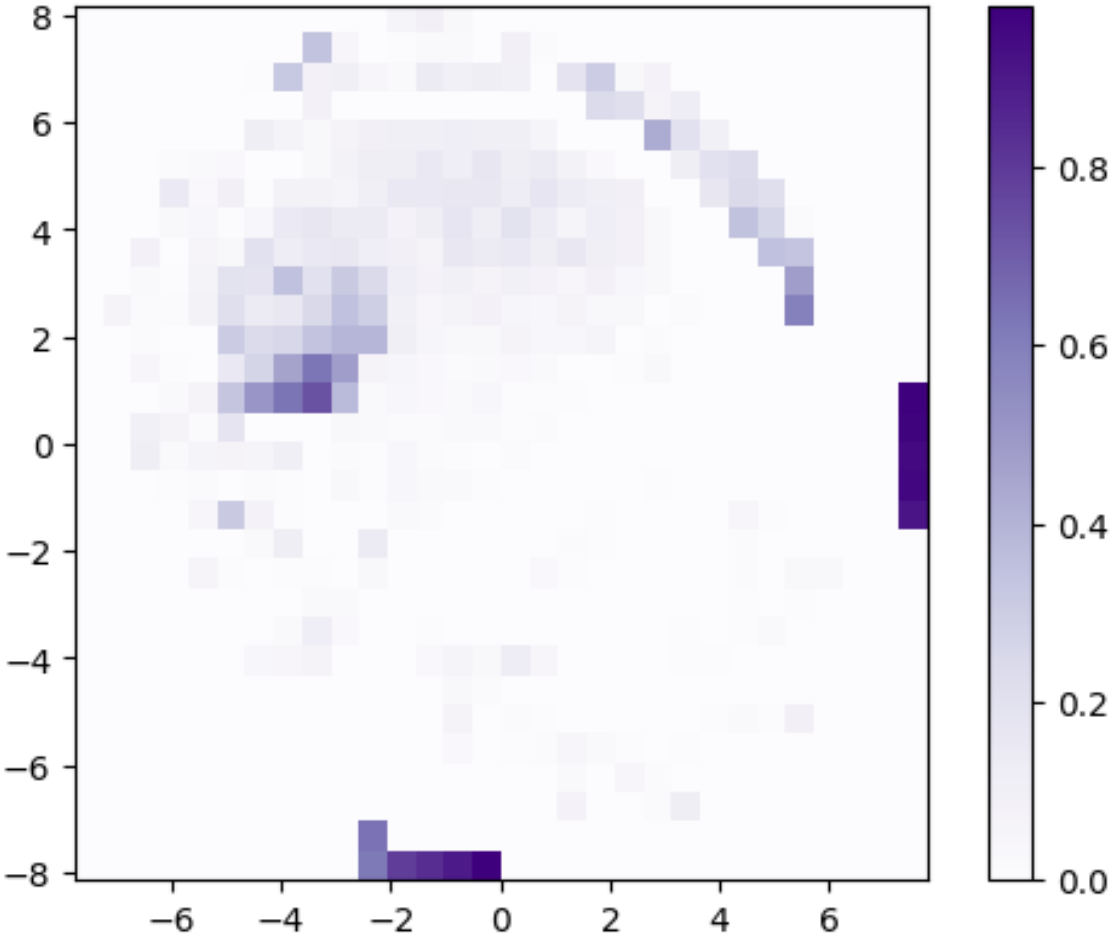

Figure 7-2: Mortality Visualization with t-SNE

Darker colors correspond to higher percentages of 24-hour decompensation.

We see that blocks with high mortality rates are clustered together either in the right or bottom portion of the graph. An additional area of medium-density blocks can be found on the left side of the graph. Patients whose t-SNE transformed representations lie in the dozen or so high-density blocks have a significantly higher mortality rates than the rest of the graph.

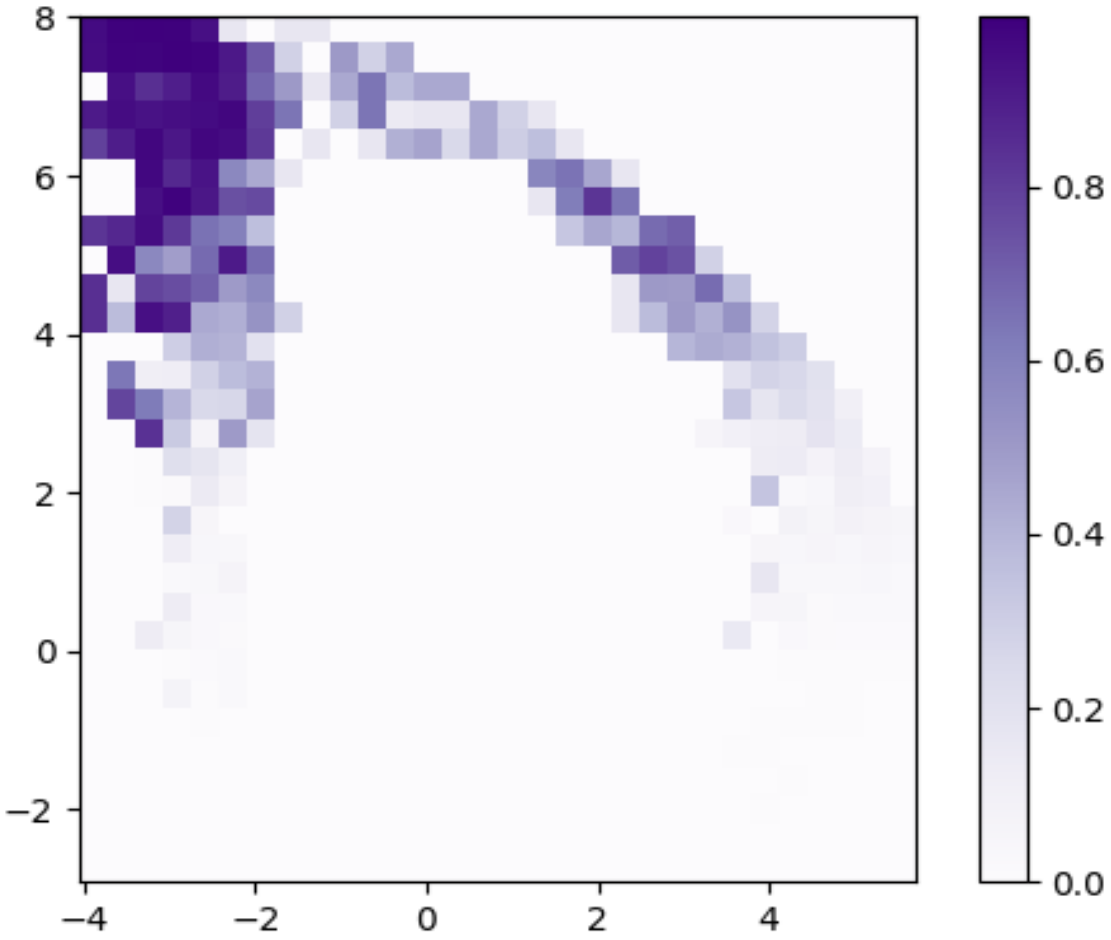

Figure 7-3: Decompensation Visualization with PCA

Darker colors correspond to higher mortality rates.

This PCA visualization of decompensation (i.e. death in the next 24 hours) is remarkable similar to that for mortality, which is unsurprising considering the relatedness of the two tasks and the fact that a large number of patients in our dataset have a total length of stay of less than 24 hours. High-risk (high density / mortality rate) blocks are clustered together in the top-left or along a curve in the top-right portion of the graph. Similarly to the mortality PCA visualiztion, decompensation rates appear to increase along the y-axis.

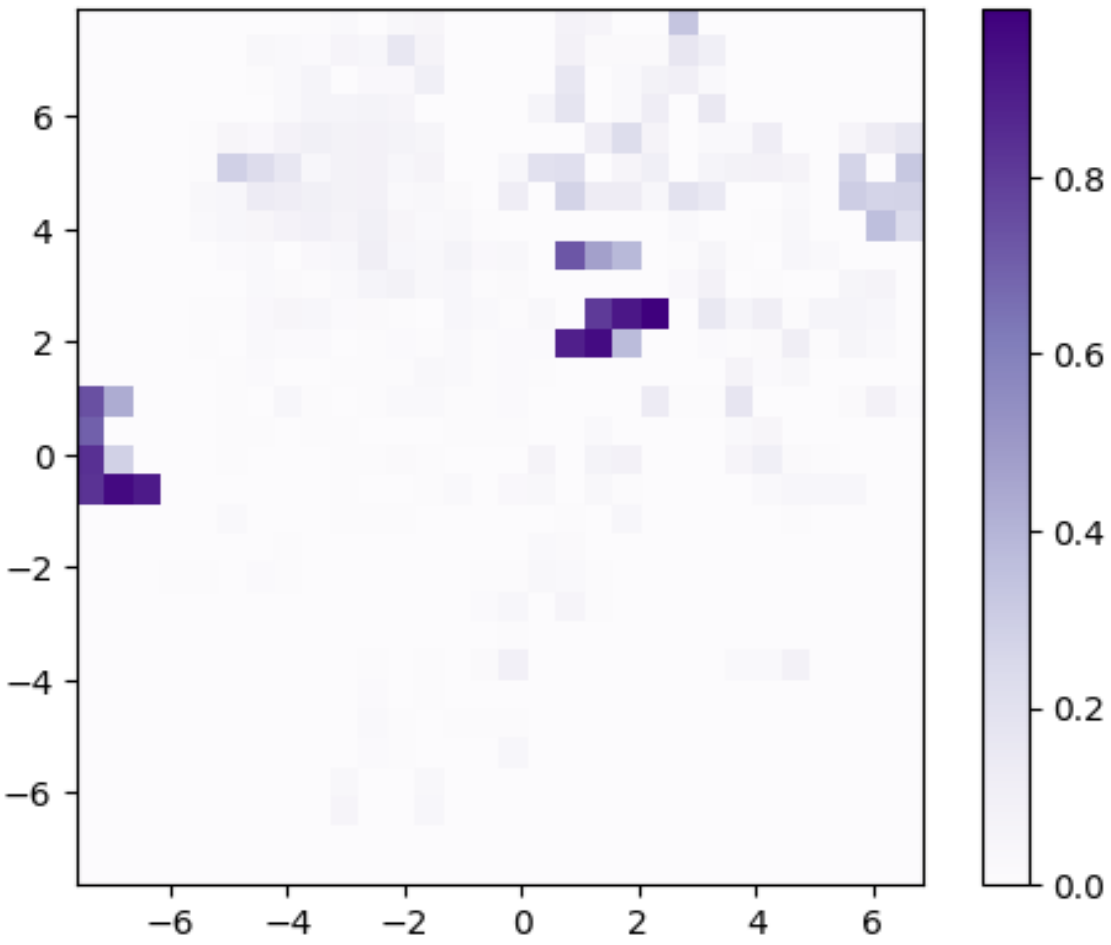

Figure 7-4: Decompensation Visualization with t-SNE

Darker colors correspond to higher percentages of 24-hour decompensation.

We see two clusters of high-risk blocks on the left and in the center of the graph. Similar to the t-SNE visualization for mortality, most at-risk patients are clustered into a dozen or so high-density blocks with significantly higher mortality rates than the rest of the graph.

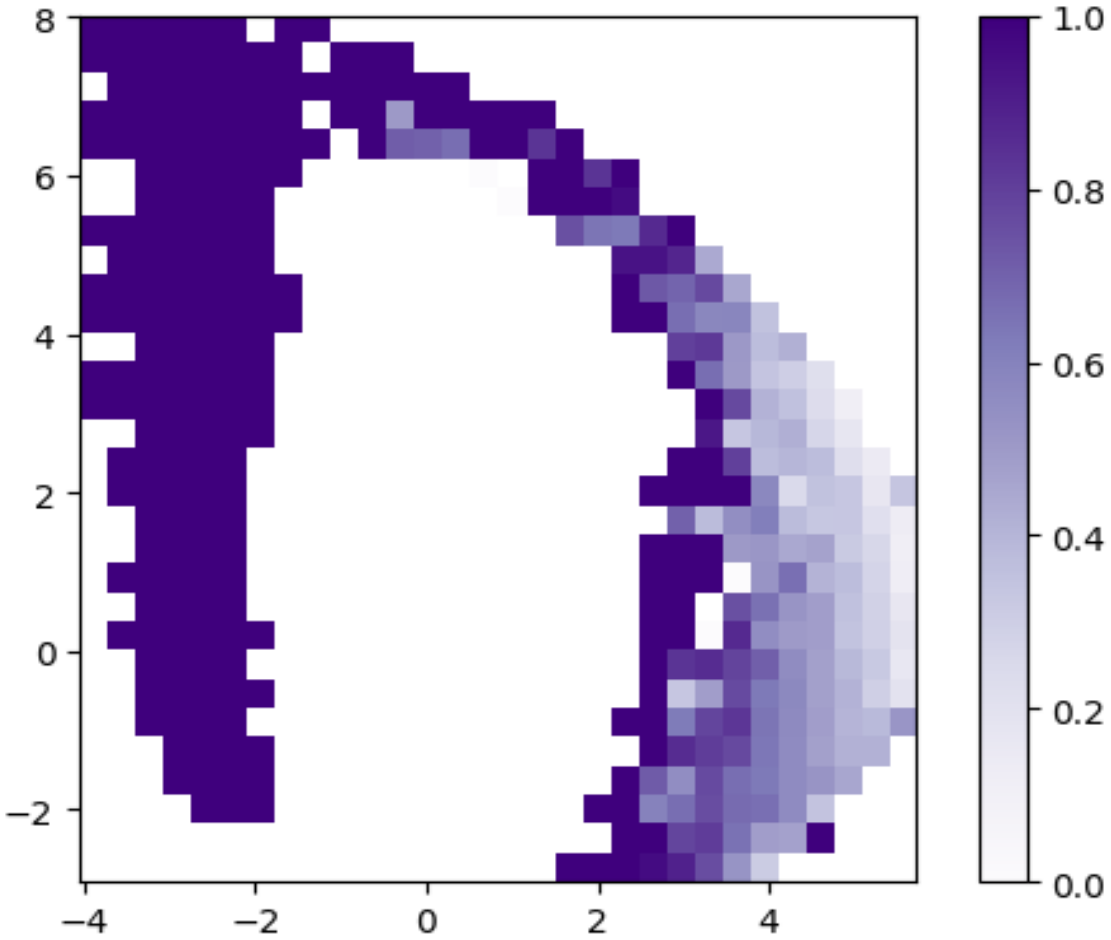

Figure 7-5: Length of Stay Visualization with PCA (< 24 hours)

Darker colors correspond to higher percentages of the remaining length of stays of < 24 hours.

Patients with length of stay less than 24 hours are clustered together on the left or along a curve on the right side of the graph. Block density decreases (or length of stay increases) as we move along the x-axis. As most patients have lengths of stay less than 24 hours, it is not surprising to see than much of the graph consists of high density blocks.

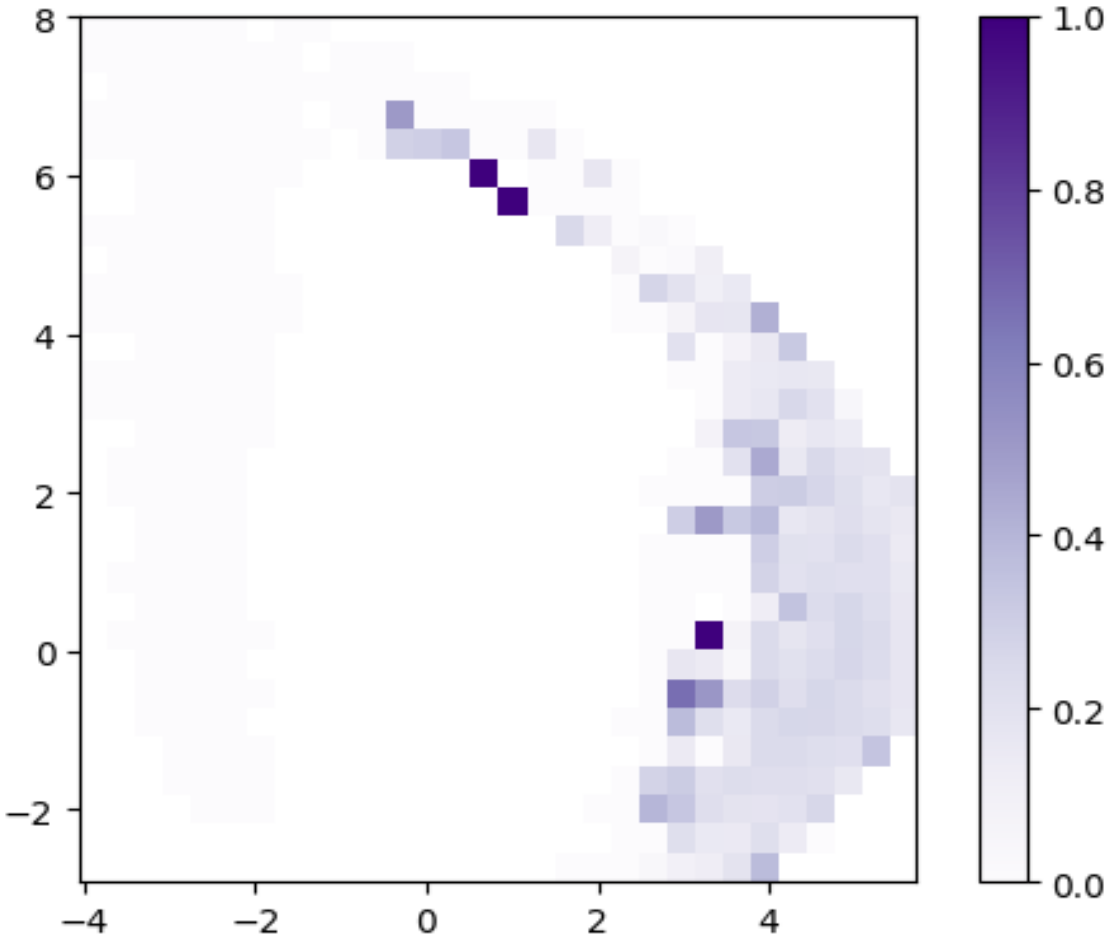

Figure 7-6: Length of Stay Visualization with PCA (1-2 days)

Darker colors correspond to higher percentages of the remaining length of stays between 1 and 2 days.

Patients with length of stay between 1 and 2 days are mostly clustered together in three high-density blocks. We again see that block density decreases as we move along the x-axis, which is consistent with length of stay increasing along the x-axis.

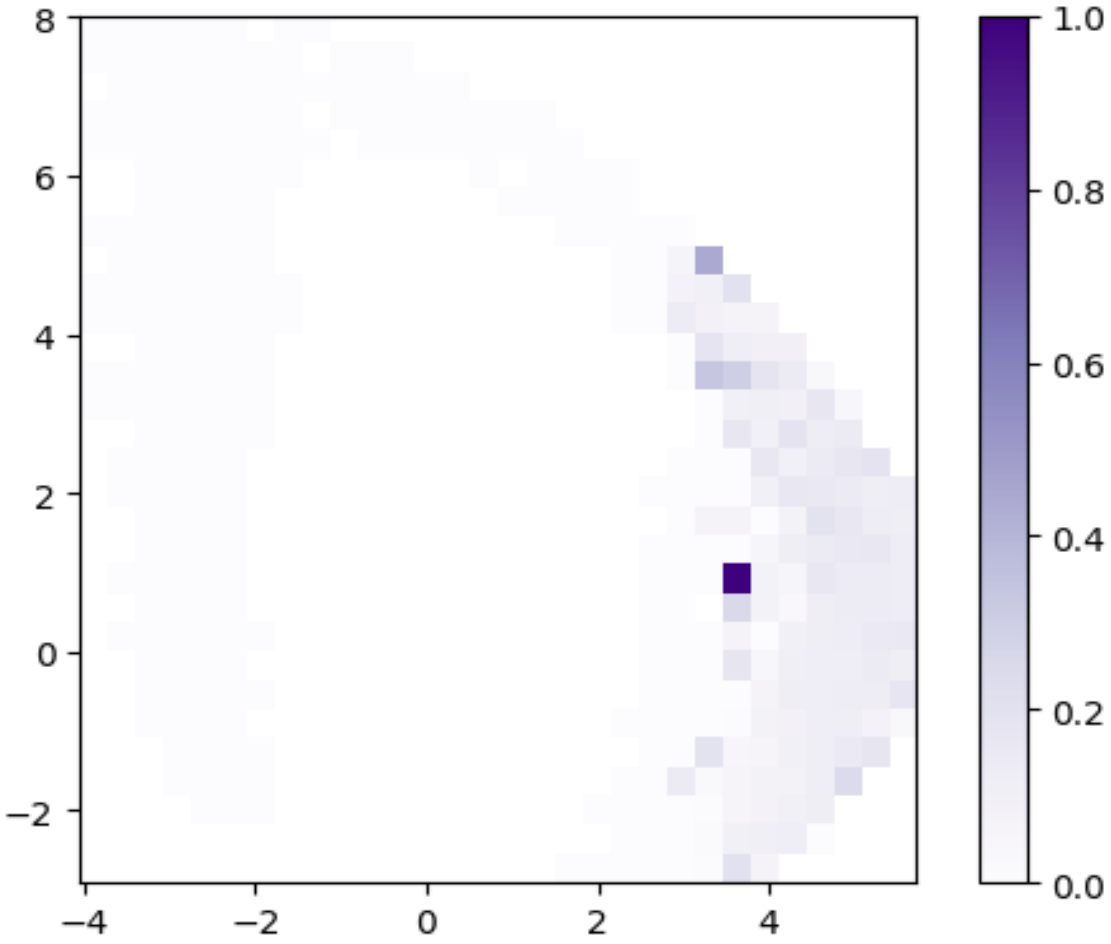

Figure 7-7: Length of Stay Visualization with PCA (2-3 days)

Darker colors correspond to higher percentages of the remaining length of stays between 2 and 3 days.

Patients with length of stay between 2 and 3 days are mostly clustered together in a single high density block.

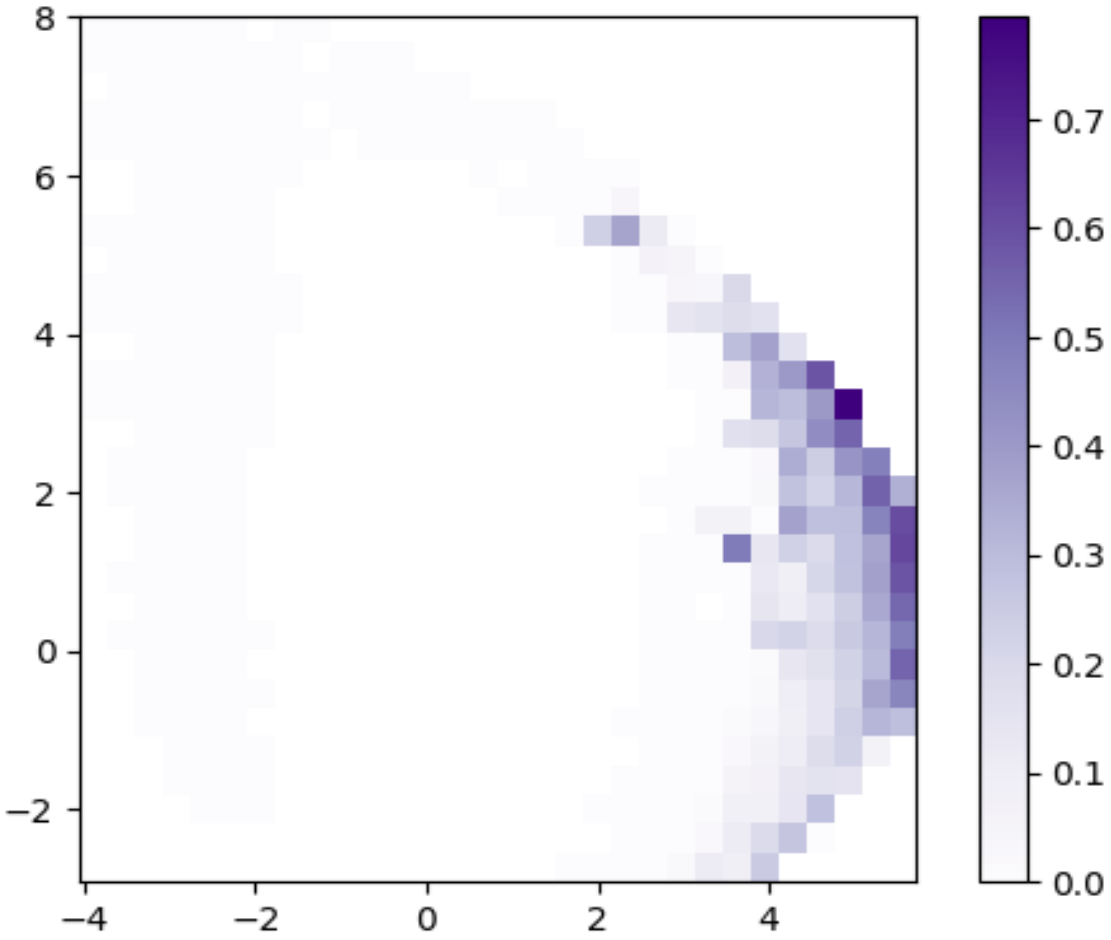

Figure 7-8: Length of Stay Visualization with PCA ($\geq$ 3 days)

Darker colors correspond to higher percentages of the remaining length of stays of $\geq$ 3 days.

Patients with length of stay greater than 3 days are clustered together in the right side of the graph in high to medium density high-density blocks. We see that block density (i.e. percentage of extended ICU stays) increases as we move along the x-axis, which is consistent with length of stay increasing along the x-axis.

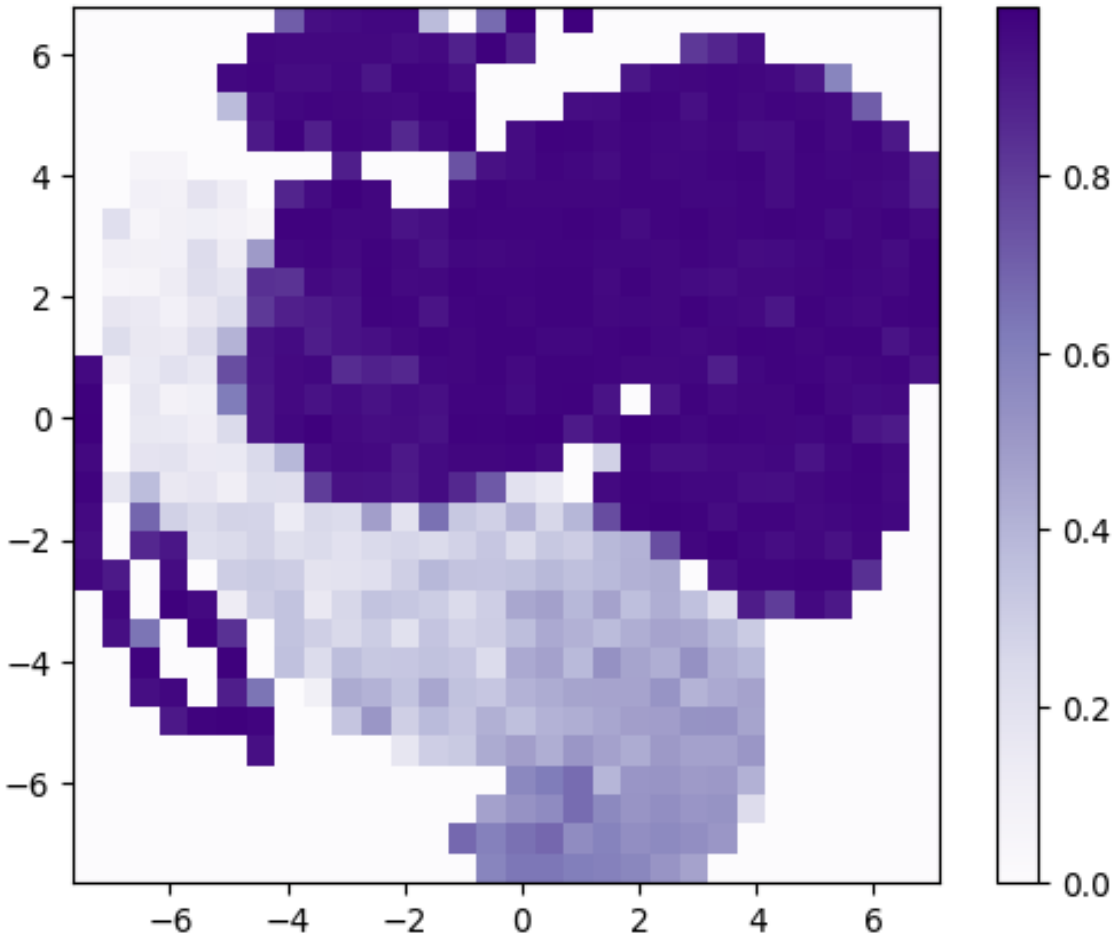

Figure 7-9: Length of Stay Visualization with t-SNE (< 24 hours)

Darker colors correspond to higher percentages of the remaining length of stays of < 24 hours.

Patients with length of stay less than 24 hours are clustered together in high-density blocks the upper and right portions of the graph or along the left side of the graph. As most patients have lengths of stay less than 24 hours, it is not surprising to see than much of the graph consists of high density blocks.

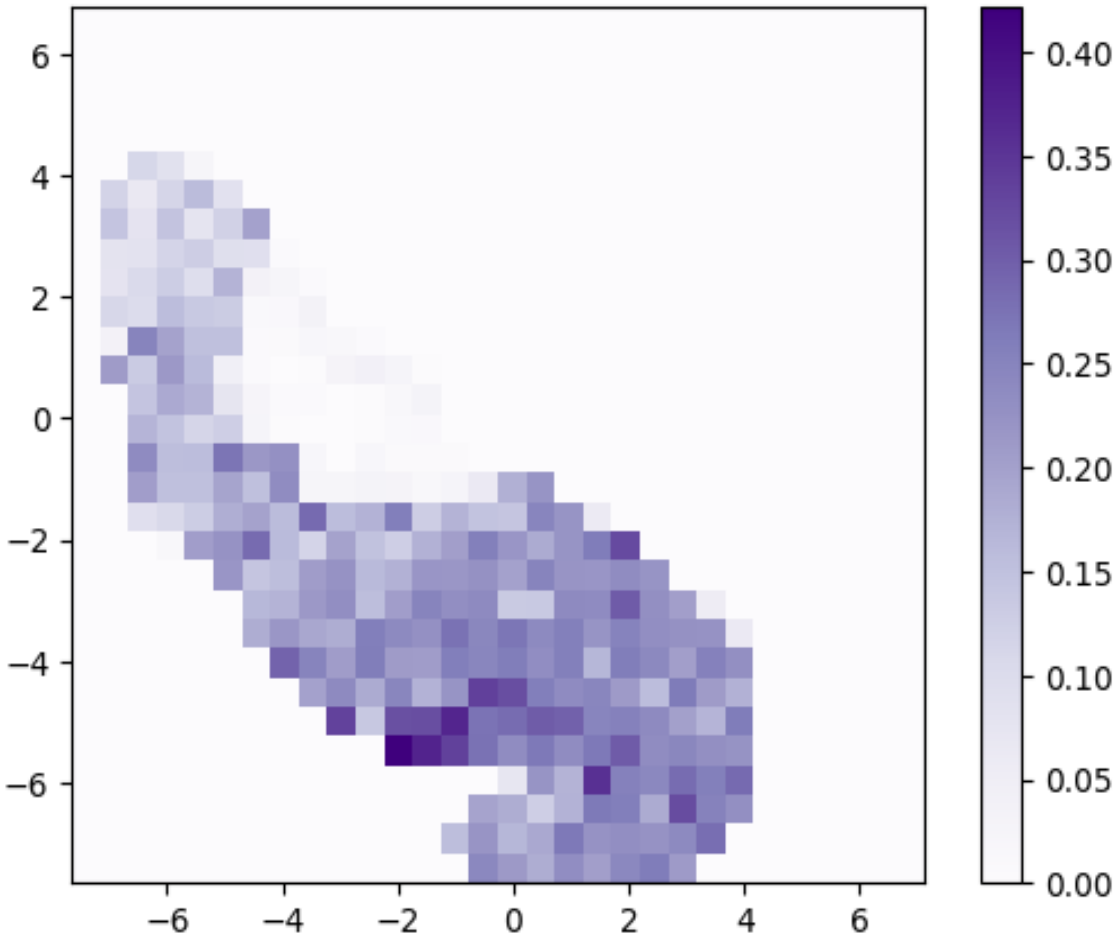

Figure 7-10: Length of Stay Visualization with t-SNE (1-2 days)

Darker colors correspond to higher percentages of the remaining length of stays between 1 and 2 days.

We see a general area of medium density blocks for lengths of stay between 1 and 2 days, corresponding to the area of low density for length of stays less than 24 hours. Block density seems to generally decrease along the y-axis, suggesting an increase in length of stay along the y-axis for patients with lengths of stay longer than a day.

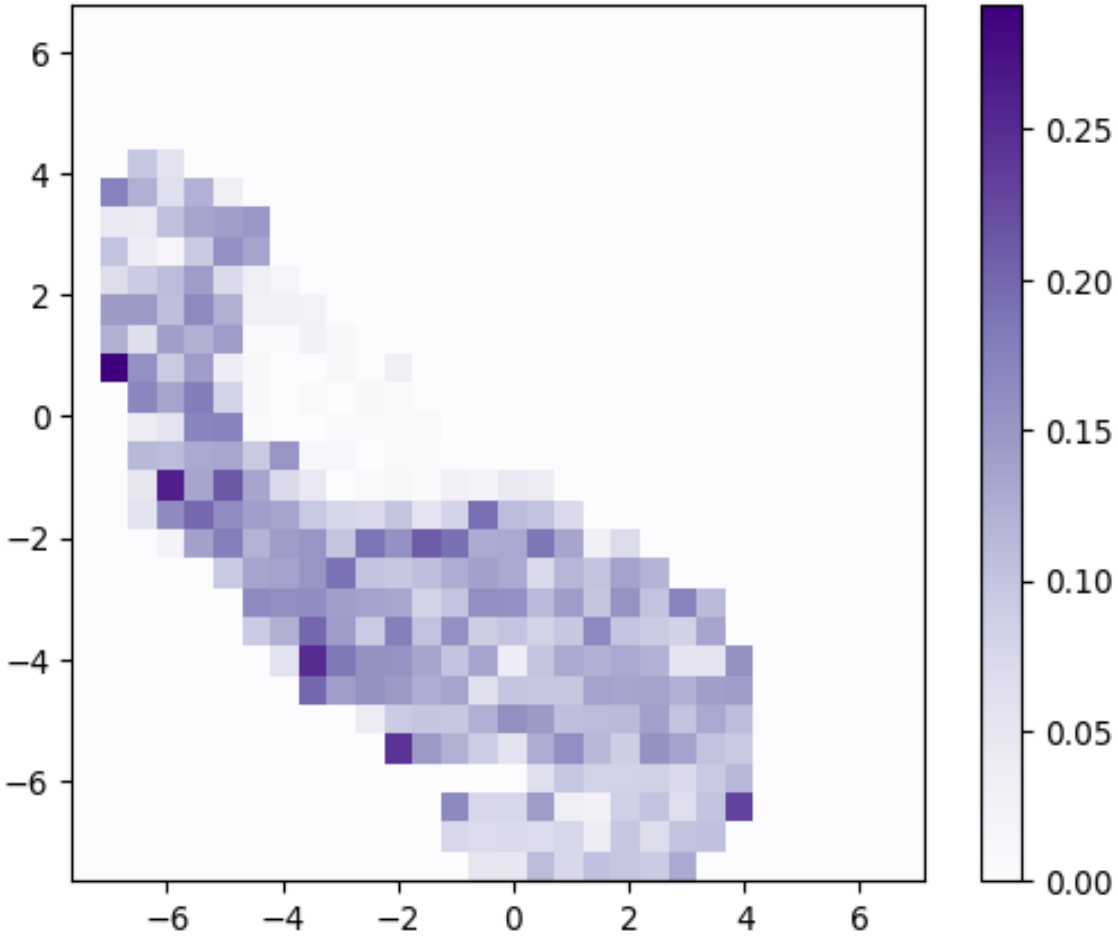

Figure 7-11: Length of Stay Visualization with t-SNE (2-3 days)

Darker colors correspond to higher percentages of the remaining length of stays between 2 and 3 days.

We see a general area of low to medium density blocks in the left portion of the graph for lengths of stay between 2 and 3 days, corresponding to the area of low density for length of stays less than 24 hours.

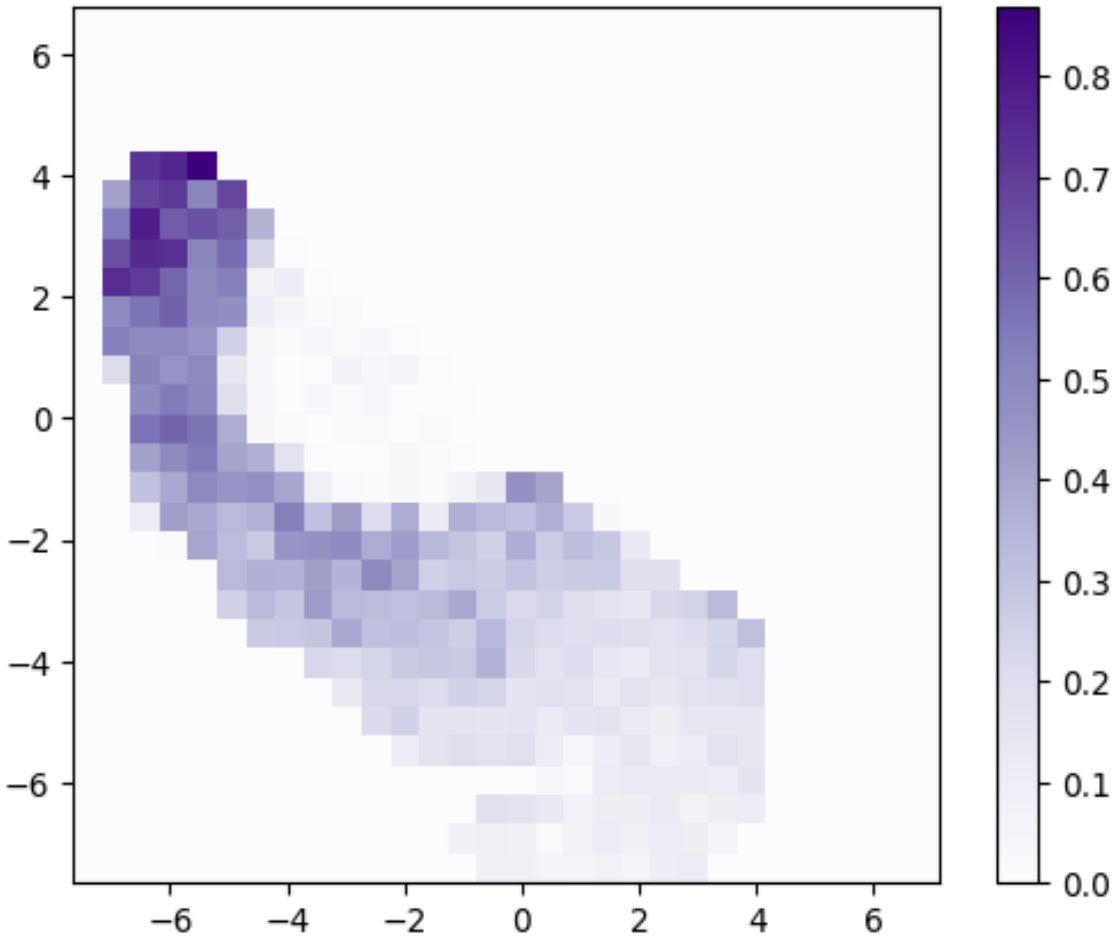

Figure 7-12: Length of Stay Visualization with t-SNE ($\geq$ 3 days)

Darker colors correspond to higher percentages of the remaining length of stays of $\geq$ 3 days.

Lengths of stay 3 days or longer are clustered together in high density blocks in the upper left portion of the graph. In general, block density decreases along the x-axis and increases along the y-axis, consistent with an increase of length of stay as one moves along those directions towards the upper left.

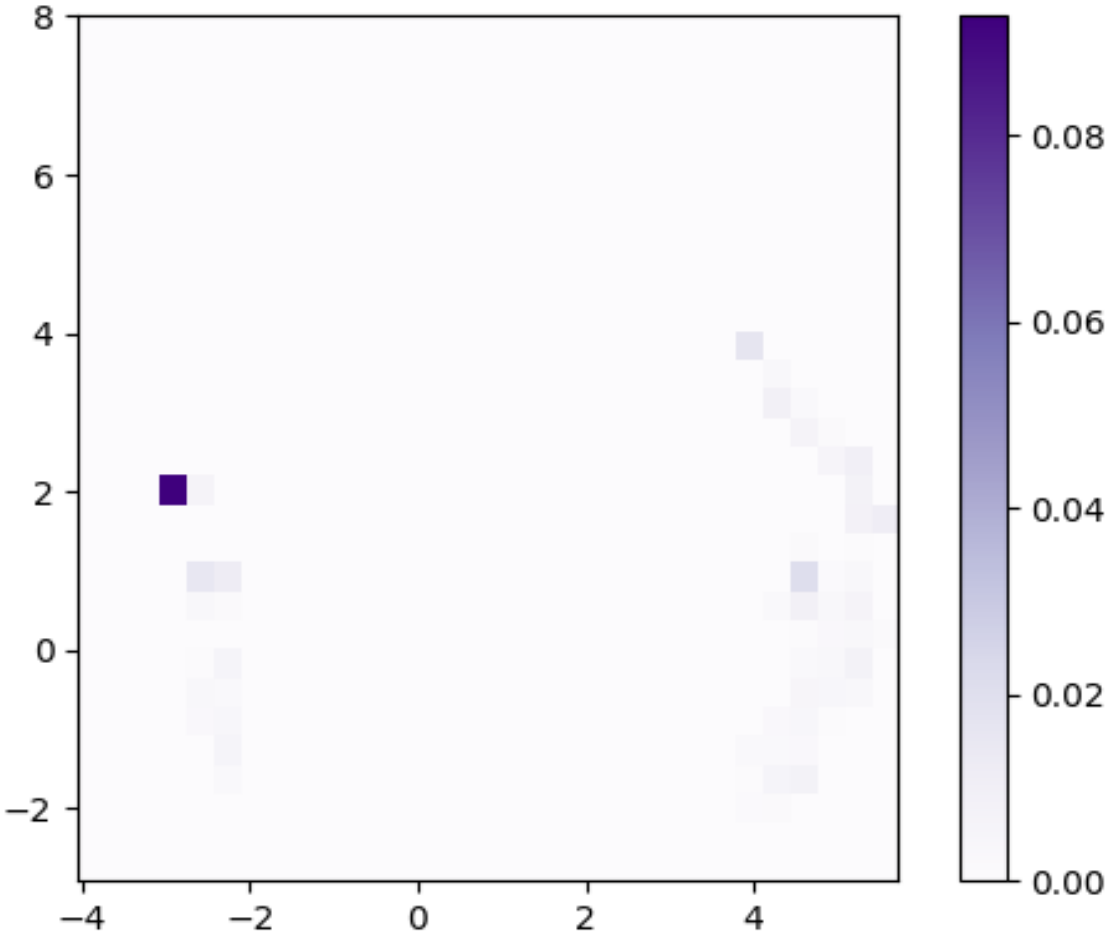

Figure 7-13: Phenotype Visualization with PCA (Acute and unspecified renal failure)

Darker colors correspond to higher percentages of acute and unspecified renal failure.

We see that most blocks have a density of 0. Of the blocks with nonzero density, the highest density block has a significantly higher relative occurrence of acute and unspecified renal failure than any of the other blocks.

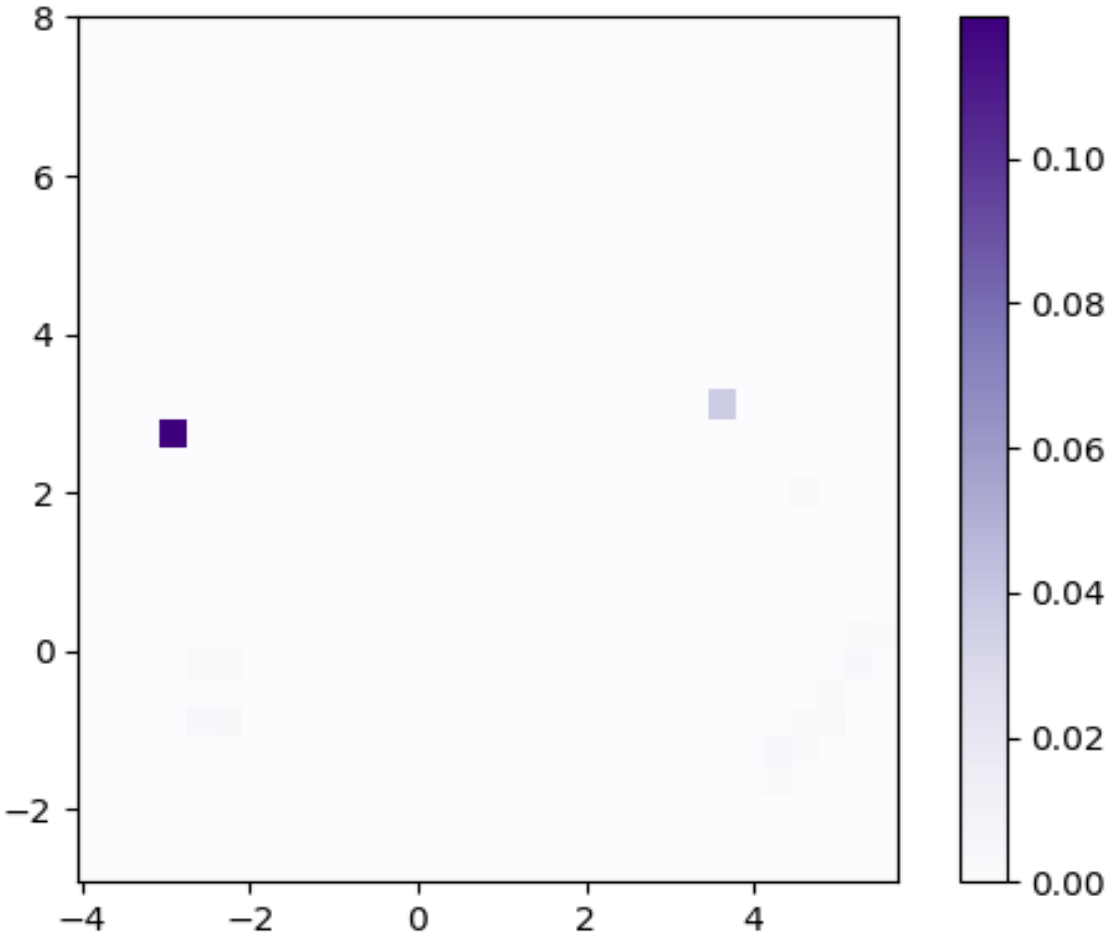

Figure 7-14: Phenotype Visualization with PCA (Chronic obstructive pulmonary disease and bronchiectasis)

Darker colors correspond to higher percentages of chronic obstructive pulmonary disease and bronchiectasis.

There are less than 10 blocks with nonzero density. Most of these have density close to zero; the highest density block has a significantly higher relative occurrence of chronic obstructive pulmonary disease and bronchiectasis than any of the other blocks.

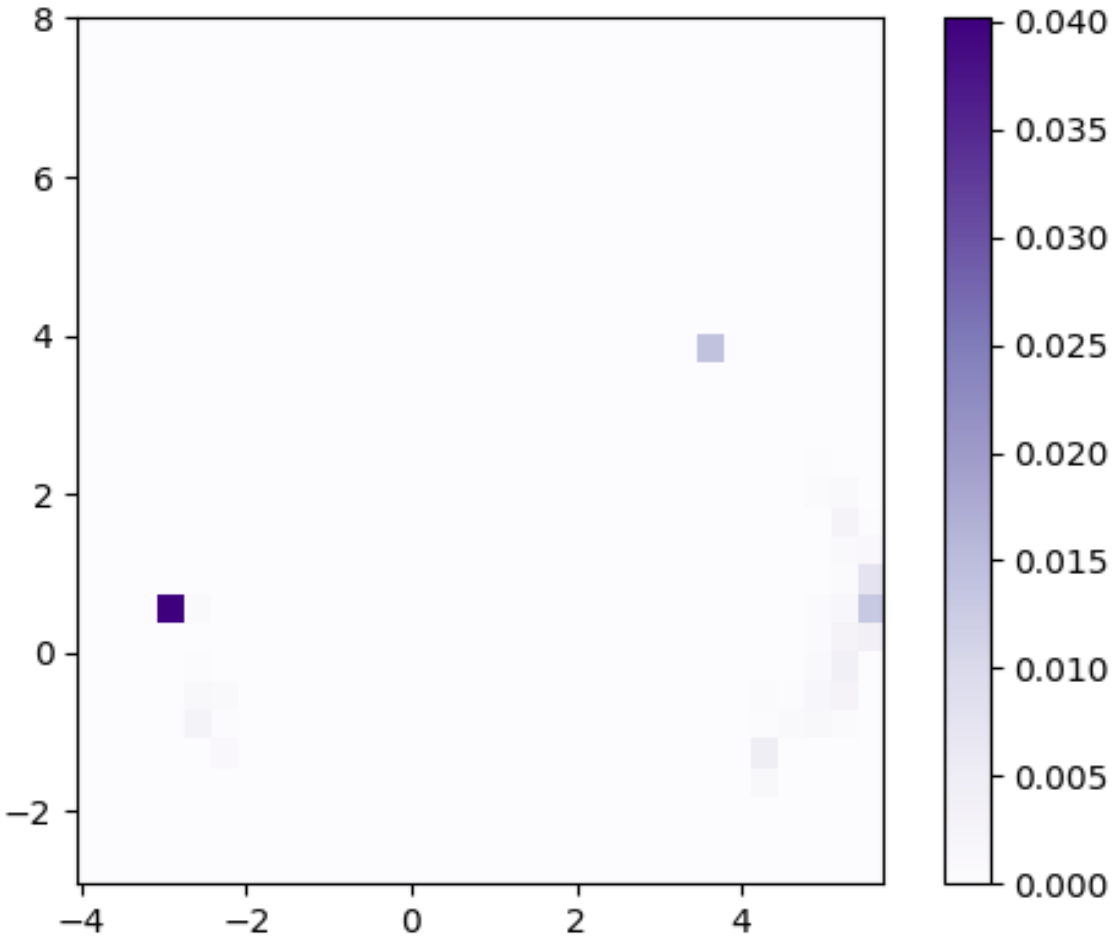

Figure 7-15: Phenotype Visualization with PCA (Essential hypertension)

Darker colors correspond to higher percentages of essential hypertension.

Again we see that most blocks have a density of 0. Of the blocks with nonzero density, the highest density block has a significantly higher relative occurrence of essential hypertension than any of the other blocks.

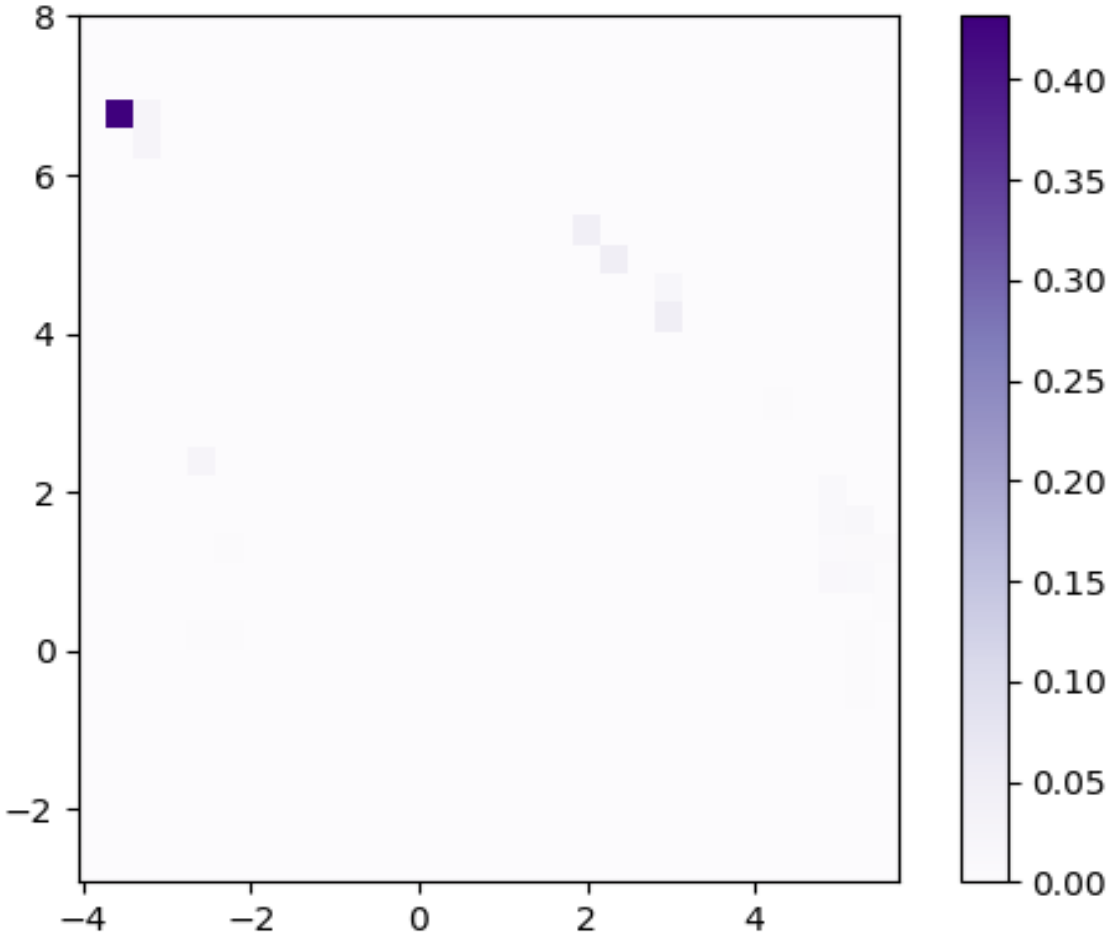

Figure 7-16: Phenotype Visualization with PCA (Pneumonia)

Darker colors correspond to higher percentages of pneumonia.

Again we see that most blocks have a density of 0. Of the patients diagnosed with pneumonia, most of them are clustered in a single medium-density block.

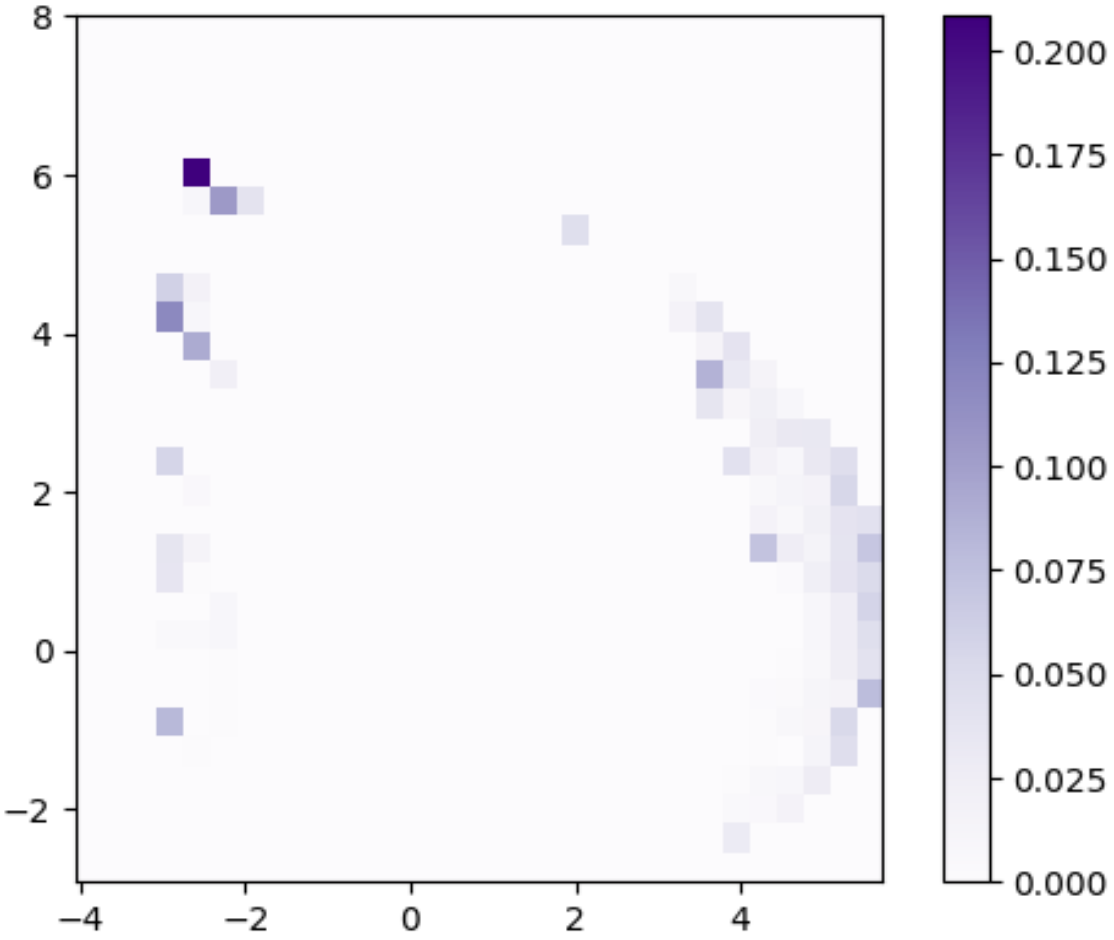

Figure 7-17: Phenotype Visualization with PCA (Respiratory failure)

Darker colors correspond to higher percentages of respiratory failure.

Again we see that most blocks have a density of 0. Most patients with respiratory failure are clustered near each other in low or medium density blocks in a line along either the left or right side of the graph.

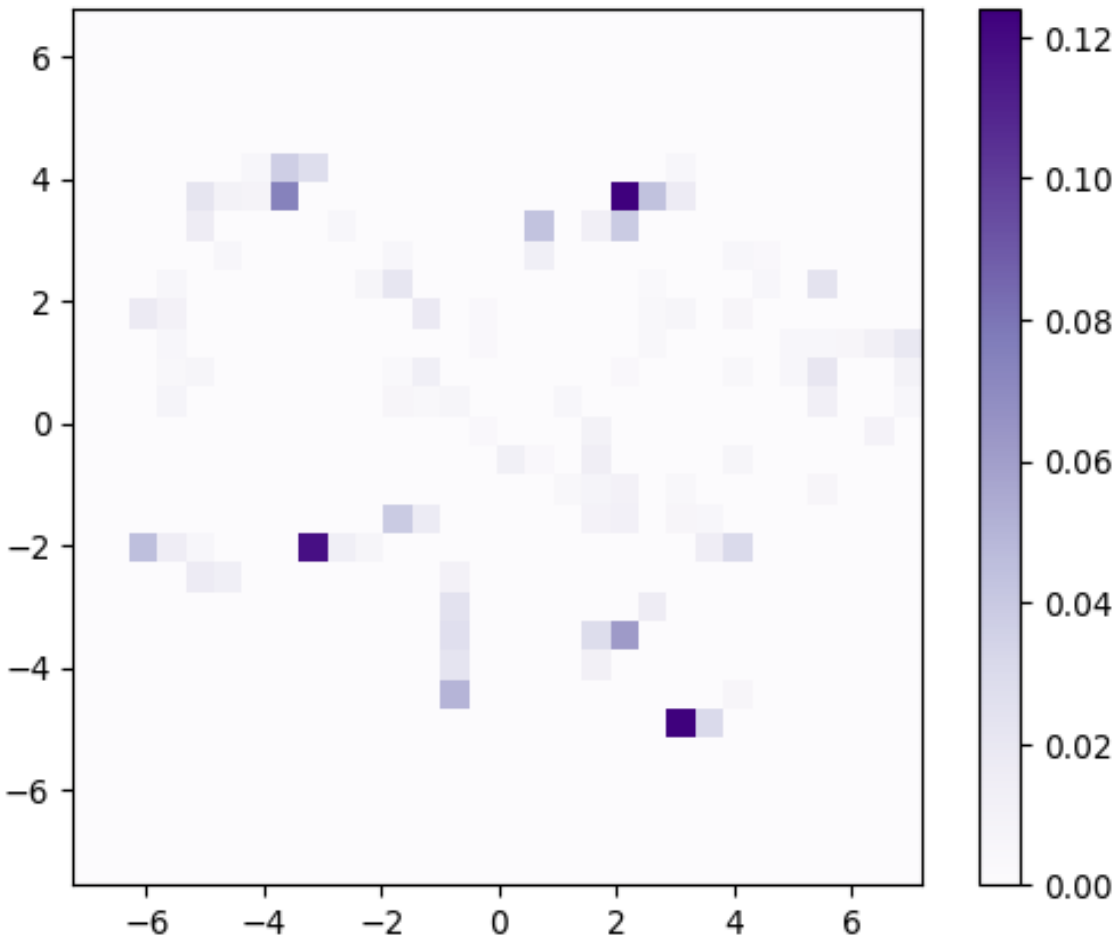

Figure 7-18: Phenotype Visualization with t-SNE (Acute and unspecified renal failure)

Darker colors correspond to higher percentages of acute and unspecified renal failure.

The blocks are low density and relatively spread out, with no real pattern in spatial location. In general we noticed that t-SNE visualizations were worse than those for PCA for phenotypes.

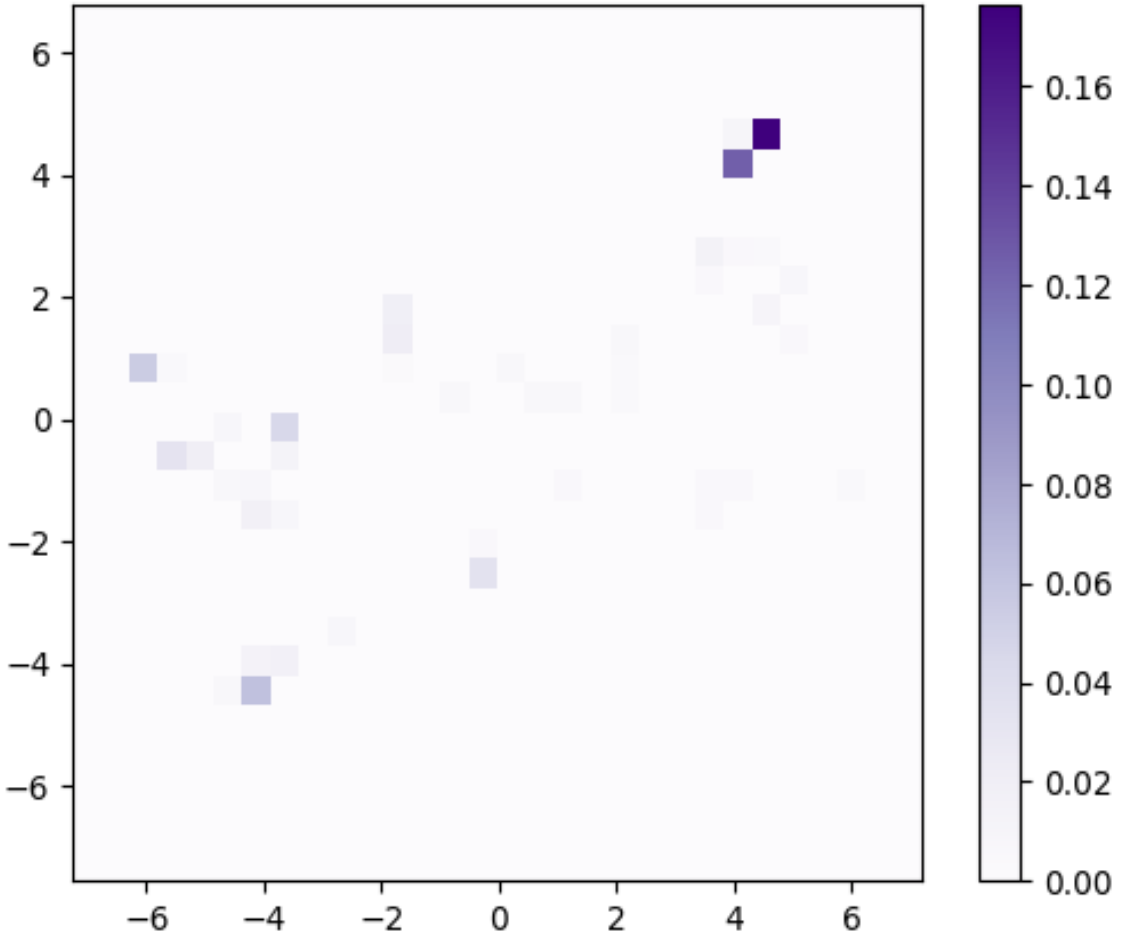

Figure 7-19: Phenotype Visualization with t-SNE (Chronic obstructive pulmonary disease and bronchiectasis)

Darker colors correspond to higher percentages of chronic obstructive pulmonary disease and bronchiectasis.

The blocks are low density and relatively spread out, with two blocks of relatively higher density located in the upper right portion of the graph.

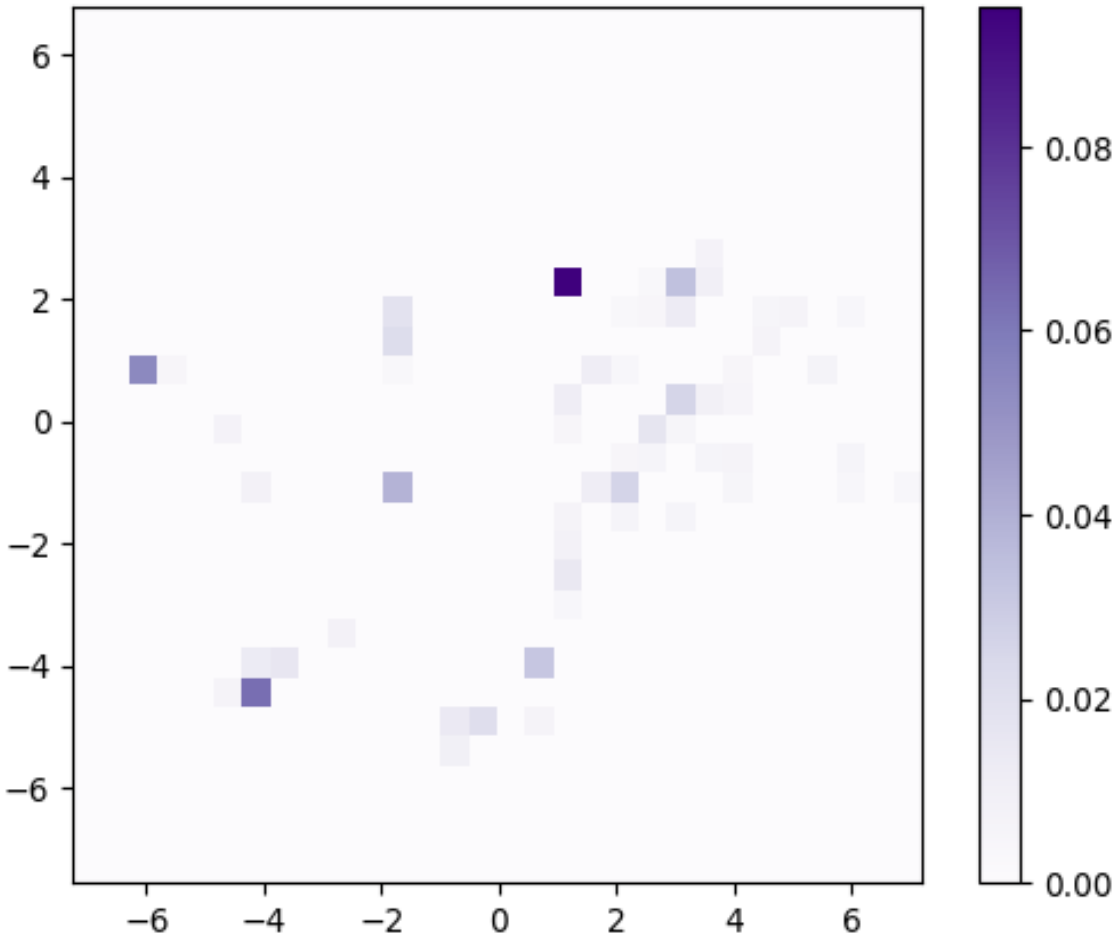

Figure 7-20: Phenotype Visualization with t-SNE (Essential hypertension)

Darker colors correspond to higher percentages of essential hypertension.

Again we see the blocks are low density and relatively spread out, with no significant pattern in spatial location.

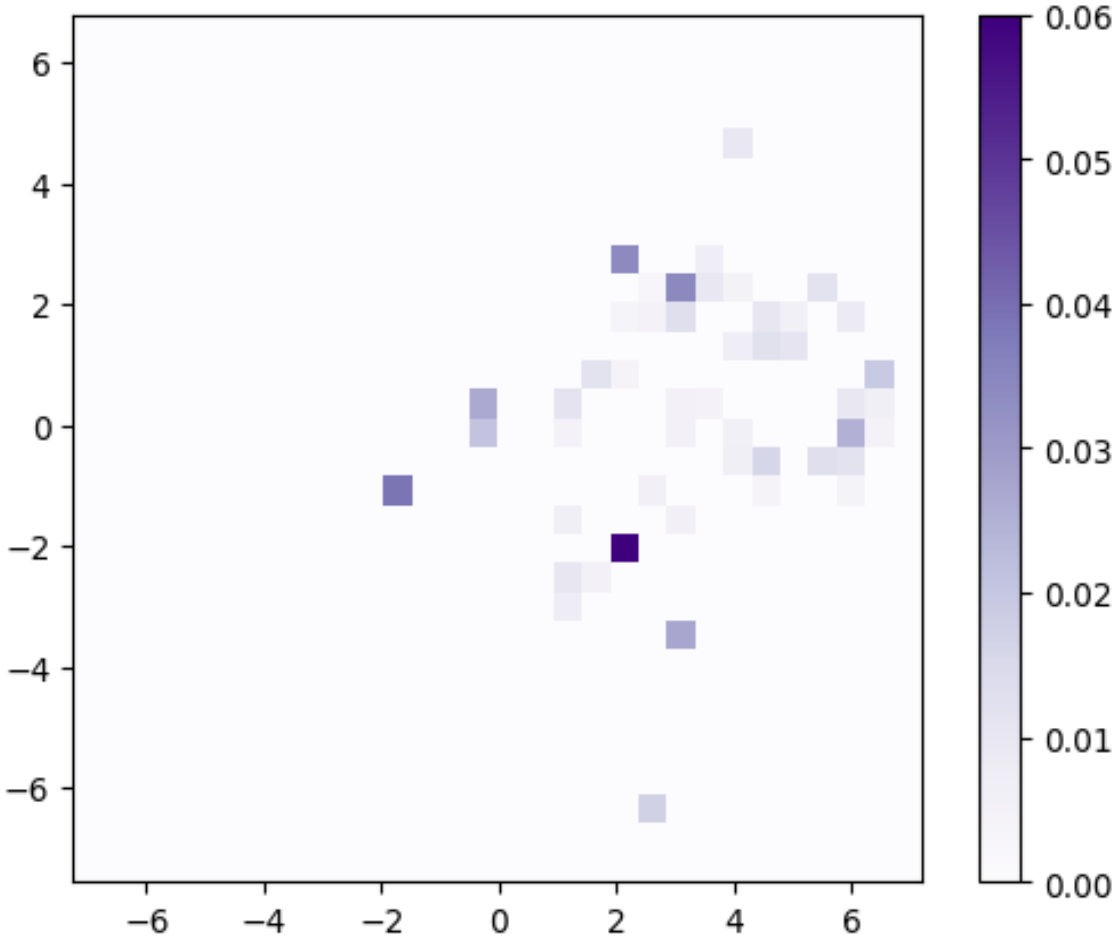

Figure 7-21: Phenotype Visualization with t-SNE (Pneumonia)

Darker colors correspond to higher percentages of pneumonia.

All of the blocks with nonzero density still have low prevelance of pneumonia, but they tend to cluster near the upper and right portions of the graph.

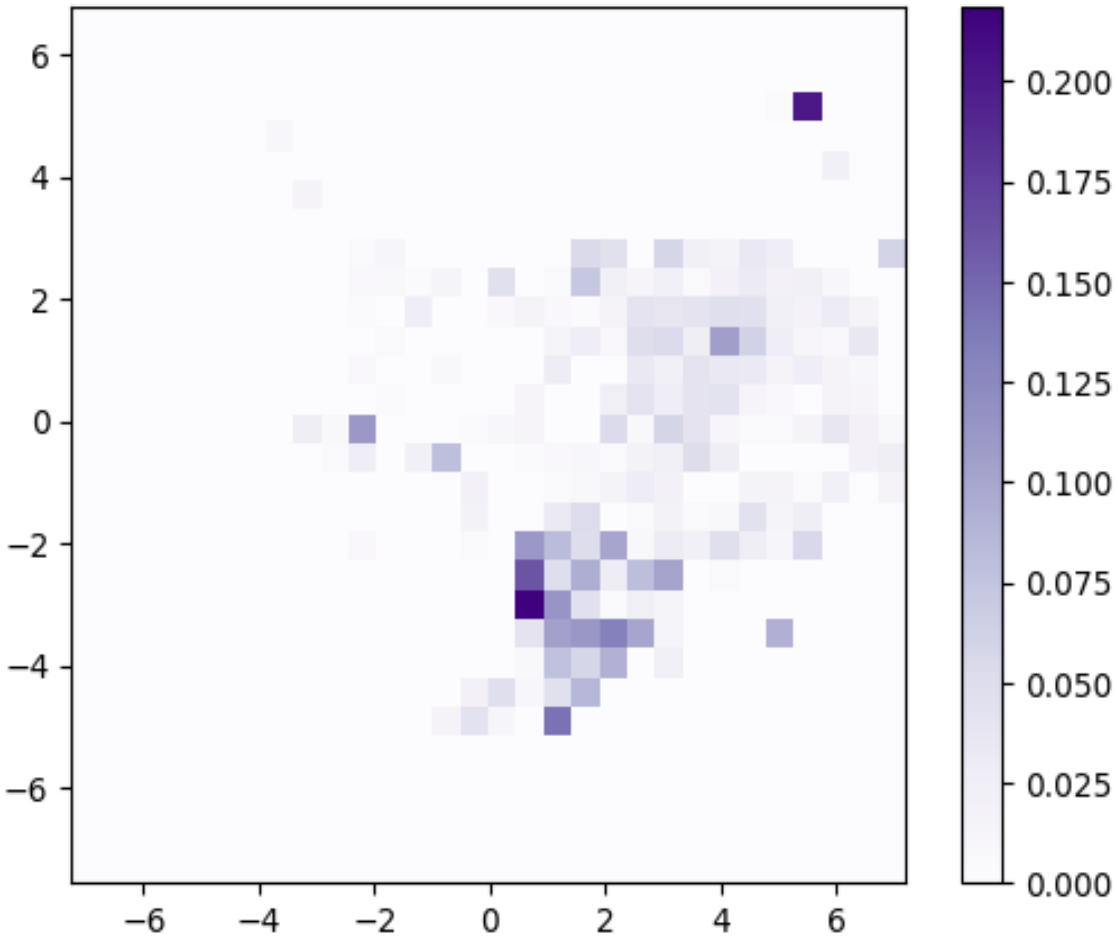

Figure 7-22: Phenotype Visualization with t-SNE (Respiratory failure)

Darker colors correspond to higher percentages of respiratory failure.

Most of the medium density blocks are clustered together towards the bottom-center of the graph.

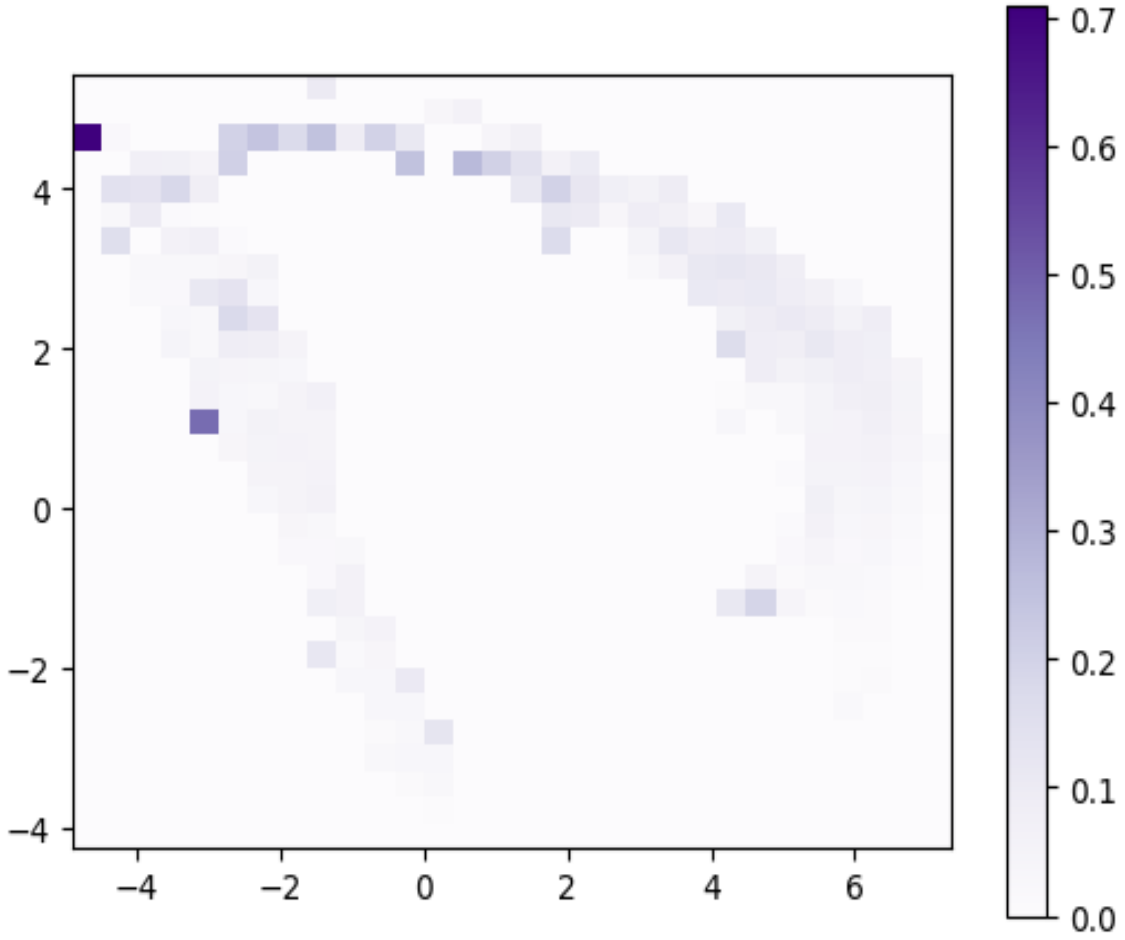

Figure 7-23: Sepsis Visualization with PCA

Darker colors correspond to higher percentages of patients with a admission diagnosis of sepsis.

Most of the nonzero blocks have low density of sepsis patients. However, we find two blocks in the left side of the graph with much higher density. Patients in these two blocks are significantly more likely to be sepsis patients.

Chapter 8

Caduceus

To facilitate use of our learned representations, particularly for patient similarity, a Python module has been released called `Caduceus`. This module is open source and publicly available at `http://github.com/ioguntol/caduceus`. Code is included for the architectures of both of our Transformer-based models. Also included is a pre-trained version of our autoregressive Transformer model for extracting deep patient representations within the module. In addition, the module contains a `PatientFinder` class to enable comparing new sequences of clinical data against a store of previous records. Provided below is the basic documentation for `PatientFinder` at the time this document was written.

8.1 PatientFinder

```
class caduceus.patient_finder.PatientFinder(
    n_pca_components=32, n_knn_neighbors=10)
```

Parameters:

- `n_pca_components: int, optional (default = 32)`

  Default number of principal components to keep when performing di-

mensionality reduction on learning representations with PCA

- `n_knn_neighbors: int, optional (default = 10)`

  Default number of neighbors to query during nearest neighbor search with learning representations

8.2 Methods:

```
fit_transform(data, patient_ids=None)
```

Use the provided patient data as a base for comparing new patient clinical sequences against. Data is first transformed by the deep model, then is further transformed by a new PCA model fit to this data, and is stored as a population matrix.

Parameters:

- `data: list`

  List of numpy nd-arrays, where each array has shape (`num_timesteps`, 17). Each row should contain the 17 variables in the order listed in Table 5.1. Missing features should have the value `np.nan`. Timesteps are expected to be hour-increments, and should start from the time of ICU admission.

- `patient_ids: list, optional (default = None)`

  A list of ids corresponding to each patient in `data`. If provided, this list should have the same length as `data`. When `patient_ids == None`, the patients are incrementally numbered starting from 0 in the order they appear in the list `data`.

```
find_similar(
    data, num_similar=self.n_knn_neighbors,
    return_distance=False)
```

Finds patient data points from the population matrix similar to the new ones provided via k-nearest neighbors with Euclidean distance on deep-PCA-transformed representations. Returns patient ids and corresponding specific timesteps. Optionally returns distances to the neighbors of each point.

Parameters:

- `data: list`

  List of numpy nd-arrays, where each array has shape (`num_timesteps`, 17). Each row should contain the 17 variables in the order listed in Table 1. Missing features should have the value `np.nan`. Timesteps are expected to be hour-increments, and should start from the time of ICU admission.

- `num_similar: int, optional (default = self.n_knn_neighbors)`

  The number of similar patient data points to find.

- `return_distance: boolean, optional (default = False)`

  If `True`, distances will be returned.

Returns:

- `patient_ids: list`

  A list of patient ids for the selected similar patient data points

- `patient_offsets: list`

  A list of timestep offsets for the selected similar patient data points

- `distances: list`

  Array representing the lengths to points, only present if `return_distance == True`

Chapter 9

Conclusions

9.1 Contributions

The primary contribution of this thesis is a method of extracting deep robust representations of teleICU clinical data. As of now, deep learning techniques are scarce in data analytics for teleICU data, although recent work has applied them to the general ICU setting. Inspired by the recent machine learning literature that utilizes a new architecture called the Transformer, we employ this architecture with teleICU clinical time-series data to learn patient representations.

9.1.1 Patient Outcome Prediction

The utility of these learned representations is demonstrated for various downstream predictions of patient outcomes. The Transformer-based models consistently outperform standard LSTM baselines in predicting in-hospital mortality, physiologic decompensation, and length and stay, with comparable or superior performance on the task of phenotype classification. In general the deep models significantly outperformed logistic regression baselines; this has been demonstrated by multiple sources for the general ICU [12, 39], and similar results were found on the teleICU clinical data used

in this thesis.

While the Transformer-based models performed the best out of the deep models, the Transformer model trained with added autoregressive loss generally outperformed even the other multitask-only Transformer model. The autoregressive loss also improved model calibration on mortality and decompensation tasks over the multitask Transformer, while still maintaining superior performance on patient outcome task metrics.

9.1.2 Patient Similarity Analysis

Furthermore, we showed various 2D visualizations of patient time-series representations learned by the autoregressive Transformer model using PCA and t-SNE, which tended to group similar patients together spatially across various different characteristics, often in high density. PCA especially produced visualizations indicating a high level of structure in the space of patient representations with respect to mortality, length of stay, and phenotypic patient characteristics. This suggests that our representations are useful in indicating general patient similarity, and can be used as a heuristic as such.

9.1.3 Open-Sourced Code

Finally, we open source the code for our model architectures, the weights for a pretrained model, and a Python module intended to facilitate use of these representations to compare patient data, and as a starting point for further research.

9.2 Future Work

One area this work can be expanded on in the future is with patient similarity analysis. This thesis uses low-dimensional approximations of Euclidean distance in learned

patient representation space such as PCA to demonstrate the utility of representations as a similarity heuristic. However this also suggests potential for their use in combination with other more complex or targeted patient similarity metrics; in this sense the Transformer models are used as a preprocessing step.

In general, the hope is that better models of patient data – and more generally better methods of modeling patient data – can enable teleICU operations in many different ways. One particularly interesting area of future work is to incorporate non-patient factors in to our models. Incorporating more hospital information such as staff size, hospital type (e.g. rural, urban, etc), or even categories of organizational strategies/systems into powerful deep models may feed insights as to which groups of patients might benefit the most from the teleICU.

Chapter 10

References

[1] David Reinsel, John Gantz, and John Rydning. Data age 2025: The evolution of data to life-critical. *IDC White Paper*, 2017.

[2] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24, 2019.

[3] Tom J Pollard, Alistair EW Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. The eicu collaborative research database, a freely available multi-center database for critical care research. *Scientific data*, 5, 2018.

[4] Christian Becker, William H Frishman, and Corey Scurlock. Telemedicine and tele-icu: the evolution and differentiation of a new medical field. *The American journal of medicine*, 129(12):e333–e334, 2016.

[5] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[6] Alfredo Vellido, Vicent Ribas, Carles Morales, Adolfo Ruiz Sanmartín, and Juan Carlos Ruiz Rodríguez. Machine learning in critical care: state-of-the-art and a sepsis case study. *Biomedical engineering online*, 17(1):135, 2018.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[11] Benjamin Shickel, Patrick James Tighe, Azra Bihorac, and Parisa Rashidi. Deep ehr: a survey of recent advances in deep learning techniques for electronic health record (ehr) analysis. *IEEE journal of biomedical and health informatics*, 22(5):1589–1604, 2017.

[12] Sanjay Purushotham, Chuizheng Meng, Zhengping Che, and Yan Liu. Benchmarking deep learning models on large healthcare datasets. *Journal of biomedical informatics*, 83:112–134, 2018.

[13] Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1(1):18, 2018.

[14] Harini Suresh, Nathan Hunt, Alistair Johnson, Leo Anthony Celi, Peter Szolovits, and Marzyeh Ghassemi. Clinical intervention prediction and understanding with deep neural networks. In *Machine Learning for Healthcare Conference*, pages 322–337, 2017.

[15] Riccardo Miotto, Li Li, Brian A Kidd, and Joel T Dudley. Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports*, 6:26094, 2016.

[16] Truyen Tran, Tu Dinh Nguyen, Dinh Phung, and Svetha Venkatesh. Learning vector representation of medical objects via emr-driven nonnegative restricted boltzmann machines (enrbm). *Journal of biomedical informatics*, 54:96–105, 2015.

[17] Znaonui Liang, Gang Zhang, Jimmy Xiangji Huang, and Qmming Vivian Hu. Deep learning for healthcare decision making with emrs. In *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 556–559. IEEE, 2014.

[18] Zhengping Che, Sanjay Purushotham, Robinder Khemani, and Yan Liu. Interpretable deep models for icu outcome prediction. In *AMIA Annual Symposium Proceedings*, volume 2016, page 371. American Medical Informatics Association, 2016.

[19] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3504–3512. Curran Associates, Inc., 2016.

[20] Sebastiano Barbieri, Oscar Perez-Concha, Sradha Kotwal, Martin Gallagher, Angus Ritchie, and Louisa Jorm. A deep representation of longitudinal emr data

used for predicting readmission to the icu and describing patients-at-risk. *CoRR*, 2019.

[21] Chongyu Zhou, Yao Jia, and Mehul Motani. Optimizing autoencoders for learning deep representations from health data. *IEEE journal of biomedical and health informatics*, 23(1):103–111, 2018.

[22] Liqi Lei, Yangming Zhou, Jie Zhai, Le Zhang, Zhijia Fang, Ping He, and Ju Gao. An effective patient representation learning for time-series prediction tasks based on ehrs. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 885–892. IEEE, 2018.

[23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[24] Jinghe Zhang, Kamran Kowsari, James H Harrison, Jennifer M Lobo, and Laura E Barnes. Patient2vec: A personalized interpretable deep representation of the longitudinal electronic health record. *IEEE Access*, 6:65333–65346, 2018.

[25] Deepak A Kaji, John R Zech, Jun S Kim, Samuel K Cho, Neha S Dangayach, Anthony B Costa, and Eric K Oermann. An attention based deep learning model of clinical events in the intensive care unit. *PloS one*, 14(2):e0211057, 2019.

[26] Huan Song, Deepta Rajan, Jayaraman J Thiagarajan, and Andreas Spanias. Attend and diagnose: Clinical time series analysis using attention models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[27] Ping Zhang, Fei Wang, Jianying Hu, and Robert Sorrentino. Towards personalized medicine: leveraging patient similarity and drug similarity analytics. *AMIA Summits on Translational Science Proceedings*, 2014:132, 2014.

[28] Shraddha Pai and Gary D Bader. Patient similarity networks for precision medicine. *Journal of molecular biology*, 430(18):2924–2938, 2018.

[29] Zihao Zhu, Changchang Yin, Buyue Qian, Yu Cheng, Jishang Wei, and Fei Wang. Measuring patient similarities via a deep architecture with medical concept embedding. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 749–758. IEEE, 2016.

[30] Qiuling Suo, Fenglong Ma, Ye Yuan, Mengdi Huai, Weida Zhong, Aidong Zhang, and Jing Gao. Personalized disease prediction using a cnn-based similarity learning method. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 811–816. IEEE, 2017.

[31] Qiuling Suo, Fenglong Ma, Ye Yuan, Mengdi Huai, Weida Zhong, Jing Gao, and Aidong Zhang. Deep patient similarity learning for personalized healthcare. *IEEE transactions on nanobioscience*, 17(3):219–227, 2018.

[32] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[33] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111, 2014.

[34] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[35] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 1(8), 2019.

[36] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[37] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[38] Zachary Chase Lipton, David C. Kale, Charles Elkan, and Randall C. Wetzel. Learning to diagnose with LSTM recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[39] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific data*, 6(1):96, 2019.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[41] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[42] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[43] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *The 9th ISCA*

Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016, page 125, 2016.

[44] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[45] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[46] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, 2016.

[47] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[48] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.