Deep Learning Solutions to Computational Phenotyping in Health Care

Zhengping Che
Department of Computer Science
University of Southern California
Los Angeles, CA, USA 90089
zche@usc.edu

Yan Liu
Department of Computer Science
University of Southern California
Los Angeles, CA, USA 90089
yanliu.cs@usc.edu

***Abstract*—Exponential growth in electronic health record (EHR) data has resulted in new opportunities and urgent needs to discover meaningful data-driven representations and patterns of diseases, i.e., *computational phenotyping*. Recent success and development of deep learning provides promising solutions to the problem of prediction and feature discovery tasks, while lots of challenges still remain and prevent people from applying standard deep learning models directly. In this paper, we discussed three key challenges in this field: how to deal with missing data, how to build scalable models, and how to get interpretations of features and models. We proposed novel and effective deep learning solutions to each of them respectively. All proposed solutions are evaluated on several real-world health care datasets and experimental results demonstrated their superiority over existing baselines.**

I. Introduction

The national push [1] for electronic health records (EHR) has resulted in an exponential surge in volume, detail, and availability of digital health data which offers an unprecedented opportunity to solve many difficult and important problems in health care. Clinicians are collaborating with computer scientists by using this opportunity to improve the state of health care services towards the goal of *Personalized Healthcare* [2]. One important step towards this goal is learning richer, data-driven descriptions of illness from a variety of data sources and types. This field of research, known as *computational phenotyping*, has attracted many machine learning and data mining researchers [3]–[5]. However, the recent rise of this research field with more available data and new applications has also introduced several challenges which have not been answered well. Among these challenges, handling missingness in health care data, building predictive model with scalability, and interpreting learned features and models are three most urgent and important ones to be solved.

Properly handling and even effectively exploiting missingness in real-world health care data is very important in health care domain. Unlike other data sources, medical data such as EHR often inevitably carry missing observations due to various reasons, such as medical events, cost saving, anomalies, inconvenience. It has been noted that these missing values are usually *informative missingness* [6], i.e., the missing values and patterns provide rich information about target labels in supervised learning tasks (e.g, time series classification). A variety of methods have been developed to fill in the missing values [7]–[9], but they usually result in a two-step process where imputations are disparate from prediction models and missing patterns are not effectively explored, thus leading to suboptimal analyses and predictions [10]. Recent works [11]–[13] tried to handle missingness in recurrent neural networks (RNNs) by concatenating missing entries or timestamps with the input or performing simple imputations. However, there have not been works which model missing patterns into a systematically modified RNN structure for time series classification problems. We develop a novel deep learning model based on Gated Recurrent Units (GRU) [14], namely **GRU-D**, to effectively exploit *masking* and *time interval*, which are two representations of informative missingness patterns [15]. Masking informs the model which inputs are observed (or missing), while time interval encapsulates the input observation patterns. Our model captures the observations and their dependencies by applying masking and time interval (using a decay term) to the inputs and network states of a GRU cell, and jointly train all model components using back-propagation. Thus, our model not only captures the long-term temporal dependencies of time series observations but also utilizes the missing patterns to improve the prediction results.

Another major challenge comes from the requirement of training scalable model which utilize a large and increasing amount of data. First, the daily increasing volume of health care data raises the critical problem on finding an efficient schema to build models with newly coming data. Second and more important, while plenties of EHR data are available, only a limited amount of them are for one specific disease or one single patient. Leveraging the advantages of the big data for all cases to small data with limited and imbalanced labels would therefore become quite useful. We propose a general deep learning framework which has a more efficient training process and achieves better prediction performance in this situation [16]. We formulate a prior-based regularization framework to guide the training of multi-label neural networks using medical ontologies and other structured knowledge. To be more specific, *graph Laplacian* [17] priors are applied on the prediction layer of the neural networks to incorporate relational information from domain knowledge or training data. We also propose an efficient incremental training pro-

cedure for building a series of neural networks that detect physiologic patterns of increasing length by utilizing existing neural networks to initialize the a new neural net designed to detect longer temporal patterns or newly included features.

Furthermore, in health care domain, model and feature interpretability is not only important but also necessary, since primary care providers, physicians and clinical experts are increasingly depending on the new data-driven health care technologies to help them in patient monitoring and decision-making. A good interpretable model is shown to result in faster adoptability among clinical staffs and better quality of patient care [18], [19]. Even though powerful, deep learning models (usually with millions of model parameters) are difficult to interpret. On the other hand, decision trees [20], due to their easy interpretability, have been quite successfully employed in health care domain [21], [22], but they can easily overfit and perform poorly on large heterogeneous EHR datasets. Thus, an important question naturally arises: how can we develop novel data-driven solutions which can achieve state-of-the-art performance as deep learning models and at the same time can be easily interpreted by health care professionals and medical practitioners? Inspired by the recently developed *mimic learning* [23] and *knowledge distillation* [24] approaches, we introduce a simple yet effective knowledge-distillation approach called **interpretable mimic learning** [25], [26], to learn interpretable models with robust prediction performance as deep learning models. Our interpretable mimic learning framework uses gradient boosting trees (GBT) [27] to learn interpretable models from deep learning models. Experimental results demonstrate that our interpretable mimic learning framework can maintain state-of-the-art prediction performance of deep models and provide interpretable features and decision rules.

The rest of this paper is organized as follows: Section II introduces our *GRU-D* model for utilizing informative missingness. In section III, we describe our prior-based network and incremental training strategies to build scalable deep computational phenotyping models. The *interpretable mimic learning* framework is described in section IV. We summarize our work in section V.

II. GRU-D: Model for Missingness in Data

A. Methodology

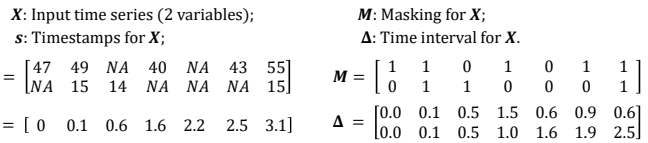


Fig. 1. An example of measurement vectors $\boldsymbol{x}_t$, time stamps s_t, masking $\boldsymbol{m}_t$, and time interval $\boldsymbol{\delta}_t$.

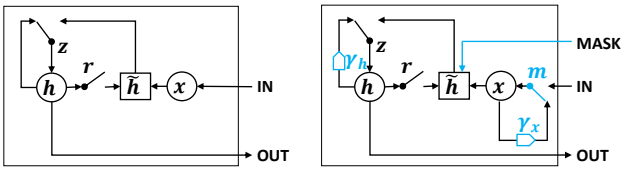
1) Notations: We first clarify the notations before introducing our models. We denote a multivariate time series with D variables of length T as $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T] \in \mathbb{R}^{D \times T}$. Let $s_t \in \mathbb{R}$ denote the time-stamp when the tth observation is obtained and we assume that the first observation is made at time-stamp 0 (i.e., $s_1 = 0$). A time series $\boldsymbol{X}$ could have missing values. We introduce the *masking* $\boldsymbol{m}_t \in \{0,1\}^D$ to denote which variables are missing at time step t. The value of masking is 1 if the variable is observed, otherwise 0. For each variable d, we also maintain the *time interval* $\delta_t^d \in \mathbb{R}$ since its last observation. An example of these notations is illustrated in Figure 1. In this paper, we are interested in the time series classification problem, where we predict the labels $l_n \in \{1, \ldots, L\}$ given the time series data $\mathcal{D}$, where $\mathcal{D} = \{(\boldsymbol{X}_n, \boldsymbol{s}_n, \boldsymbol{M}_n, \Delta_n)\}_{n=1}^N$, and $\boldsymbol{X}_n = \left[\boldsymbol{x}_1^{(n)}, \ldots, \boldsymbol{x}_{T_n}^{(n)}\right]$, $\boldsymbol{s}_n = \left[s_1^{(n)}, \ldots, s_{T_n}^{(n)}\right]$, $\boldsymbol{M}_n = \left[\boldsymbol{m}_1^{(n)}, \ldots, \boldsymbol{m}_{T_n}^{(n)}\right]$, $\Delta_n = \left[\boldsymbol{\delta}_1^{(n)}, \ldots, \boldsymbol{\delta}_{T_n}^{(n)}\right]$.

2) GRU-RNN for time series classification: We investigate the use of recurrent neural networks (RNN) for time-series classification, as their recursive formulation allow them to handle variable-length sequences naturally. We specifically consider an RNN with gated recurrent units (GRU) [14], [28] (whose block structure is shown in Figure 2(a)), but similar discussion and modifications are also valid for other RNN models such as Long Short-Term Memory (LSTM) [29].

Existing work on handling missing values lead to three possible solutions with no modification on GRU network structure. One straightforward approach is simply replacing each missing observation with the mean of the variable across the training examples. In the context of GRU, we have $x_t^d \leftarrow m_t^d x_t^d + (1 - m_t^d)\tilde{x}^d$ where $\tilde{x}^d = \sum_{n=1}^N \sum_{t=1}^{T_n} m_{t,n}^d x_{t,n}^d \Big/ \sum_{n=1}^N \sum_{t=1}^{T_n} m_{t,n}^d$. We refer to this approach as *GRU-mean*. A second approach is to exploit the temporal structure. For example, we may assume any missing value is the same as its last measurement and use forward imputation (*GRU-forward*), i.e., $x_t^d \leftarrow m_t^d x_t^d + (1 - m_t^d) x_{t'}^d$ where $t' < t$ is the last time the d-th variable was observed. Instead of explicitly imputing missing values, the third approach simply indicates which variables are missing and how long they have been missing as a part of input by concatenating the measurement, masking and time interval vectors as $\boldsymbol{x}_t^{(n)} \leftarrow \left[\boldsymbol{x}_t^{(n)}; \boldsymbol{m}_t^{(n)}; \boldsymbol{\delta}_t^{(n)}\right]$ where $\boldsymbol{x}_t^{(n)}$ can be the same as either GRU-mean or GRU-forward. We later refer to this approach as *GRU-simple*.

Several recent works [11]–[13], [30] use RNNs on EHR data to model diseases and predict patient diagnosis from health care time series data with irregular time stamps or missing values, but none of them have explicitly attempted to capture and utilize the missing patterns into their RNNs via systematically modified network architectures.

3) GRU-D: Model with trainable decays: To fundamentally address the issue of missing values in time series, we notice two important properties of the missing values in time series, especially in health care domains: First, the value of the missing variable tend to be close to some default value if its last observation happens a long time ago. This property usually exists in health care data for human body as homeostasis mechanisms and is considered to be critical for disease diagnosis



(a) GRU (b) GRU-D

Fig. 2. Graphical illustrations of the original GRU (left) and the proposed GRU-D (right) models. Parts in cyan refer to modifications in GRU-D.

and treatment [31]. Second, the influence of the input variables will fade away over time if the variable has been missing for a while. For example, one medical feature in electronic health records (EHRs) is only significant in a certain temporal context [32]. Therefore we propose a GRU-based model called **GRU-D**, in which a *decay* mechanism is designed for the input variables and the hidden states to capture the aforementioned properties. We introduce *decay rates* in the model to control the decay mechanism by considering the following important factors. First, each input variable in health care time series has its own medical meaning and importance. The decay rates should be flexible to differ from variable to variable based on the underlying properties associated with the variables. Second, as we see lots of missing patterns are informative in prediction tasks, the decay rate should be indicative of such patterns and benefits the prediction tasks. Furthermore, since the missing patterns are unknown and possibly complex, we aim at learning decay rates from the training data rather than fixed a priori. That is, we model a vector of decay rates $\boldsymbol{\gamma}$ as $\boldsymbol{\gamma}_t = \exp\{-\max(\mathbf{0}, \boldsymbol{W}_\gamma \boldsymbol{\delta}_t + \boldsymbol{b}_\gamma)\}$ where $\boldsymbol{W}_\gamma$ and $\boldsymbol{b}_\gamma$ are model parameters that we train jointly with all the other parameters of the GRU. We chose the exponentiated negative rectifier in order to keep each decay rate monotonically decreasing in a reasonable range between 0 and 1. Note that other formulations such as a sigmoid function can be used instead, as long as the resulting decay is monotonic and is in the same range.

Our proposed GRU-D model incorporates two different trainable decays to utilize the missingness directly with the input feature values and implicitly in the RNN states. First, for a missing variable, we use an *input decay* $\boldsymbol{\gamma}_{\boldsymbol{x}}$ to decay it over time toward the empirical mean (which we take as a *default* configuration), instead of using the last observation as it is. Under this assumption, the trainable decay scheme can be readily applied to the measurement vector by $x_t^d \leftarrow m_t^d x_t^d + (1-m_t^d)\gamma_{\boldsymbol{x}_t}^d x_{t'}^d + (1-m_t^d)(1-\gamma_{\boldsymbol{x}_t}^d)\tilde{x}^d$ where $x_{t'}^d$ is the last observation of the d-th variable ($t' < t$) and $\tilde{x}^d$ is the empirical mean of the dth variable. When decaying the input variable directly, we constrain $\boldsymbol{W}_{\gamma_{\boldsymbol{x}}}$ to be diagonal, which effectively makes the decay rate of each variable independent from the others. Sometimes the input decay may not fully capture the missing patterns since not all missingness information can be represented in decayed input values. In order to capture richer knowledge from missingness, we also have a *hidden state decay* $\boldsymbol{\gamma}_{\boldsymbol{h}}$ in GRU-D. Intuitively, this has an effect of decaying the extracted features (GRU hidden states) rather than raw input variables directly. This is implemented by decaying the previous hidden state $\boldsymbol{h}_{t-1}$ before computing the new hidden state $\boldsymbol{h}_t$ as $\boldsymbol{h}_{t-1} \leftarrow \boldsymbol{\gamma}_{\boldsymbol{h}_t} \odot \boldsymbol{h}_{t-1}$, in which case we do not constrain $\boldsymbol{W}_{\gamma_{\boldsymbol{h}}}$ to be diagonal. In addition, we feed the masking ($\boldsymbol{m}_t$) directly into the model. The update functions of GRU-D are

$$
\begin{aligned}
\boldsymbol{r}_t &= \sigma\left(\boldsymbol{W}_r \boldsymbol{x}_t + \boldsymbol{U}_r \boldsymbol{h}_{t-1} + \boldsymbol{V}_r \boldsymbol{m}_t + \boldsymbol{b}_r\right) \\
\boldsymbol{z}_t &= \sigma\left(\boldsymbol{W}_z \boldsymbol{x}_t + \boldsymbol{U}_z \boldsymbol{h}_{t-1} + \boldsymbol{V}_z \boldsymbol{m}_t + \boldsymbol{b}_z\right) \\
\tilde{\boldsymbol{h}}_t &= \tanh\left(\boldsymbol{W} \boldsymbol{x}_t + \boldsymbol{U}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{V} \boldsymbol{m}_t + \boldsymbol{b}\right) \\
\boldsymbol{h}_t &= (\mathbf{1} - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t
\end{aligned}
$$

where $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$ are updated as discussed before, and $\boldsymbol{V}_z, \boldsymbol{V}_r, \boldsymbol{V}$ are new parameters for masking $\boldsymbol{m}_t$. We also propose and compare several model variations of GRU-D [15].

B. Experiments

1) Experimental Settings: We demonstrate the performance of our proposed models on one synthetic and two real-world health-care datasets and compare it to several strong machine learning and deep learning approaches in classification tasks. We evaluate our models for different settings such as early prediction and different training sizes and investigate the impact of missing values.

Gesture phase segmentation dataset (Gesture) This UCI dataset [33] has multivariate time series features, regularly sampled and with no missing values, for 5 different gesticulations. We extracted 378 time series and generate 4 synthetic datasets for the purpose of understanding model behaviors with different missing patterns. We treat it as multi-class classification task.

PhysioNet Challenge 2012 dataset (PhysioNet) This dataset, from *PhysioNet Challenge 2012* [34], is a publicly available collection of multivariate clinical time series from 8000 intensive care unit (ICU) records. Each record is a multivariate time series of roughly 48 hours and contains 33 variables such as *Albumin, heart-rate, glucose* etc. We used *Training Set A* subset in our experiments since outcomes (such as in-hospital mortality labels) are publicly available only for this subset. We conduct mortality prediction task on this dataset.

MIMIC-III dataset (MIMIC-III) This public dataset [35] has deidentified clinical care data collected at Beth Israel Deaconess Medical Center from 2001 to 2012. It contains over 58,000 hospital admission records. We extracted 99 time series features from 19714 admission records collected during 2008-2012 by Metavision data management system which is still employed at the hospital, and only use the first 48 hours data after admission from each time series. We chose four modalities namely *input events* (fluids into patient, e.g. insulin), *output events* (fluids out of the patient, e.g. urine), *lab events* (lab test results, e,g. pH, Platelet count) and *prescription events* (drugs prescribed by doctors, e.g. aspirin and potassium chloride) to collect the patient data recorded in critical care

units and hospital record systems. Mortality prediction task is conducted on this dataset.

We categorize all evaluated prediction models into three groups:

- *Non-RNN Baselines (Non-RNN)*: We evaluate logistic regression (LR), support vector machines (SVM) and Random Forest (RF) which are widely used in health care applications.
- *RNN Baselines (RNN)*: We take GRU-mean, GRU-forward, GRU-simple, and LSTM-mean (LSTM model with mean-imputation on the missing measurements) as RNN baselines. As mentioned before, these models are widely used in existing work [11], [12], [30] on applying RNN on health care time series data with missing values or irregular time stamps.
- *Proposed Methods (Proposed)*: This is our proposed GRU-D model from Section II-A3.

The non-RNN baselines cannot handle missing data directly. We carefully design experiments for non-RNN models to capture the *informative missingness* as much as possible to have fair comparison with the RNN methods. Since non-RNN models only work with fixed length inputs, we regularly sample the time-series data to get a fixed length input and perform imputation to fill in the missing values. Similar to RNN baselines, we can concatenate the masking vector along with the measurements and feed it to non-RNN models. In order to fairly compare the capacity of all GRU-RNN models, we build each model in proper size so they share similar number of parameters. Table I shows the statistics of all GRU-based models for on three datasets.

TABLE I
SIZE COMPARISON OF GRU MODELS USED IN OUR EXPERIMENTS.
Vars. REFERS TO ALL INPUT FEATURES/VARIABLES IN THAT DATASET.
Size REFERS TO THE NUMBER OF HIDDEN STATES ($\boldsymbol{h}$) IN GRU.
Pars. REFERS TO ALL PARAMETERS IN THE NEURAL NETWORK MODEL.

		GRU-mean GRU-forward	GRU-simple	GRU-D
Gesture	# of *Vars.*	18	18	18
	Size	64	50	55
	# of *Pars.*	16281	16025	16561
MIMIC-III	# of *Vars.*	99	99	99
	Size	100	56	67
	# of *Pars.*	60105	59533	60436
PhysioNet	# of *Vars.*	33	33	33
	Size	64	43	49
	# of *Pars.*	18885	18495	18838

2) Quantitative results: To evaluate the impact of modeling missingness we conduct experiments on the synthetic Gesture datasets. We process the data in 4 different settings with the same missing rate but different correlations between missing rate and the label. A higher correlation implies more informative missingness. Figure 3 shows the AUC score comparison of

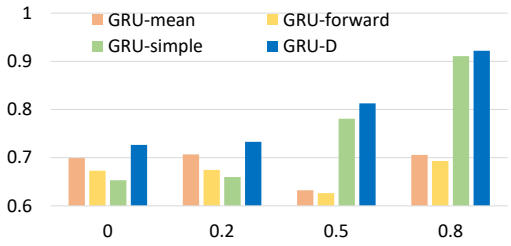


Fig. 3. Classification performance on Gesture synthetic datasets.

three GRU baseline models (GRU-mean, GRU-forward, GRU-simple) and the proposed GRU-D. Since GRU-mean and GRU-forward do not utilize any missingness (i.e., masking or time interval), they perform similarly across all 4 settings. GRU-simple and GRU-D benefit from utilizing the missingness, especially when the correlation is high. Our GRU-D achieves the best performance in all settings, while GRU-simple fails when the correlation is low. The results on synthetic datasets demonstrates that our proposed model can model and distinguish useful missing patterns in data properly compared with baselines.

TABLE II
AUC SCORE ($mean \pm std$) FOR MORTALITY PREDICTION.

	Models	MIMIC-III	PhysioNet
Non-RNN	LR-forward	0.7589 ± 0.015	0.7423 ± 0.011
	SVM-forward	0.7908 ± 0.006	0.8131 ± 0.018
	RF-forward	0.8293 ± 0.004	0.8183 ± 0.015
	LR-simple	0.7715 ± 0.015	0.7625 ± 0.004
	SVM-simple	0.8146 ± 0.008	0.8277 ± 0.012
	RF-simple	0.8294 ± 0.007	0.8157 ± 0.013
RNN	LSTM-mean	0.8142 ± 0.014	0.8025 ± 0.013
	GRU-mean	0.8192 ± 0.013	0.8195 ± 0.004
	GRU-forward	0.8252 ± 0.011	0.8162 ± 0.014
	GRU-simple	0.8380 ± 0.008	0.8155 ± 0.004
Proposed	GRU-D	$\mathbf{0.8527 \pm 0.003}$	$\mathbf{0.8424 \pm 0.012}$

Next, we evaluate all methods in Section II-B1 on MIMIC-III and PhysioNet datasets. We noticed that dropout in the recurrent layer helps a lot for all RNN models on both of the datasets, probably because they contain more input variables and training samples than synthetic dataset. Similar to [36], we apply dropout rate of 0.3 with same dropout samples at each time step on weights $\boldsymbol{W}, \boldsymbol{U}, \boldsymbol{V}$. Table II shows the prediction performance of all the models on mortality task. All models except for random forest improve their performance when they feed missingness indicators along with inputs. The proposed GRU-D achieves the best AUC score on both datasets. We use GRU-simple as a representative for all GRU-simple variant models [11], [12], [30] since it obtains the best or comparable performance among them.

III. Training Scalable Deep Models

A. Methodology

In this section, we describe our framework for performing effective deep learning model training with increasing data and for data with limited and imbalanced labels. We begin by discussing the Laplacian graph-based prior framework to effectively train neural networks with smaller data sets and structured domain knowledge, then describe our incremental neural network procedure to rapidly train a collection of neural networks to detect physiologic patterns of increasing length.

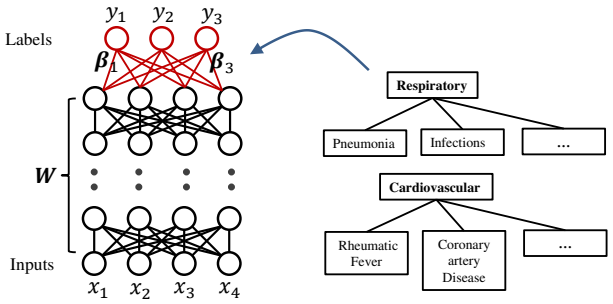


Fig. 4. Illustrations of regularized deep network and categorical structure. The regularization is applied to the output layer of the network.

1) Prior-based Regularization: When we have access to only a few examples of each class label, incorporating prior knowledge can improve learning the deep neural networks. Graph Laplacian-based regularization [17], [37] provides one such framework and is able to incorporate any relational information that can be represented as a (weighted) graph, including the tree-based prior as a special case. Given a matrix $\boldsymbol{A} \in \mathbb{R}^{K\times K}$ representing pairwise connections or similarities, the Laplacian matrix is defined as $\boldsymbol{L} = \boldsymbol{C} - \boldsymbol{A}$, where $\boldsymbol{C}$ is a diagonal matrix with kth diagonal element $C_{k,k} = \sum_{k'=1}^{K}(A_{k,k'})$. $\boldsymbol{L}$ has the following property that makes it interesting for regularization. Given a set of K vectors vector of parameters $\boldsymbol{\beta}_k \in \mathbb{R}^{D^{(L)}}$ and $\mathrm{tr}(\boldsymbol{\beta}^\top \boldsymbol{L}\boldsymbol{\beta}) = \frac{1}{2}\sum_{1\le k,k'\le K} A_{k,k'}\|\boldsymbol{\beta}_k - \boldsymbol{\beta}_{k'}\|_2^2$, where $\mathrm{tr}(\cdot)$ represents the *trace* operator. According to this equation, the graph Laplacian regularizer enforces the parameters $\boldsymbol{\beta}_k$ and $\boldsymbol{\beta}_{k'}$ to be similar, proportional to $A_{k,k'}$.

The graph Laplacian regularizer can represent any pairwise relationships between parameters. We use two different types of priors to incorporate both structured domain knowledge (e.g., label hierarchies based on medical ontologies) and empirical similarities. First, the graph Laplacian regularizer can represent a tree-based prior based on hierarchical relationships found in medical ontologies. In our experiments, we use diagnostic codes from the Ninth Revision of the *International Classification of Diseases* (ICD-9) system, which are widely used for classifying diseases and coding hospital data. The three digits (and two optional decimal digits) in each code form a natural hierarchy including broad body system categories (e.g., *Respiratory*), individual diseases (e.g., *Pneumonia*), and subtypes (e.g., *viral* vs. *Pneumococcal* pneumonia). Figure 4 illustrates two levels of the hierarchical structure of the ICD-9 codes. When using ICD-9 codes as labels, we can treat their ontological structure as prior knowledge. If two diseases belong to the same category, then we add an edge between them in the adjacency graph $\boldsymbol{A}$. Second, we can also incorporate empirical priors, in the form of similarity matrices, estimated from data. For example, we can use the *co-occurrence* matrix $\boldsymbol{A} \in \mathbb{R}^{K\times K}$ whose elements are defined as $A_{k,k'} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{I}(y_{ik}y_{ik'} = 1)$ where N is the total number of the training data points, and $\mathcal{I}(\cdot)$ is the indicator function. Such regularization encourages the learning algorithm to find similar prediction weights based on the pairwise joint probability of the labels.

2) Incremental Training: Next we describe our algorithm for efficiently training a series of deep models to discover and detect physiologic patterns of varying lengths. This framework utilizes a simple and robust strategy for incremental learning of larger neural networks from smaller ones by iteratively adding new units to one or more layers, based upon intelligent initialization of the larger network's parameters using those of the smaller network.

Given a multivariate time series $\boldsymbol{X} \in \mathbb{R}^{P\times T}$, which usually come with varying or increasing lengths, we propose an incremental training procedure that leverages a neural net trained on windows of size T_S to initialize and accelerate the training of a new neural net that detects patterns of length $T' = T_S + \Delta T_S$ (i.e., ΔT_S additional time steps). Suppose that the existing and new networks have $D^{(1)}$ and $D^{(1)} + d^{(1)}$ hidden units in their first hidden layers, respectively, and thus the larger (new) neural network has a $(D^{(1)} + d^{(1)}) \times (D + d)$ weight matrix $\boldsymbol{W}'^{(1)}$. The first D columns of $\boldsymbol{W}'^{(1)}$ correspond exactly to the D columns of $\boldsymbol{W}^{(1)}$ because they take the same D inputs. We can assume that first $D^{(1)}$ hidden units of $\mathbf{h}'^{(1)}$ are highly similar to $\mathbf{h}^{(1)}$ and construct $\boldsymbol{W}'^{(1)}$ by adding d new columns and $d^{(1)}$ new rows to $\boldsymbol{W}^{(1)}$. As illustrated in Figure 5, the new weights can be divided into three categories and be initialized in different ways.

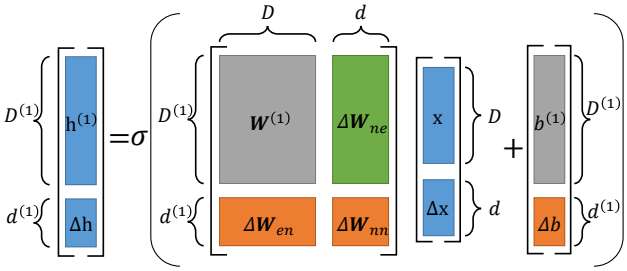


Fig. 5. How adding various units changes the weights.

- $\Delta\boldsymbol{W}_{ne}$: connect new inputs to existing features.
- $\Delta\boldsymbol{W}_{en}$: connect existing inputs to new features.
- $\Delta\boldsymbol{W}_{nn}$: connect new inputs to new features.

Similarity-based initialization for new inputs To initialize $\Delta\boldsymbol{W}_{ne}$, we leverage the fact that we can compute or estimate the similarity among inputs. We can estimate the weight between the ith new input (i.e., input $D + i$) and the jth hidden unit as a linear combination of the parameters for the

existing inputs, weighted by each existing input's similarity to the ith new input. We find using sample covariance or cosine similarity to estimate similarity empirically works well for both time series inputs and arbitrary hidden layers.

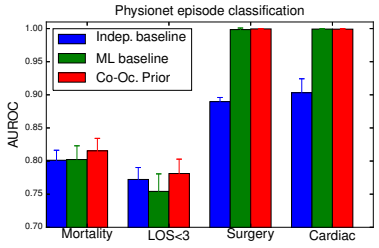
Sampling-based initialization for new features When initializing the weights for $\boldsymbol{W}_{en}$, we do not have the similarity structure, but the weights in $\boldsymbol{W}^{(1)}$ provide information. A simple but reasonable strategy is to sample random weights from the empirical distribution of entries in $\boldsymbol{W}^{(1)}$. We found that estimating all new feature weights from the same simple distribution (based on $\boldsymbol{W}^{(1)}$) worked best.

This framework generalizes beyond the input and first layers. Adding d' new hidden units to $\boldsymbol{h}'^{(1)}$ is equivalent to adding d' new inputs to $\boldsymbol{h}'^{(2)}$. We can still estimate empirical similarity from training data activations in, e.g., $\boldsymbol{h}'^{(2)}$. As if our initializations from the previous pretrained values are sufficiently good, we may be able to forego pretraining. Thus, we choose to initialize with pretrained weights, then do the supervised finetuning on all weights.

B. Experiments

To evaluate our framework, we ran a series of classification and feature-learning experiments using two collections of clinical time series collected during the delivery of care in intensive care units (ICUs) at large hospitals.

1) Experimental Settings: We use PhysioNet and PICU datasets. For PhysioNet dataset, we resample the time series on an hourly basis and propagate measurements forward (or backward) in time to fill gaps. We scale each variable to fall between $[0, 1]$. For PICU dataset, we exclude episodes shorter than 12 hours or longer than 128 hours, yielding a data set of 8,500 multivariate time series of a dozen physiologic variables, which we resample once per hour and scale to $[0, 1]$. For more details on the dataset and our preprocessing steps can be found in [16].



(a) Physionet classification performance.

(b) Co-occurrence.

Fig. 6. Experiment results on Physionet dataset.

2) Benefits of Prior-based Regularization: Our first set of experiments demonstrates the utility of using priors to regularize the training of multi-label neural networks, especially when labels are sparse and highly correlated or similar. From each time series, we extract all subsequences of length $T = 12$ in sliding window fashion, with an overlap of 50% (i.e., stride $R = 0.5T$), and each subsequence receives its episode's labels (e.g., diagnostic code or outcome). We create a small multi-label classification problem on 27,000 subsequences extracted from PhysioNet dataset consisting of four binary labels with strong correlations: in-hospital mortality (*mortality*), length-of-stay less than 3 days (*los<3*), whether the patient had a cardiac condition (*cardiac*), and whether the patient was recovering from surgery (*surgery*), and take the data-driven similarity. Figure 6(b) shows the co-occurrence similarity between the labels. The results for Physionet are shown in Figure 6(a). We observe two trends, which both suggest that multi-label neural networks work well and that priors help. First, jointly learning features, even without regularization, can provide a significant benefit. Both multi-label neural networks dramatically improve performance for the *surgery* and *cardiac* tasks, which are strongly correlated and easy to detect because of our imputation procedure. In addition, the addition of the co-occurrence prior yields clear improvements in the *mortality* and *los<3* tasks while maintaining the high performance in the other two tasks.

3) Efficacy of Incremental Training: In these experiments we show that our incremental training procedure not only produces more effective classifiers (by allowing us to combine features of different lengths) but also speeds up training. We train a series of neural networks designed to model and detect patterns of lengths $T_S = 12, 16, 20, 24$. Each neural net has PT_S inputs (for P variables) and five layers of $2PT_S$ hidden units each. We use each neural network to make an episode-level prediction as before (i.e., the mean real-valued output for all frames) and then combine those predictions to make a single episode level prediction. We compare two training strategies. *Full:* separately train each neural net, with unsupervised pretraining followed by supervised finetuning. *Incremental:* fully train the smallest ($T_S = 12$) neural net and then use its weights to initialize supervised training of the next model ($T_S = 16$). Repeat for subsequent networks. We run experiments on a subset of the ICU data set, including only the 6,200 episodes with at least 24 hours and no more than 128 hours of measurements. This data set yields 50000, 40000, 30000, and 20000 frames of lengths 12, 16, 20, and 24, respectively. Based on our experimental results shown in Figure 7, the incremental training method reduces training time for a single neural net by half. Table III shows the that the incremental training reaches comparable performance, and the combination of incremental training and Laplacian prior leads to better performance than using Laplacian prior only.

IV. Interpretable Mimic Learning

A. Proposed Methods

1) Mimic Learning: Mimicking the performance of deep learning models using shallow models is a recent breakthrough in deep learning which has captured the attention of the machine learning community. [23] showed empirically that shallow neural networks are capable of learning the same function as deep neural networks. They demonstrated this by first training a state-of-the-art deep model, and then training a shallow neural networks to mimic the deep model.

TABLE III
AUROC FOR INCREMENTAL TRAINING.

Size	Level	*Full*	*Inc*	*Prior+Full*	*Prior+Inc*
16	Subseq.	0.6928	0.6874	0.6556	0.6581
	Episode	0.7148	0.7090	0.6668	0.6744
20	Subseq.	0.6853	0.6593	0.6674	0.6746
	Episode	0.7022	0.6720	0.6794	0.6944
24	Subseq.	0.7002	0.6969	0.6946	0.7008
	Episode	0.7185	0.7156	0.7136	0.7171

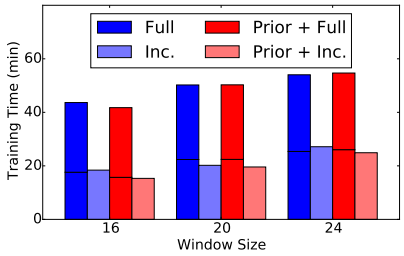


Fig. 7. Training time comparison of different training strategies.

[24] proposed an efficient knowledge distillation approach to transfer (dark) knowledge from model ensembles into a single model. All these previous works motivate us to explore the possibility of employing the mimic learning strategy to learn an interpretable model from a well-trained deep neural network.

2) Proposed Interpretable Mimic Learning Framework: The basic idea distilling knowledge from deep models [23], [24] is utilizing soft labels learned from the based model (i.e, deep neural networks) to train the mimic model (i.e., shallow neural networks or other simple models). The general training procedure of the Interpretable Mimic Learning model is shown in Figure 8.

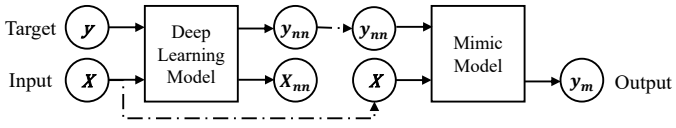


Fig. 8. Training pipeline for interpretable mimic learning method.

In the first step, we train a deep learning model, which can be a simple feedforward network or GRU, or a more powerful multimodal deep learning model, given the input $\boldsymbol{X}$ and the original target y (which is either 0 or 1 for binary classification). Then, for each input sample $\boldsymbol{X}$, we obtain the soft prediction score $y_{nn} \in [0, 1]$ from the prediction layer of the neural network. In the second step, we train a mimic Gradient boosting model, given the raw input $\boldsymbol{X}$ and the soft label y as the model input and target, respectively. We train the mimic model to minimize the mean squared error of the output y_m to the soft label y_{nn}. After training, the mimic model will perform similarly as the original deep learning model, but with its own intrinsic interpretability. Finally, we can directly apply the mimic model trained in the second step to the original classification task.

Our interpretable mimic learning model has several advantages over existing deep learning and gradient boosting methods. First, the original deep learning model can usually boost the performance on many tasks over traditional methods, and gradient boosting methods are good at maintaining the performance of the original complex model by mimicing its predictions. Second, the original deep learning model is too complex to interpret. Our mimic methods, however, provides better interpretability than original model, by explaining each feature individually and examining simple rules from the tree structures. Furthermore, using soft targets from deep learning models avoids overfitting to the original data and provides good generalizations, which can not be achieved by standard decision tree methods or other simple models.

3) Interpretable Model - Gradient Boosting Trees: Gradient boosting [27] is a method which takes an ensemble of weak learners to optimize a differentiable loss function by stages. The basic idea is that the prediction function $F(x)$ can be approximated by a linear combination of several functions (under some assumptions), and these functions can be sought using gradient descent approaches. Gradient Boosting Trees (GBT) takes a simple classification or regression tree as each weak learner. At each stage m, assume the current model is $F_m(x)$, then the Gradient Boosting method tries to find a weak model $h_m(x)$ to fit the gradient of the loss function with respect to $F(x)$ at $F_m(x)$. The coefficient γ_m of the stage function is computed by the line search strategy to minimize the loss. To keep gradient boosting from overfitting, a regularization method called shrinkage is usually employed, which multiplies a small learning rate ν to the stage function in each stage. The final model with M stages can be written as $F_M(x) = \sum_{i=1}^{M} \nu\gamma_i h_i(x) + const$.

B. Evaluations of Interpretable Mimic Learning

1) Experimental Settings: We conduct experiments with a group of our mimic learning models: For each of the deep models we take its soft prediction scores and apply gradient boosting methods. These methods are denoted by GBTmimic with GBT as the mimic (student) model.

We conduct experiments on MIMIC-III and VENT datasets. In MIMIC-III dataset, we remove ambiguous and noisy observations and extract features per 2-hour within first 24 hours after admissions, and perform forward imputation for lab event variables and zero imputation for other variables to fill in the missing values. Our main task on the MIMIC-III dataset is predicting the ICD-9 Diagnosis Code for each admission record (MIMIC-III-ICD9). For VENT dataset [38], we perform simple imputation for filling the missing values where we take the majority value for binary variables, and empirical mean for other variables. We perform two binary classification tasks on the VENT dataset:

- Mortality (VENT-MOR): we predict whether the patient dies within 60 days after admission or not. 20.10% of all the patients are mortality positive (patients who die).
- Ventilator Free Days (VENT-VFD): in this task, we are interested in evaluating a surrogate outcome of morbidity and mortality (Ventilator free Days, of which lower value is bad), by identifying patients who survive and are on a ventilator for longer than 14 days. Since here lower VFD is bad, it is a bad outcome if the value ≤ 14, otherwise it is a good outcome. 59.05% of all the patients are VFD positive VFD (patients who survive and stay long enough on ventilators).

TABLE IV
CLASSIFICATION RESULTS ON VENT DATASET. ONLY THE BEST PERFORMANCES FOR EACH MODEL TYPE ARE SHOWN.

Models	MOR (Mortality)		VFD (Ventilator Free Days)	
	AUROC	AUPRC	AUROC	AUPRC
Non-Deep	0.7196 ± 0.06	0.4171 ± 0.10	0.7592 ± 0.05	0.8142 ± 0.05
Deep	0.7813 ± 0.07	$\mathbf{0.4874 \pm 0.13}$	$\mathbf{0.7896 \pm 0.05}$	$\mathbf{0.8397 \pm 0.05}$
Mimic	$\mathbf{0.7898 \pm 0.08}$	0.4766 ± 0.13	0.7889 ± 0.05	0.8324 ± 0.04

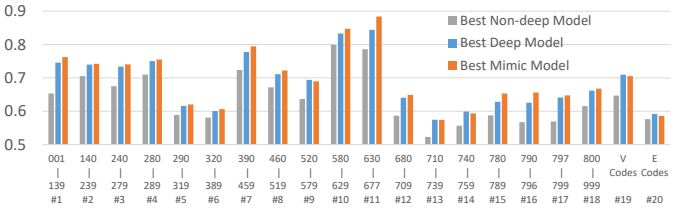


Fig. 9. Classification results (AUROC) on MIMIC-III dataset. x-axis: ICD-9 tasks; y-axis: AUROC.

2) Performance of Mimic Learning Models: Table IV shows the prediction performance of different methods on the predictive tasks of the VENT dataset, and the detailed results can be found in our paper [26]. Our interpretable mimic methods obtain similar or even slightly better performance compared with deep models. Figure 9 shows the prediction performance of different methods for all 20 ICD-9 diagnosis tasks on the MIMIC-III dataset. For the simple baselines used in these tasks (LR and SVM), we also include the statistics for each time series variable (average, min, and max) as the features, which is recognized as one of the state-of-art methods in ICU data analysis. Again, our proposed mimic models clearly outperform the baselines and achieve similar performance of multimodal models on most of the tasks.

3) Interpretations: Since our mimic models come from additive and tree-based Gradient Boosting methods, there are several tools for interpreting the models. Among them, the feature importance measurement, partial dependence plots and the important decision rules are commonly used in practice. We will discuss these interpretation approaches and provide some case studies.

a) Feature Influence: One of the most common interpretation tools for tree-based algorithms is the feature importance (influence of variable) [27]. The influence of one variable j in a single tree T with L splits is based on the numbers of times when the variable is selected to split the data samples. Formally, the influence Inf is defined as $Inf_j(\mathrm{T}) = \sum_{l=1}^{L-1} I_l^2 \mathbb{I}(S_l = j)$ where I_l^2 refers to the empirical squared improvement after the split l, and $\mathbb{I}$ is the identity function. The importance scores of the entire GBT is defined as the average influence across all the trees, and are normalized across all the variables. Although importance scores does not tell anything about how the feature is actually used in the model, it is a quite useful metric for feature selection.

Analysis on VENT dataset Table V shows the most useful features for MOR and VFD tasks on VENT dataset, respectively, from both GBT and GBTmimic models. We find that some important features are shared with several methods in these two tasks, e.g., MAP (Mean Airway Pressure) at day 1, δPF (Change of PaO2/FIO2 Ratio) at day 1, etc. Besides, almost all the top features are temporal features, while among all static features, the PRISM (Pediatric Risk of Mortality) score, which is developed and commonly used by the doctors and medical experts, is the most useful static variable.

TABLE V
TOP FEATURES AND THEIR CORRESPONDING IMPORTANCE SCORES ON VENT DATASET.

Task	MOR (Mortality)		VFD (Ventilator Free Days)	
Model	GBT	GBTmimic	GBT	GBTmimic
Features	PaO2-Day2 (0.0539)	BE-Day0 (0.0433)	MAP-Day1 (0.0423)	MAP-Day1 (0.0384)
	MAP-Day1 (0.0510)	δPF-Day1 (0.0431)	PH-Day3 (0.0354)	PIM2S (0.0322)
	BE-Day1 (0.0349)	PH-Day1 (0.0386)	MAP-Day2 (0.0297)	VE-Day0 (0.0309)
	FiO2-Day3 (0.0341)	PF-Day0 (0.0322)	MAP-Day3 (0.0293)	VI-Day0 (0.0288)
	PF-Day0 (0.0324)	MAP-Day1 (0.0309)	PRISM12 (0.0290)	PaO2-Day0 (0.0275)

b) Partial Dependence Plots: Visualizations provide better understanding and comparison of complex mimic models. In GBT, we can get visualizations by plotting the partial dependence of a specific variable or a subset of variables. The partial dependence can be treated as the approximation of the prediction function given only the specific variable(s). It can be computed by getting the prediction values with marginalizing over the values of all other variables.

One-way partial dependence and additive function Based on the feature list shown in Table V, we are more interested in finding how the important features exactly influence the model predictions. Furthermore, by investigating the influence of the same variable in different models, people can easily check and compare different mimic models. Figure 10 shows the one-way partial dependence from GBTmimic models for mortality task on VENT dataset, and both these two methods

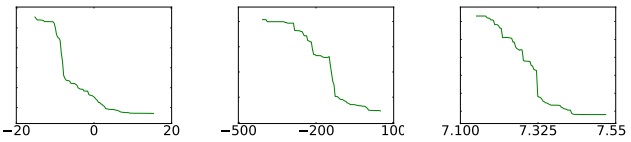


Fig. 10. One-way partial dependence and function plots of top three important features in GBTmimic models for VENT-MOR task. From left to right in each row: BE-Day0, δPF-Day1, PH-Day1. x-axis: variable value; y-axis: dependence/function value.

mimic the same HMMDL model. First, we can see that while the two mimic methods are different, what they have learned from the base model are surprisingly similar. This stable result by different mimic learning models also verifies that the mimic models are rather robust and effective in distilling the useful knowledge from the base model. Second, the results are easy to interpret and match existing findings. For instance, both of our mimic models predict a higher chance of mortality when the patient has value of PH-Day0 below 7.325. That conforms to the clinical knowledge that human blood stays in a very narrow pH range around 7.35 - 7.45. If the pH value of blood decreases from 7.45 to 7.30, the blood will loss 64.9% oxygen, which leads to severe symptoms and disease. More useful rules from our mimic models can be found via the partial dependence plots, which demonstrate an insightful approach to understand and utilize the more powerful yet complex deep models through mimic learning.

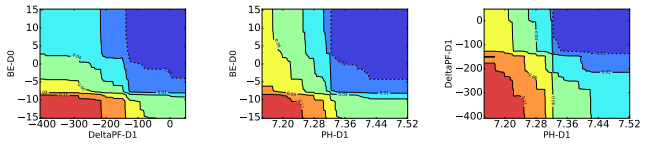


Fig. 11. Pairwise partial dependence plots for top important features of GBTmimic model on VENT-MOR task. Red: Positive dependence; Blue: Negative dependence.

Two-way partial dependence It would be more helpful to investigate these interactions among the most important features. One possible way is to generate 2-dimensional partial dependence for two important features. Figure 11 demonstrates the 2-way dependence scores of the top three features used in our GBTmimic model. One can see that these plots are consistent with the corresponding one way plots, and the interactions between the two features can be exposed by the former figures. These plots can also be examined domain experts to obtain a better understanding of the model and to identify unknown interactions between the features.

c) Top Decision Rules: Another way to evaluate our mimic methods is to compare and interpret the trees obtained from our models. Figure 12 shows the examples of the most important trees built by our interpretable mimic learning methods for MOR and VFD tasks on VENT dataset and the ICD-9 diagnosis prediction task on MIMIC-III dataset. We choose the tree with the highest coefficient weight in the final prediction function as the most important tree. Some

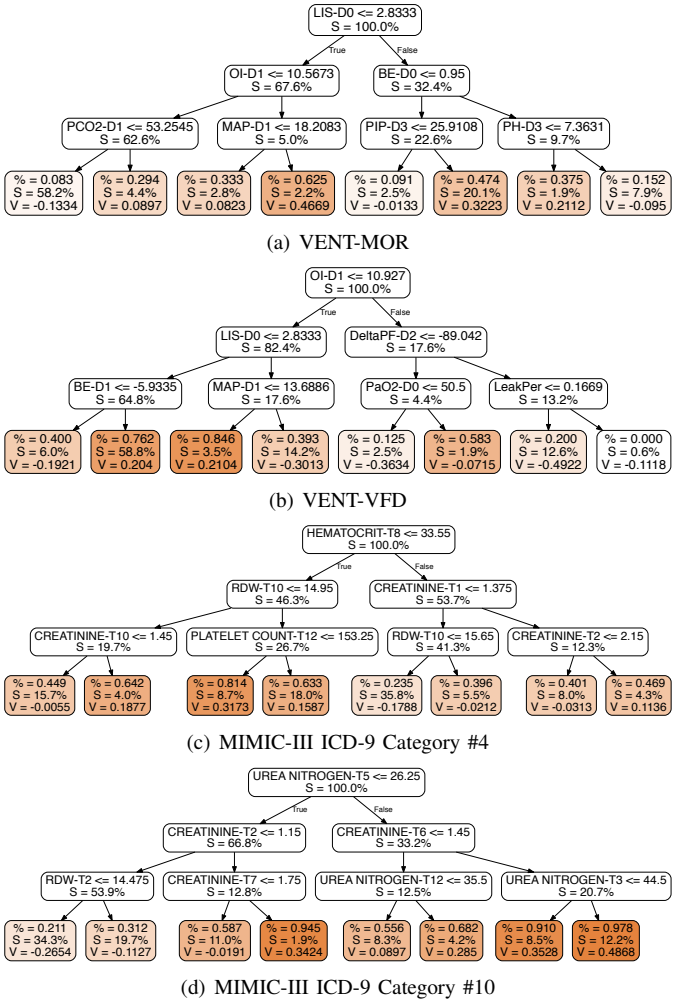


Fig. 12. Sample decision trees from best GBTmimic models. %: Class distribution for samples belong to that node; S: # of Samples to that node; V: Prediction value of that node; Darker leaf node has more samples with positive labels.

observations from this figure are as follows: ICD-9 codes for task 10 (ICD-9 Code range 580-629) correspond to the diseases of Genitourinary System, and the top features found in our trees are *Urea Nitrogen* and *Creatinine* which are present in the lab tests performed on urine and blood collected from the patients. It is known that blood urea-nitrogen(BUN)-to-creatinine ratio generally provides a precise information about kidney function and may be used to determine the cause of acute kidney injury or dehydration [39]. Similarly, features in the top tree of task 4 are *Hematocrit*, *RDW*, *Creatinine* and *plate count*; and their analysis reveals that these are quite useful for the ICD-9 task 4 prediction task which corresponds to the diseases of blood.

V. Summary

In this paper, we provide efficient and effective deep learning solutions for three key challenges in computational phenotyping research in health care. For the challenge from missing data, we designed GRU-D, a variation of GRU model,

to utilize informative missingness. We propose prior-based regularization framework for health care data with imbalanced and sparse labels, and incremental training strategy to efficiently handle more input data. Furthermore, we propose a simple and effective interpretable mimic learning framework, which has both superior performance and good interpretability. All the proposed methods are shown to outperform existing baselines on real-world health care datasets.

REFERENCES

[1] G. Hripcsak and D. J. Albers, "Next-generation phenotyping of electronic health records," *Journal of the American Medical Informatics Association*, vol. 20, no. 1, pp. 117–121, 2013.

[2] N. V. Chawla and D. A. Davis, "Bringing big data to personalized healthcare: a patient-centered framework," *Journal of general internal medicine*, vol. 28, no. 3, pp. 660–665, 2013.

[3] J. Zhou, F. Wang, J. Hu, and J. Ye, "From micro to macro: Data driven phenotyping by densification of longitudinal electronic medical records," in *KDD*, 2014.

[4] J. C. Ho, J. Ghosh, and J. Sun, "Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization," in *KDD*, 2014.

[5] R. L. Richesson, J. Sun, J. Pathak, A. N. Kho, and J. C. Denny, "Clinical phenotyping in selected national networks: Demonstrating the need for high-throughput, portable, and computational methods," *Artificial intelligence in medicine*, vol. 71, pp. 57–61, 2016.

[6] D. B. Rubin, "Inference and missing data," *Biometrika*, vol. 63, no. 3, pp. 581–592, 1976.

[7] D. M. Kreindler and C. J. Lumsden, "The effects of the irregular sample and missing data in time series analysis," *Nonlinear Dynamical Systems Analysis for the Behavioral Sciences Using Real Data*, p. 135, 2012.

[8] P. J. García-Laencina, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal, "Pattern classification with missing data: a review," *Neural Computing and Applications*, vol. 19, no. 2, pp. 263–282, 2010.

[9] K. Rehfeld, N. Marwan, J. Heitzig, and J. Kurths, "Comparison of correlation analysis techniques for irregularly sampled time series," *Nonlinear Processes in Geophysics*, vol. 18, no. 3, pp. 389–404, 2011.

[10] B. J. Wells, K. M. Chagin, A. S. Nowacki, and M. W. Kattan, "Strategies for handling missing data in electronic health record derived data," *EGEMS*, vol. 1, no. 3, 2013.

[11] Z. C. Lipton, D. C. Kale, and R. Wetzel, "Modeling missing data in clinical time series with rnns," *Machine Learning for Healthcare*, 2016.

[12] E. Choi, M. T. Bahadori, and J. Sun, "Doctor ai: Predicting clinical events via recurrent neural networks," *arXiv preprint arXiv:1511.05942*, 2015.

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

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

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

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

[17] K. Q. Weinberger, F. Sha, Q. Zhu, and L. K. Saul, "Graph laplacian regularization for large-scale semidefinite programming," in *NIPS*, 2006.

[18] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R. A. Greenes, R. Hall, P. D. Johnson, N. Jones, A. Kumar *et al.*, "Comparing computer-interpretable guideline models: a case-study approach," *Journal of the American Medical Informatics Association*, vol. 10, no. 1, pp. 52–68, 2003.

[19] K. F. Kerr, A. Bansal, and M. S. Pepe, "Further insight into the incremental value of new markers: the interpretation of performance measures and the importance of clinical context," *American journal of epidemiology*, p. kws210, 2012.

[20] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[21] G. Bonner, "Decision making for health care professionals: use of decision trees within the community mental health setting," *Journal of Advanced Nursing*, vol. 35, no. 3, pp. 349–356, 2001.

[22] Z. Yao, P. Liu, L. Lei, and J. Yin, "R-c4. 5 decision tree model and its applications to health care dataset," in *Services Systems and Services Management, 2005. Proceedings of ICSSSM'05. 2005 International Conference on*, vol. 2. IEEE, 2005, pp. 1099–1103.

[23] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Advances in Neural Information Processing Systems*, 2014, pp. 2654–2662.

[24] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

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

[26] ——, "Interpretable deep models for icu outcome prediction," in *AMIA Annual Symposium Proceedings*, vol. 2016. American Medical Informatics Association, 2016, p. 371.

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

[28] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

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

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

[31] Y. Vodovotz, G. An, and I. P. Androulakis, "A systems engineering perspective on homeostasis and disease," *Frontiers in bioengineering and biotechnology*, vol. 1, 2013.

[32] L. Zhou and G. Hripcsak, "Temporal reasoning with medical dataa review with emphasis on medical natural language processing," *Journal of biomedical informatics*, vol. 40, no. 2, pp. 183–202, 2007.

[33] R. C. Madeo, C. A. Lima, and S. M. Peres, "Gesture unit segmentation using support vector machines: segmenting gestures from rest positions," in *SAC*, 2013.

[34] I. Silva, G. Moody, D. J. Scott, L. A. Celi, and R. G. Mark, "Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012," in *CinC*, 2012.

[35] A. Johnson, T. Pollard, L. Shen, L. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Celi, and R. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific Data*, 2016.

[36] Y. Gal, "A theoretically grounded application of dropout in recurrent neural networks," *arXiv preprint arXiv:1512.05287*, 2015.

[37] T. Zhang, A. Popescul, and B. Dom, "Linear prediction models with graph regularization for web-page categorization," in *KDD*, 2006.

[38] R. G. Khemani, D. Conti, T. A. Alonzo, R. D. Bart, and C. J. Newth, "Effect of tidal volume in children with acute hypoxemic respiratory failure," *Intensive care medicine*, vol. 35, no. 8, pp. 1428–1437, 2009.

[39] S. Uchino, R. Bellomo, and D. Goldsmith, "The meaning of the blood urea nitrogen/creatinine ratio in acute kidney injury," *Clinical kidney journal*, vol. 5, no. 2, pp. 187–191, 2012.