Distilling Knowledge from Deep Networks with Applications to Healthcare Domain

Zhengping Che*, Sanjay Purushotham*, Robinder Khemani, Yan Liu***
*Department of Computer Science, University of Southern California
**Children's Hospital Los Angeles
*{zche, spurusho, yanliu.cs}@usc.edu, **RKhemani@chla.usc.edu

Abstract

Exponential growth in Electronic Healthcare Records (EHR) has resulted in new opportunities and urgent needs for discovery of meaningful data-driven representations and patterns of diseases in *Computational Phenotyping* research. Deep Learning models have shown superior performance for robust prediction in computational phenotyping tasks, but suffer from the issue of model interpretability which is crucial for clinicians involved in decision-making. In this paper, we introduce a novel knowledge-distillation approach called **Interpretable Mimic Learning**, to learn interpretable phenotype features for making robust prediction while mimicking the performance of deep learning models. Our framework uses Gradient Boosting Trees to learn interpretable features from deep learning models such as Stacked Denoising Autoencoder and Long Short-Term Memory. Exhaustive experiments on a real-world clinical time-series dataset show that our method obtains similar or better performance than the deep learning models, and it provides interpretable phenotypes for clinical decision making.

1 Introduction

With the exponential surge in the amount of electronic health records (EHR) data, there come both the opportunities and the urgent needs for discovering meaningful data-driven characteristics and patterns of diseases, which is known as *phenotype discovery*. Clinicians are collaborating with machine learning researchers to tackle many computational phenotyping problems to improve the state of healthcare services and this is paving the way for *Personalized Healthcare* [8]. Robust prediction is critical in healthcare research since it is directly related to saving patient lives. Recent works [9, 28] have used deep learning models to achieve state-of-the-art performance on computational phenotype prediction problems. However, deep learning models are less interpretable while clinicians mainly rely on interpretable models to make informed clinical decisions. Thus, the fundamental question is how we can develop new data-driven machine learning techniques which can achieve state-of-the-art performance as deep learning models and also discover interpretable features (phenotypes).

Deep learning models are revolutionizing many fields such as computer vision [27, 23, 44], and speech and language processing [31, 47], and have achieved the status as the go-to state-of-the-art techniques for many machine learning tasks. With the flexibility and power of all kinds of neural networks, some deep network variants are also potentially suitable for healthcare tasks [9, 30]. Autoencoder [42] is used to capture structures in data and aims to reconstruct the input. It has been successfully used for feature extraction or as a pre-training step for a neural network [37, 19]. Long Short-Term Memory (LSTM) architecture [21] has been widely used for sequential data and tasks recently [39, 2]. It reads the input sequence one time step at a time and provides fixed-length or step-by-step representations, while keeping the long range temporal dependencies. While deep models perform superior to many approaches when the data is abundant, their performance can drop

significantly when the data is noisy and sparse or when the model is not properly initialized [12]. Also, the features learned using deep models are generally not interpretable.

There is limited work on interpreting the features learned by deep learning outside of computer vision [13]. Recent research has begun to provide a more rigorous understanding of the representations learned by deep architectures. It has been shown that the semantics encoded by hidden unit activations in one layer are preserved when projected onto random bases, instead of the next layer's bases [40]. This implies that the practice of interpreting individual units can be misleading and that the behavior of deep models may be more complex than previously believed. In healthcare, model interpretability is not only important but also *necessary*, since the primary care providers, physicians and clinical experts alike depend on the new healthcare technologies to help them in monitoring and decision-making for patient care. A good interpretable model is shown to result in faster adoptability among the clinical staff and results in better quality of patient care [35, 24]. Therefore we need to identify novel solutions which can provide interpretable models and achieve similar prediction performance as deep models in healthcare domain.

In order to capture the performance of deep learning models using other models, a knowledge distillation process such as mimic learning [1] or dark knowledge [20, 26] is useful. In these works, it was noted that once a deep learning model is trained on a large-scale dataset, we can use a smaller model to distill the knowledge by training it on the "soft target", i.e., the class probability produced by the former model. This means that simple models can possibly match (mimic) the prediction performance of deep models. Thus, choosing the right simple model will help to extract informative physiologic patterns which in-turn helps to discover meaningful interpretable features.

Building upon the recent breakthrough in mimic learning, in this paper, we introduce our knowledge-distillation approach called **Interpretable Mimic Learning**, to learn interpretable features for making robust prediction while mimicking the performance of deep learning models. Unlike the standard mimic learning [1], our interpretable mimic learning framework uses Gradient Boosting Trees (GBT) to learn interpretable features from deep learning models. We use GBT as our mimicking model since they not only provide interpretable decision rules and tree structures, but also successfully maintain the performance of original complex models such as deep networks. Our main contributions in this paper include:

- We propose a novel knowledge distillation methodology called *Interpretable Mimic Learning* where we mimic the performance of state-of-the-art deep learning models using well-known Gradient Boosting Trees (GBT).
- We conduct extensive experiments on several deep learning architectures including Stacked denoising autoencoders (SDA) and Long Short Term Memory (LSTM) to show that our Interpretable Mimic Learning models can achieve state-of-the-art performance on multiple deep learning models.
- We discuss the interpretable features and decision rules learned by our Interpretable Mimic Learning models, which is validated by the expert clinicians. We also conduct experiments to investigate, for different deep networks, whether using neural network extracted features rather than soft labels improves mimicking performance.

The remainder of this paper is arranged as follows: In Section 2, we provide an overview of the related work; In Section 3, we describe our proposed Interpretable Mimic Learning framework and discuss the related deep learning models; An evaluation on empirical results and interpretable features is presented in the Section 4; We conclude with discussion, summary and future work in the Section 5.

2 Related Work

In this section, we first provide an overview of the state-of-the-art deep learning approaches used in the healthcare domain, and then we discuss the recent advances in *Mimic learning* approaches. Recently, there is a growing interest in applying deep learning techniques to computational phenotyping [33] due to the increasing availability of the Electronic Healthcare Records (EHR) and the need for Personalized Healthcare [4, 17]. One of the first applications of modern deep learning to clinical time series was described in [28], where the authors use autoencoders to learn features

from longitudinal clinical measurement time series and show interpretable features which are useful for classifying and clustering different types of patients. In our previous work [9], we proposed a novel scalable deep learning framework which models the prior-knowledge from medical ontologies to learn interpretable and clinically relevant features for patient diagnosis in Intensive Care Units (ICU). A recent study [11] showed that a neural network model can improve the prediction of the likelihood of several psychological conditions such as anxiety, behavioral disorders, depression, and post-traumatic stress disorder. Other recent works [18, 30] also leverage the power of deep learning approaches to model diseases and clinical time series data. These previous works have successfully showed the state-of-the-art performance of deep learning models for the healthcare domain but they have made limited attempts at the interpretability of the features learned by deep learning models, which prevents the clinician from understanding and applying these models.

As pointed out in the introduction, model interpretability is not only important but also *necessary* in healthcare domain. Decision trees [36] - due to their easy interpretability - have been quite successfully employed in the healthcare domain [6, 45, 14] and clinicians have embraced it to make informed decisions. However, decision trees can easily overfit and they do not achieve good performance on datasets with missing values which is common in today's healthcare datasets. On the otherhand, deep learning models have achieved remarkable performance in healthcare as discussed in the previous paragraph, but their learned features are hardly interpretable. Here, we review some recent works on interpretability of deep learning features conducted in computer vision field. [13] studied the visualizations of the hierarchical representations learned by deep networks. [46] investigated not only the visualizations but also demonstrated the feature generalizablility in convolutional neural networks. [40] argued that interpreting individual units can be misleading and postulated that the semantics encoded by hidden unit activations in one layer are preserved when projected onto random bases, instead of the next layer's bases. These works show that interpreting deep learning features is possible but the behavior of deep models may be more complex than previously believed. Therefore we believe there is a need to identify novel solutions which can provide interpretable models and achieve similar prediction performance as deep models.

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. [1] 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 model to mimic the deep model. Motivated by the model compression idea from [7], [20] proposed an efficient knowledge distillation approach to transfer (dark) knowledge from model ensembles into a single model. [26] takes a Bayesian approach for distilling knowledge from a teacher neural network to train a student neural network. Excellent performance on real world tasks using mimic learning has been recently demonstrated in [29]. All these previous works, motivate us to employ mimic learning strategy to learn an interpretable model from a well-trained deep neural network, which will be clearly discussed in the following section.

3 Methods

In this section, we will first describe the notations used in the paper, and then we describe the state-of-the-art deep learning models which we use as the original models and the Gradient Boosting Trees which we use as interpretable models for mimicking original models. Finally, we present the general pipeline of our interpretable mimic learning framework.

3.1 Notations

EHR data contains both static and temporal features. Without loss of generality, we assume each data sample has static records with Q variables and temporal data of length T and P variables, as well as a binary label $y \in \{0, 1\}$, where y usually represents a patient's health state. By flattening the time series and concatenating static variables, we get an input vector $\boldsymbol{X} \in \mathbb{R}^D$ for each sample, where $D = TP + Q$.

3.2 Feedforward Network and Stacked Denosing Autoencoder

A multilayer feedforward network [22] is a neural network with multiple nonlinear layers and possibly one prediction layer on the top. The first layer takes $\boldsymbol{X}$ as the input, and the output of each layer is used as the input of the next layer. The transformation of each layer l can be written as

$$\boldsymbol{X}^{(l+1)} = f^{(l)}(\boldsymbol{X}^{(l)}) = s^{(l)}\left(\boldsymbol{W}^{(l)}\boldsymbol{X}^{(l)} + \boldsymbol{b}^{(l)}\right)$$

where $\boldsymbol{W}^{(l)}$ and $\boldsymbol{b}^{(l)}$ are respectively the weight matrix and bias vector of layer l, and $s^{(l)}$ is a nonlinear activation function, which usually takes *logistic sigmoid*, *tanh*, or *ReLU* [32] functions. While we optimize the cross-entropy prediction loss and get the prediction output from topmost prediction layer, the activation of the hidden layers are also very useful as learned features.

The Stacked Autoencoder [5] has a very similar structure as feedforward network, but its main objective is to minimize the squared reconstruction loss to the input instead of the cross-entropy prediction loss, by using encoder and decoder networks with tied weights. Assume the encoder has the same structure as feedforward network, then the l-th layer of the decoder takes the output $\boldsymbol{Z}^{(l+1)}$ from the next layer and transforms it by

$$\boldsymbol{Z}^{(l)} = s^{(l)}\left({\boldsymbol{W}^{(l)}}^{\mathrm{T}}\boldsymbol{Z}^{(l+1)} + \boldsymbol{b_d}^{(l)}\right)$$

where $\boldsymbol{Z}^{L+1} = \boldsymbol{X}^{(L+1)}$ is the output from the encoder, and finally $\boldsymbol{Z}^{(0)}$ can be treated as the reconstruction of the original input. By adding noise to the input, i.e. hiding some input variables randomly, but still trying to recover the uncorrupted input, we obtain the Stacked Denoising Autoencoder [42, 43] which is more robust to noise than the Stacked Autoencoder. After training a stacked autoencoder, we add a logistic prediction layer on the encoder to solve the prediction task.

3.3 Long Short-Term Memory

If we want to apply temporal model and only focus on the P temporal variables, we can apply time series models on input $\boldsymbol{X}_{ts} = (x_1, x_2, \cdots, x_T)^{\mathrm{T}} \in \mathbb{R}^{T\times P}$, where $x_t \in \mathbb{R}^P$ is the variables at time t. Long Short-Term Memory (LSTM) [21] is a popular recurrent neural networks for sequential data and tasks. It is used to avoid the vanishing gradient problem which is prevalent in other recurrent neural network architectures. Figure 1(a) shows the standard structure of an LSTM block with input, forget, and output gates, which we use in our method. In step t, one LSTM block takes the time series input x_t at that time, cell state C_{t-1} and output h_{t-1} from previous time step, and calculates the cell state C_t and output h_t at this time step. We use the following steps to compute the output from each gate:

$$f_t = \sigma\left(\boldsymbol{W}_{fh}h_{t-1} + \boldsymbol{W}_{fx}x_t + b_f\right) \qquad i_t = \sigma\left(\boldsymbol{W}_{ih}h_{t-1} + \boldsymbol{W}_{ix}x_t + b_i\right)$$
$$\tilde{C}_t = tanh\left(\boldsymbol{W}_{Ch}h_{t-1} + \boldsymbol{W}_{Cx}x_t + b_C\right) \qquad o_t = \sigma\left(\boldsymbol{W}_{oh}h_{t-1} + \boldsymbol{W}_{ox}x_t + b_o\right)$$

And the outputs from this LSTM block is computed as:

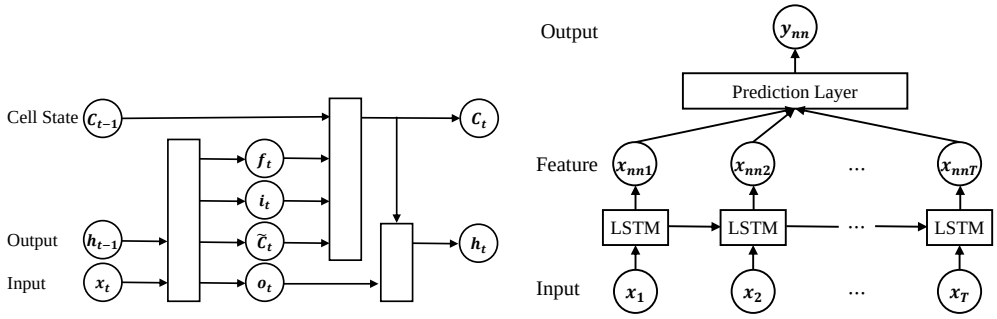
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \qquad h_t = o_t * tanh(C_t)$$

where $*$ refers to the element-wise multiplication of two vectors.

In our LSTM prediction model, we flatten the output from each block, which is denoted as $\boldsymbol{X}_{nn} = (x_{nn1}, x_{nn2}, \cdots, x_{nnT}) = (h_1, h_2, \cdots, h_T)$, and we add another prediction layer on top of them. The model is shown in Figure 1(b).

3.4 Gradient Boosting Trees

Gradient boosting [15, 16] is a method which takes an ensemble of weak learners, usually decision trees, 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 seeked using gradient descent approaches. 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



(a) LSTM Block (b) LSTM Prediction Model

Figure 1: A Sketch of Long Short-Term Memory Model

the stage function is computed by the line search strategy to minimize the loss. The final model with M stages can be written as

$$F_M(x) = \sum_{i=1}^{M} \gamma_i h_i(x) + const$$

In Gradient Boosting Trees, each weak learner is a simple classification or regression tree. 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. In this situation, the term $\gamma_i h_i(x)$ in the update rule above is replaced by $\nu\gamma_i h_i(x)$.

3.5 Interpretable Mimic Learning method

In this section, we describe a simple but effective knowledge distillation framework - the *Interpretable Mimic Learning* method also termed as the *GBTmimic model*, which trains Gradient Boosting Trees to mimic the performance of deep network models. Our mimic method aims to recognize interpretable features while maintaining the state-of-the-art classification performance of the deep learning models. To investigate, for different deep networks, whether using neural network extracted features rather than soft labels improve the mimicking performance we present two general pipelines for our GBTmimic model. The two pipelines of GBTmimic model are shown in Figure 2.

In Pipeline 1, we utilize the learned features from deep networks and resort to another classifier such as Logistic Regression:

1. Given the input features $\boldsymbol{X}$ and target y, we train a deep neural network, either Stacked Denoising Autoencoder or Long Short-Term Memory, with several hidden layers and one prediction layer. We take the activations of the highest hidden layers as the extracted features $\boldsymbol{X}_{nn}$ from that deep network.
2. We then feed these new features into a standard classifier, e.g., Logistic Regression, to train on the same classification task, i.e. the target is still y, and take the soft prediction scores y_c from the classifier.
3. Finally we train a mimic model, i.e., Gradient Boosting Regression Trees, given the raw input $\boldsymbol{X}$ and the soft targets y_c to get the final output y_m with minimum mean squared error.

In Pipeline 2, we directly use the predicted soft-labels from deep networks.

1. The first step is similar to that in Pipeline 1, where we train a deep neural network with input features $\boldsymbol{X}$ and target y, but we take take the soft prediction scores y_{nn} directly from the prediction layer of the neural network.
2. Instead of training an extra classifier with extracted features, we take the soft prediction scores y_{nn} use it as the target in training the mimic model. In other words, we train Gradient Boosting Trees, which can output y_m with minimum mean squared error to y_{nn}, given the raw input $\boldsymbol{X}$.

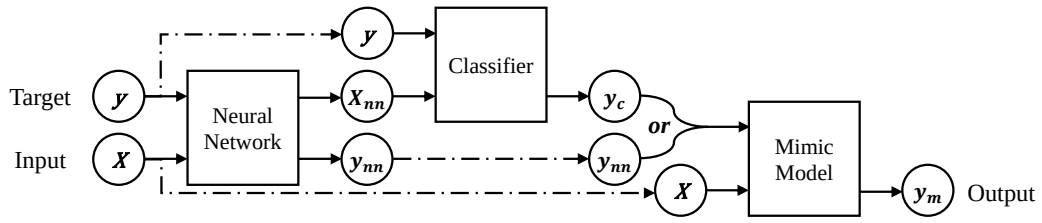


Figure 2: Training Pipeline for Mimic Method

After finishing the training procedure, we can directly apply the mimic model trained in the final step to the original classification task. We compare these two different pipelines to investigate whether utilizing the features extracted from the neural networks (Pipeline 1) will provide more benefits than only taking the soft-labels from the neural networks (Pipeline 2), which is what existing mimic methods usually do. These two pipelines will be evaluated and discussed with detailed experimental results in Section 4.

Our interpretable mimic learning model using GBT has several advantages over existing (mimic) methods. First, gradient boosting trees is good at maintaining the performance of the original complex model such as deep networks by mimicing its predictions. Second, it provides better interpretability than original model, from its decision rules and 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.

4 Experiments

We demonstrate the performance of our interpretable mimic learning framework on a real-world healthcare dataset and compare it to the several methods introduced in the previous section. Our experiments will help us to answer the following questions:

- How does our interpretable mimic learning perform when compared with state-of-the-art deep learning and other machine learning methods?
- What are the interpretable features identified by our mimic learning framework?
- Do soft-labels from top layer of deep networks (Pipeline 2 in Section 3.5) obtain better results than soft-labels of the same networks with Logistic Regression (Pipeline 1 in Section 3.5) for prediction tasks?
- Do static features help in performance improvement in our mimic learning framework?

In the remainder of this section, we will describe the datasets, experimental design and discuss our empirical results and interpretations to answer the above questions.

4.1 Dataset Descriptions

We conducted a series of experiments on *VENT dataset* [25]. This dataset consists of data from 398 patients with acute hypoxemic respiratory failure in the intensive care unit at Children's Hospital Los Angeles (CHLA). It contains a set of 27 static features, such as demographic information and admission diagnoses, and another set of 21 temporal features (recorded daily), including monitoring features and discretized scores made by experts, during the initial 4 days of mechanical ventilation. Two of the time series features start from 0 in time step 0, so when we flatten time series and concatenate all features together, we omit these two 0-valued features and obtain the input feature vector with length $27 + 21 \times 4 - 2 = 109$. The missing value rate of this dataset is 13.43%, with some patients/variables having a missing rate of $> 30\%$. We perform simple imputation for filling the missing values where we take the majority value for binary variables, and empirical mean for other variables. Please see table 1 for a detailed summary of this dataset.

4.2 Experimental Design

We conduct two binary classification tasks on VENT dataset:

Table 1: VENT Dataset Details

Feature type	Number of features	Feature examples
Static features	27	PIM scores, Demographics, Admission diagnosis, etc.
Temporal features	21×4	Injury markers, Ventilator settings, blood gas values, etc.

- Mortality (MOR) task – In this task we predict whether the patient dies within 60 days after admission or not. In the dataset, there are 80 patients with positive mortality label (patients who die).
- Ventilator Free Days (VFD) task – 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. In the dataset, there are 235 patients with positive VFD labels (patients who survive and stay long enough on ventilators).

We report Area Under ROC (AUC) as the evaluation metric to compare proposed and related methods.

4.3 Methods and Implementation Details

We categorize the methods in our main experiments into three groups:

- Baseline machine learning algorithms which are popularly used in the healthcare domain: Linear Support Vector Machine (SVM), Logistic Regression (LR), Decision Tree (DT), and Gradient Boosting Trees (GBT).
- Neural network-based methods (NN-based): Deep Feed-forward Neural Network (DNN), Stack Denoising Autoencoder (SDA), and Long Short-Term Memory (LSTM). Based on the two pipelines of our Interpretable Mimic Learning methods, we have two kinds of NN-based methods:
  - Using the neural network models to directly make classification. We denote these methods as DNN, SDA, and LSTM. (Pipline 2 in Section 3.5)
  - Taking the activations of the highest hidden layers of the networks as the output features, and feeding them into Logistic Regression to obtain final prediction. These methods are denoted by LR-* (LR-DNN, LR-SDA, LR-LSTM) in this section. (Pipline 1 in Section 3.5)
- Our Interpretable Mimic Learning methods: For each of the NN-based methods described above, we take its soft predictions and treat it as the training target of Gradient Boosting Trees. These methods are denoted by GBTmimic-* (E.g., GBTmimic-LSTM, GBTmimic-LR-SDA, etc). As a comparison, we also try Decision Tree (DTmimic-*) as the mimic method.

We train all the algorithms with 5 random trials of 5-fold cross validation. Our DNN and SDA implementations have two hidden layers and one prediction layer. We set the size of each hidden layer twice as large as input size. We do 50 epochs of stochastic gradient descent (SGD) with learning rate 0.001. For LSTM, we only take the time series features to fit this model and do 50 epochs RMSprop [41] training with learning rate 0.001. We stack a prediction layer over the sequence output of LSTM. When we take the output features from the LSTM model, we take the flattened sequence output. We implement all the deep networks in Theano [3] and Keras [10] platforms on a desktop with 4-core CPU and 16GB RAM. For Decision Trees, we expand the nodes as deep as possible until all leaves are pure. For Gradient Boosting Trees, we use stage shrinking rate 0.1 and maximum number of boosting stages 100. We set the depth of each individual trees to be 3, i.e., the number of terminal nodes is no more than 8, which is fairly enough for boosting. We implement all decision tree based methods using the scikit-learn [34] package.

4.4 Quantitative Results

Table 2 shows the prediction performance comparison of the models introduced in Section 4.3. We observe that for both the classification tasks (MOR and VFD tasks), the deep models obtain better performance than standard machine learning baselines; and our interpretable mimic methods obtain similar or better performance than the deep models. Our GBTmimic-LR-SDA and GBTmimic-LR-DNN obtains the best performance in MOR and VFD tasks, respectively. We found that the predictions of DNN and SDA with Logistic Regression is better than just using the deep models, however this is not true for LSTM model. One possible reason is that LSTM captures the temporal dependencies which help in prediction, while in other methods the time series are flattened during input and thus the temporal relations are not efficiently modeled. Similarly, the performance of our interpretable mimic learning methods always improve upon DNN and SDA, and are comparable to LSTM based methods.

Table 2: Classification Results.
AUC: Mean of Area Under ROC;
AUC(std): Standard Deviation of Area Under ROC.

Method		**Task**			
		MOR		**VFD**	
		AUC	**AUC(std)**	**AUC**	**AUC(std)**
Baseline	**SVM**	0.6431	0.059	0.7248	0.056
	LR	0.6888	0.068	0.7602	0.053
	DT	0.5965	0.081	0.6024	0.044
	GBT	0.7233	0.065	0.7630	0.051
NN-based	**DNN**	0.7288	0.084	0.7756	0.053
	SDA	0.7313	0.083	0.7211	0.051
	LSTM	**0.7726**	0.062	0.7720	0.061
	LR-DNN	0.7300	0.084	0.7759	0.052
	LR-SDA	0.7459	0.068	**0.7818**	0.051
	LR-LSTM	0.7658	0.063	0.7665	0.063
Mimic	**GBTmimic-DNN**	0.7574	0.064	**0.7835**	0.054
	GBTmimic-SDA	0.7382	0.084	0.7194	0.049
	GBTmimic-LSTM	**0.7668**	0.059	0.7357	0.054
	GBTmimic-LR-DNN	**0.7673**	0.070	**0.7862**	0.058
	GBTmimic-LR-SDA	**0.7793**	0.066	**0.7818**	0.049
	GBTmimic-LR-LSTM	0.7555	0.067	0.7524	0.060

Based on the observations from the above prediction results in Table 2, and by noticing that LSTM only takes temporal features, we investigated whether time series features themselves are sufficient for our prediction tasks (i.e. we do not consider static features in input vectors). In other words, it is useful to demostrate whether that the temporal models are more relevant than the just static models based on the initial settings.

We conducted two new sets of experiments, 1) with only temporal features as input, and 2) with only static features and the initial values of temporal features at day 0. We present the results of these experiments in Table 3 and Table 4, respectively. From Table 3 we can notice that, for MOR task, the prediction differences between temporal and all features are quite small (i.e. AUC(diff)), while in VFD task, we find that adding static features is relatively more critical to the prediction performance. The different behaviours on these two tasks also explain why LSTM performs better in MOR task than in VFD task. Note that we don't show the LSTM results in Table 3 since we have already used only temporal features for LSTM prediction task and the corresponding results is in Table 2. Results from Table 4 further verified the superiority of the temporal models over just the static model. For both MOR and VFD tasks, the performances of only static variables and initial values of temporal variables degraded significantly on all tested models, and are even worse than the models with only temporal features in Table 3.

Table 3: Classification Results of Input with Only Temporal Features.
AUC: Mean of Area Under ROC;
AUC(diff): AUC of all features (Table 2) - AUC of temporal features (Table 3);
AUC(std): Standard Deviation of Area Under ROC.

Method	Task					
	MOR			VFD		
	AUC	AUC(diff)	AUC(std)	AUC	AUC(diff)	AUC(std)
LR	0.7013	-0.0125	0.064	0.7344	0.0258	0.069
GBT	0.7202	0.0031	0.068	0.7420	0.0210	0.048
DNN	0.7455	-0.0167	0.071	0.7591	0.0165	0.068
SDA	0.7332	-0.0019	0.082	0.7175	0.0036	0.053
LR-DNN	0.7453	-0.0153	0.716	0.7590	0.0169	0.068
LR-SDA	0.7395	0.0064	0.741	0.7622	0.0196	0.053
GBTmimic-DNN	0.7632	-0.0058	0.065	0.7579	0.0256	0.061
GBTmimic-SDA	0.7380	0.0002	0.083	0.7185	0.0009	0.056
GBTmimic-LSTM	**0.7667**	0.0001	0.058	0.7380	-0.0023	0.050
GBTmimic-LR-DNN	0.7613	0.0060	0.055	0.7595	0.0267	0.066
GBTmimic-LR-SDA	0.7574	0.0219	0.060	**0.7726**	0.0092	0.056
GBTmimic-LR-LSTM	0.7565	-0.0010	0.060	0.7263	0.0261	0.058

Table 4: Classification Results of Input with Static and Initial Temporal (at Day 0) Features.
AUC: Mean of Area Under ROC;
AUC(diff): AUC of all features (Table 2) - AUC of static and initial temporal features (Table 4);
AUC(diff2): AUC of temporal features (Table 3) - AUC of static and initial temporal features (Table 4);

Method	Task					
	MOR			VFD		
	AUC	AUC(diff)	AUC(diff2)	AUC	AUC(diff)	AUC(diff2)
LR	0.6797	0.0091	0.0216	0.7158	0.0444	0.0186
GBT	0.6812	0.0421	0.0390	0.7081	0.0549	0.0339
DNN	0.7058	0.0230	0.0397	0.7214	0.0542	0.0377
SDA	0.6981	0.0332	0.0351	0.6919	0.0292	0.0256
LR-DNN	0.7067	0.0233	0.0386	0.7237	0.0522	0.0353
LR-SDA	0.6967	0.0492	0.0428	0.7317	0.0501	0.0305
GBTmimic-DNN	**0.7378**	0.0196	0.0254	0.721	0.0625	0.0369
GBTmimic-SDA	0.6979	0.0403	0.0401	0.6878	0.0316	0.0307
GBTmimic-LR-DNN	0.7305	0.0368	0.0308	0.7235	0.0627	0.0145
GBTmimic-LR-SDA	0.7296	0.0497	0.0278	**0.7322**	0.0496	0.0273

4.5 Interpretations

One advantage of decision tree methods is their interpretable feature selection and decision rules. In this section, we first interpret the trees learned by our mimic framework, and then we compare and contrast trees from our GBTmimic with trees obtained using original GBT.

Table 5 shows the top useful features, found by GBT and our GBTmimic models, in terms of the importance scores among all cross validations. 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. Another interesting finding is that almost all the top features are temporal features, while among all static features, the PIM2 (Pediatric Index of Mortality 2) and PRISM (Pediatric Risk of Mortality) scores, which are developed and widely used by the doctors and medical experts, are the most useful variables.

We can compare and interpret the trees obtained by our Interpretable Mimic learning with the original GBT trees. Figure 3 shows the examples of the most important tree built by the original GBT and our interpretable mimic learning methods on the same cross validation fold for the MOR prediction task. As we can see, they share some common features and similar rules. These selected features and rules can be evaluated and explained by healthcare experts which will help them to understand these models better and to make better decisions on patients.

Table 5: Top Features and Corresponding Importance Scores.
Bold lines refer to the methods with the best classification results in that task.

(a) MOR Task

Model	Features (Importance Scores)			
GBT	MAP-D1(0.052)	PaO2-D2(0.052)	FiO2-D3(0.037)	PH-D3(0.027)
GBT-DNN	MAP-D1(0.031)	δPF-D1(0.031)	PH-D1(0.029)	PIM2S(0.027)
GBT-SDA	OI-D1(0.036)	MAP-D1(0.032)	OI-D0(0.028)	LIS-D0(0.028)
GBT-LSTM	δPF-D1(0.058)	MAP-D1(0.053)	BE-D0(0.043)	PH-D1(0.042)
GBT-LR-DNN	δPF-D1(0.032)	PRISM12ROM(0.031)	PIM2S(0.031)	Unplanned(0.030)
GBT-LR-SDA	**PF-D0(0.036)**	**δPF-D1(0.036)**	**BE-D0(0.032)**	**MAP-D1(0.031)**
GBT-LR-LSTM	δPF-D1(0.066)	PH-D1(0.044)	MAP-D1(0.044)	PH-D3(0.041)

(b) VFD Task

Model	Features (Importance Scores)			
GBT	MAP-D1(0.035)	MAP-D3(0.033)	PRISM12ROM(0.030)	VT-D1(0.029)
GBT-DNN	MAP-D1(0.042)	PaO2-D0(0.033)	PRISM12ROM(0.032)	PIM2S(0.030)
GBT-SDA	LIS-D0(0.049)	LIS-D1(0.039)	OI-D1(0.036)	PF-D3(0.032)
GBT-LSTM	δPF-D1(0.054)	MAP-D1(0.049)	PH-D1(0.046)	BE-D0(0.040)
GBT-LR-DNN	**PaO2-D0(0.047)**	**PIM2S(0.038)**	**MAP-D1(0.038)**	**VE-D0(0.034)**
GBT-LR-SDA	PaO2-D0(0.038)	VE-D0(0.034)	PH-D3(0.030)	MAP-D1(0.030)
GBT-LR-LSTM	PH-D3(0.062)	PaO2-D0(0.055)	δPF-D1(0.043)	MAP-D1(0.037)

Gradient Boosting Trees method has several internal tree estimators. Because of this, it can be more complex and harder to interpret than a single decision tree since a decision tree can be used to find one decision path for a single data sample. So, we compare our GBTmimic-* with DTmimic-* which is obtained by mimicking a single decision tree. From Table 6 we notice that DTmimic-* methods perform poorly compared to GBTmimic-* methods, which is not satisfying even if it (a single decision tree) can be better interpreted and visualized.

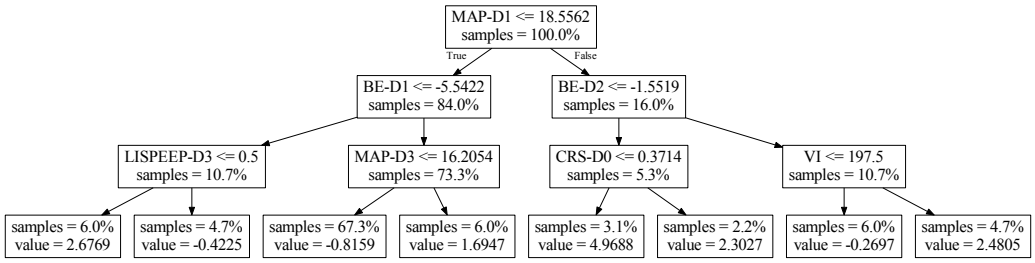
Table 6: Comparison of Mimic Methods with Decision Tree and Gradient Boosting Trees.
AUC: Mean of Area Under ROC;
AUC(diff): AUC of GBTmimic (Table 2) - AUC of DTmimic (Table 6);
AUC(std): Standard Deviation of Area Under ROC.

Method	Task					
	MOR			VFD		
	AUC	**AUC(diff)**	**AUC(std)**	**AUC**	**AUC(diff)**	**AUC(std)**
DTmimic-DNN	0.6683	0.0891	0.079	0.6769	0.1152	0.062
DTmimic-SDA	0.7138	0.0244	0.087	0.7058	0.0056	0.053
DTmimic-LSTM	0.7117	0.0551	0.076	0.6898	0.0240	0.054
DTmimic-LR-DNN	0.6931	0.0742	0.068	0.6992	0.0931	0.048
DTmimic-LR-SDA	0.6994	0.0799	0.072	0.6933	0.0824	0.063
DTmimic-LR-LSTM	0.6976	0.0579	0.075	0.7098	0.0548	0.057

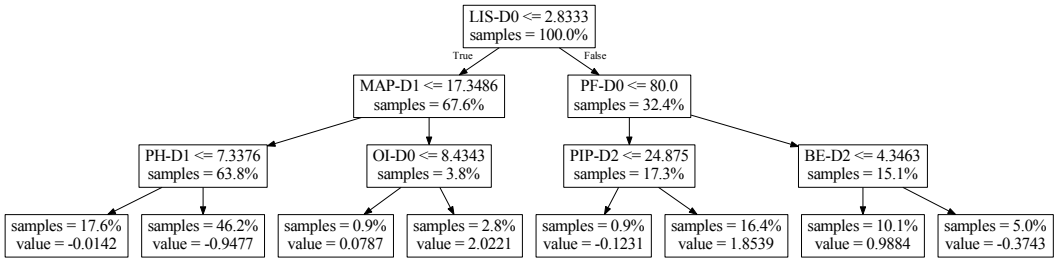
5 Discussions

In this paper, we proposed a novel knowledge distillation approach from deep networks via Gradient Boosting Trees, which can be used to learn interpretable features and prediction rules. Our preliminary experimental results show similar or even better performance from our mimic methods on a real world dataset, and demonstrate a very promising direction for future machine learning research in healthcare domain.

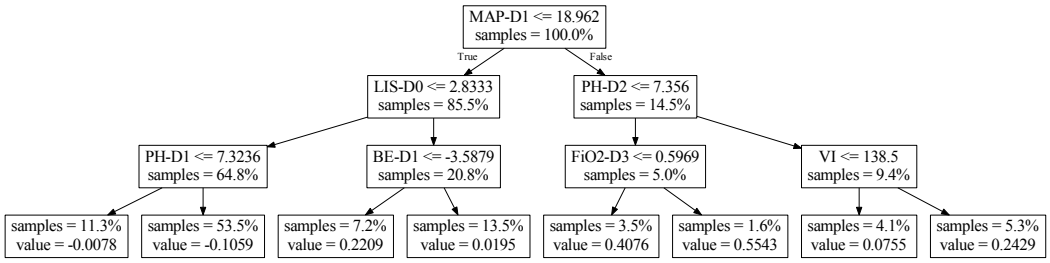
For future work, we aim to extract more useful information like decision rules or tree node features from our mimic methods for better diagnosis interpretability. We will also apply our proposed approaches on a larger healthcare dataset, such as MIMIC-II [38] which is derived from multiple clinical sources, to further verify our methods. We also plan to extend our mimic methods to other state-of-the-art machine learning models, such as structured deep network models, to explore their



(a) GBT



(b) GBTmimic-LR-SDA



(c) GBTmimic-LSTM

Figure 3: Important Decision Trees on MOR Task
Value of a leaf node: The prediction score of a sample from the corresponding decision rules (path).

application abilities in difficult practical applications and help domain experts have a better understanding of these models.

References

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

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

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

[4] A. Belle, M. A. Kon, and K. Najarian. Biomedical informatics for computer-aided decision support systems: a survey. *The Scientific World Journal*, 2013, 2013.

[5] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[6] G. Bonner. Decision making for health care professionals: use of decision trees within the community mental health setting. *Journal of Advanced Nursing*, 35(3):349–356, 2001.

[7] C. Bucilu, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM, 2006.

[8] N. V. Chawla and D. A. Davis. Bringing big data to personalized healthcare: a patient-centered framework. *Journal of general internal medicine*, 28(3):660–665, 2013.

[9] 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*, pages 507–516. ACM, 2015.

[10] F. Chollet. Keras: Theano-based deep learning library. Code: `https://github.com/fchollet`. Documentation: `http://keras.io`, 2015.

[11] F. Dabek and J. J. Caban. A neural network based model for predicting psychological conditions. In *Brain Informatics and Health*, pages 252–261. Springer, 2015.

[12] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

[13] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *Dept. IRO, Université de Montréal, Tech. Rep*, 4323, 2009.

[14] C.-Y. Fan, P.-C. Chang, J.-J. Lin, and J. Hsieh. A hybrid model combining case-based reasoning and fuzzy decision tree for medical data classification. *Applied Soft Computing*, 11(1):632–644, 2011.

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

[16] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[17] M. A. Hamburg and F. S. Collins. The path to personalized medicine. *New England Journal of Medicine*, 363(4):301–304, 2010.

[18] N. Y. Hammerla, J. M. Fisher, P. Andras, L. Rochester, R. Walker, and T. Plötz. Pd disease state assessment in naturalistic environments using deep learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[19] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

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

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

[22] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[23] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306*, 2014.

[24] 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*, page kws210, 2012.

[25] R. G. Khemani, D. Conti, T. A. Alonzo, R. D. Bart III, and C. J. Newth. Effect of tidal volume in children with acute hypoxemic respiratory failure. *Intensive care medicine*, 35(8):1428–1437, 2009.

[26] A. Korattikara, V. Rathod, K. Murphy, and M. Welling. Bayesian dark knowledge. *arXiv preprint arXiv:1506.04416*, 2015.

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

[28] T. A. Lasko, J. C. Denny, and M. A. Levy. Computational phenotype discovery using unsupervised feature learning over noisy, sparse, and irregular clinical data. *PloS one*, 8(6):e66341, 2013.

[29] J. Li, R. Zhao, J.-T. Huang, and Y. Gong. Learning small-size dnn with output-distribution-based criteria. In *Proc. Interspeech*, 2014.

[30] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[32] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[33] A. Oellrich, N. Collier, T. Groza, D. Rebholz-Schuhmann, N. Shah, O. Bodenreider, M. R. Boland, I. Georgiev, H. Liu, K. Livingston, et al. The digital revolution in phenotyping. *Briefings in bioinformatics*, page bbv083, 2015.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[35] 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*, 10(1):52–68, 2003.

[36] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[37] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 833–840, 2011.

[38] M. Saeed, M. Villarroel, A. T. Reisner, G. Clifford, L.-W. Lehman, G. Moody, T. Heldt, T. H. Kyaw, B. Moody, and R. G. Mark. Multiparameter intelligent monitoring in intensive care ii (mimic-ii): a public-access intensive care unit database. *Critical care medicine*, 39(5):952, 2011.

[39] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[40] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[41] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4, 2012.

[42] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[43] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

[44] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.

[45] 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*, volume 2, pages 1099–1103. IEEE, 2005.

[46] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.

[47] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, et al. On rectified linear units for speech processing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3517–3521. IEEE, 2013.