PEARL: Prototype Learning via Rule Learning

Tianfan Fu*
tfu42@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Tian Gao*
tgao@us.ibm.com
IBM Research
Yorktown Heights, USA

Cao Xiao
cao.xiao@iqvia.com
Analytics Center of Excellence,
IQVIA
Cambridge, USA

Tengfei Ma
tengfei.ma1@us.ibm.com
IBM Research
Yorktown Heights, USA

Jimeng Sun
jsun@cc.gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

ABSTRACT

Deep neural networks have demonstrated promising prediction and classification performance on many healthcare applications. However, the interpretability of those models is often lacking. In comparison, classical interpretable models such as decision rule learning do not lead to the same level of accuracy as deep neural networks and can be too complex to interpret (e.g., due to large tree depths). In this work, we present PEARL, Prototype LeArNing via Rule Learning, which iteratively constructs a decision rule list to guide a neural network to learn representative prototypes. The resulting prototype neural network provides accurate prediction, and the prediction can be easily explained by a prototype and its corresponding rules. Thanks to the prediction power of neural networks and interpretability associated with rules, PEARL demonstrates state of the art accuracy to various neural networks baselines and provides simple and interpretable decision rules to explain the prediction. Experimental results also show the resulting interpretation of PEARL is simpler than the standard decision rule list while achieving much higher accuracy.

CCS CONCEPTS

• **Computer systems organization** **Embedded systems**; *Redundancy*; Robotics; • **Networks** Network reliability.

KEYWORDS

Deep Learning; Interpretable machine learning; healthcare

ACM Reference Format:
Tianfan Fu*, Tian Gao*, Cao Xiao, Tengfei Ma, and Jimeng Sun. 2018. PEARL: Prototype Learning via Rule Learning. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The rapid growth of sizes and complexities of electronic health records (EHR) data has motivated the use of deep learning models, which demonstrated state-of-the-art performance in many tasks, including diagnostics disease detection [9, 22, 39], medication prediction [18, 42], risk prediction [11, 40], and patient subtyping [2, 5]. Although deep learning models can produce accurate predictions and classifications, they are often treated as black-box models that lack interpretability and transparency of their inner working [21]. This can limit the adoption of deep learning in medical decision making, since in real clinical practice, clinicians often need to understand why a certain output is produced and how the model generates this output for a given input [26].

Recently, there have been great efforts of trying to explain black-box deep models, including via attention mechanism [9, 41], mimic decisions of deep models with decision tree [6], visualization [32], and explanation with prototype learning [19]. Among them, decision rule learning and prototype learning are two promising directions to achieve model interpretability.

Decision rule or rule list generates a set of rules from training data, in which its prediction is done via checking the conditions in each rule of the rule lists. More specifically, the results of rule learning are rule lists composed of multiple if-then statements [1]. Those rules can be interpretable to domain experts as they are expressed in simple logical forms [4, 31]. However, because of such a simple prediction model, the accuracy of rule-based models is often lower than deep neural networks. Moreover, the interpretability can be undermined as the depth of rules becomes very large and thus incomprehensible to humans with tens or hundreds of levels of the rules.

Prototype learning is another interpretable model inspired by case-based reasoning [16], where observations are classified based on their proximity to a prototype point in the dataset. Many machine learning models have incorporated prototype concepts [3, 14, 29], and learn to compute prototypes (as actual data points or synthetic points) that can represent a set of similar points. However prototypes alone may not lead to interpretability as we may still need an intuitive way to represent and explain what a prototype is, especially given recent deep prototype works [19].

Both approaches were explored in healthcare applications. For example, rule learning was employed to identify how likely patients were to be readmitted to a hospital after they had been released, with each probability associated with a set of rules as decision

criteria [7, 36]. In addition, prototypes have been selected from actual patients and genes for clinicians to make sense of patient or genetic data [3].

However, to our best knowledge no studies focus on combining rule learning and prototype learning, which can provide more interpretable prototypes. To combine them, we need to answer the following questions: How to construct simple rules with accurate prediction performance? How to produce accurate and intuitive definitions of prototypes?

In this work, we propose Prototype LEArNing via Rule Learning (PEARL), which combines decision rule learning and prototype learning on deep neural networks to harness the benefits of both approaches and alleviate their shortcomings for an accurate and interpretable prediction model. In particular, we iteratively learn decision rules, via a data reweighing procedure using prototypes, and then update prototypes via neural networks with learned rules. PEARL not only generates simple and interpretable decision rules and prototypes, but also provides neural network models which can infer the similarity of a query to all the prototypes. To summarize, we make the following contributions in this paper.

(1) We propose an integrative method to combine decision rules and prototype learning, enabling PEARL to harness the power of these methods.
(2) PEARL automatically learns prototypes corresponding to rules in a decision rule list, which are more concise than conventional decision rule learning methods and more explainable than prototype learning methods by providing logic reasoning.
(3) On real-world electronic health record datasets, PEARL demonstrates both accurate prediction and simple interpretation.

2 BACKGROUND

Prototype learning: A prototype is a representative of a set of similar instances (e.g., a patient from a cohort) and can be a part of the observed data points or an artifact summarizing a subset of them with similar characteristics. Prototype learning is one type of case-based reasoning approach [16] to find prototypes [3, 14, 29]. Prototype learning can be seen as an alternative approach to learn centroids of clusters, and have been applied to few shot learning [24, 30, 33, 35]. Let $X = \{x_i\}_{i=1}^n$ be the data set. Various approaches have been4 proposed to to learn a set of P prototypes $\{p_j\}_{j=1}^P$. Some chooses actual data points as prototypes $p_j \in X$ [3]. Some uses a linear combination of the similar data points $p_j = \sum_i^n b_{ij} x_i$ [37], or forms a Bayesian generative model [14]. In this work, we opt to use a very general representation of $p_j = f_j(X)$, where f_j is automatically learned via deep neural networks. In this case p_j has the same dimension as the learned representation of data, which is a predefined hyperparameter.

Rule learning: Decision rule lists (or rule lists for short) are logical statements over original features. A rule list $R = (r_1, r_2, ..., r_K)$ of length K is a K-tuple consisting of K distinct association rules, $r_k := z_k \rightarrow q_k$ for $k = 1, ..., K$. Each rule $r = z \rightarrow q$ is an implication corresponding to the conditional statement, "if z, then q" where z is premise and q is conclusion. In general, rule lists are easy to understand. In this paper, we focus on one particular form of decision rule list, decision tree. Decision trees can be written in forms of a rule list, where each leaf node in a decision tree corresponds to a rule in a rule list. Our rule learning was built upon the decision tree learning method to iteratively guide the prototype learning via neural networks. In addition, we use rule learning methods where each individual rule consists of logic AND clauses but not ORs.

Interpretability: Existing works define 4 aspects of interpretability on decision rules [17]: size, length, cover, and overlap.

(1) *Size*. The size is defined as number of rules K in a rule list R. The fewer the rules in a rule list, the easier it is for a user to understand all of the conditions that correspond to a particular class. In this paper, size is exactly the number of leaf node in decision tree.
(2) *Length.* We use the term length to measure the number of clauses in each rule r_i. If the number of clauses in a rule is too large, it will loose its natural interpretability.
(3) *Cover.* Cover measures the set of data points that satisfy each r_i. Cover measures how the data is divided by the rule classifiers.
(4) *Overlap.* Overlap between rules r_i and r_j is the number of points that satisfy both rules. It measures the discriminative power of each rule and whether decision boundary is clearly defined.

In this paper, we focus on reducing the size (by combining rule in decision trees into prototypes) and the length (by replacing the clauses in each rule using a prototype) of rule classifiers, while improving on the accuracy of decision trees. We propose to use the cover of each rule r_i to re-weight data, which forces rule learning methods to focus on more discriminative data points. By definition, the rules derived from decision tree have no overlap. More detailed are as follows.

3 PEARL: METHODOLOGY

Let $\mathcal{X} = \{X_1, \cdots, X_N\}$ be N data samples with corresponding weight $\{w_1, \cdots, w_N\}$, where each sample X_n (e.g., health records for patient n) is a sequences of discrete event labels (such as medical codes in electronic health records). We can represent X_n as $\{e_n^i, t_n^i\}$, where $e_n^i \in \mathcal{E}$ is the i-th event label in X_n and t_n^i is the time stamp of e_n^i. For each X_n, there is a class label y_n. For example, in health applications, $\boldsymbol{y}$ are the classification result of targeting diseases such as the onset of heart failure (binary), or subtypes of diabetes (multi-class). The goal of PEARL is to accurately predict $\boldsymbol{y} = \{y_1, \cdots, y_N\}$ and to provide explanation for such predictions. We assume both X_n and y_n are categorical variables.

In this work, we aim to accurately predict $\boldsymbol{y}$ given $\mathcal{X}$ while providing an interpretable representation of $\mathcal{X}$ and decision rules with a deep neural network. The outputs of the network include the class label y and a set of interpretable prototypes $\hat{P}$ corresponding to a rule list R. The neural network is used to performing accurate classification, under the guidance of prototypes defined by rule lists generated from decision trees. Formally, the overall objective of PEARL is:

$$
\begin{aligned}
&\arg\min_{\theta_1,\theta_2} \quad \underbrace{\lambda_1 \mathcal{L}_1(\mathbf{h}(\mathcal{X};\theta_1),\hat{P})}_{\text{distance of data to prototypes}} + \underbrace{\lambda_2 \mathcal{L}_2(\mathbf{s}_R(\mathbf{h}(\mathcal{X};\theta_1);\theta_2),\boldsymbol{y})}_{\text{classification error}},\\
&\text{where } \mathcal{L}_1(\mathbf{h}(\mathcal{X};\theta_1),\hat{P}) = \sum_{X_n \in \mathcal{X}} w_n \min_{k\in\{1,\cdots,K\}} d(\mathbf{h}(X_n;\theta_1),\mathbf{p}_k),\\
&\text{and } \mathcal{L}_2(\mathbf{s}_R(\mathbf{h}(\mathcal{X};\theta_1);\theta_2),\boldsymbol{y}) = \sum_{X_n \in \mathcal{X}} w_n \mathcal{L}_2(\mathbf{s}_R(\mathbf{h}(\mathcal{X}_n;\theta_1);\theta_2),\boldsymbol{y_n}),\\
&\text{and} \quad \hat{P} = f(\mathbf{h}(\mathcal{X};\theta_1),R). \quad |\hat{P}| \le |R|,
\end{aligned}
\tag{1}
$$

where $\boldsymbol{h}(\mathcal{X};\theta_1)$ is the learned representation of $\mathcal{X}$ with parameter θ_1. $\boldsymbol{h}(\mathcal{X};\theta_1)$ is a vector and has the same predefined dimension as p_k, R is the learned rule list, and P is the set of learned prototypes. A set of prototypes $\hat{P} = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_K\}$ contains up to K representation of data, which serve as prototypes. $d(\cdot,\cdot)$ is a distance measure; here we use Euclidean distance. f is a fixed mapping that, given R and learned $\boldsymbol{h}(\mathcal{X};\theta_1)$, $\hat{P}$ are determined without further learning. More details on f can be found in Section 3.1.2. Each $\mathbf{p}_k$ lies in the same space as $\mathbf{h}(\mathcal{X};\theta_1)$, and should correspond to one or more rules in R. The second term $\mathcal{L}_2$ is the Cross Entropy loss for the final prediction target, where $\mathbf{s}_R(\mathbf{h}(\mathcal{X};\theta_1);\theta_2)$ represents the predicted label for $\mathcal{X}$ and $\boldsymbol{y}$ is the ground-truth label. Here θ_1 represents all the model parameters for data representation learning $\mathbf{h}(\mathcal{X};\theta_1)$ and θ_2 represents those of classification model $\mathbf{s}_R(\cdot)$. We will drop θs for simplicity from now on. Minimizing $\mathcal{L}_1$ would encourage training examples to be as close as possible to at least one prototype in the latent space, motivated by [19]. However, we do not use other terms from [19] and instead introduce rule lists as the guidance for prototype learning. Note that relative weights λ_1 and λ_2 values are chosen via hyperparameter tuning. In general we chose $\lambda_2 > \lambda_1$ to emphasize the classification performance.

Since it is non-trivial to integrate rule and neural network learning, we propose a framework, PEARL, of integrating rule learning and rule-guided prototype learning together. The main intuition is to learn and produce prototypes that are closely related to rules in R, with one-to-one or many-to-one rule-prototype mapping. This serves as a constraint to make each prototype as a surrogate for clauses in each rule, transforming "if data x satisfies z, then $x = q$" to "if x is close to a prototype p, then $x = q$". We will discuss the network structures in details next.

3.1 Model Architecture

The network architecture of PEARL, illustrated in Fig. 1, mainly comprises two modules: an interpretation module with a decision rule learning procedure, and a prediction module with a prototype learning procedure.

The interpretation module generates a rule list given input data X and passes it to the prediction module. The prediction module consists of a representation network and a prototype learning network. The representation network is made of a temporal modeling component, followed by a highway network with skip connections to alleviate the numerical issue of vanishing gradients [12]. The prototype learning network learns prototypes based on both the rules from the interpretation module and learned representation from the representation network. PEARL then uses prototypes for the final prediction. Moreover, the prediction module also re-weights data per distance to learned prototypes. The re-weighted data is then used to update a new rule list and new data representation, repeating the above procedure until convergence.

Overall, the prediction module iteratively uses rule lists to guide the prototype learning via a neural network. Then the interpretation module iteratively re-weights the data and updates its own rule learning. The two modules are discussed in more details below.

3.1.1 Interpretation Module: Rule Learning. The interpretation module employs a rule list classifier (e.g., a decision tree) to provide interpretable prototype definitions. Given data X_n with weight w_n, we use a known rule list learning algorithm to generate a rule list R, with size $|R|$. In general, any rule list algorithm can be adopted, and we choose weighted decision tree method here. R is then used to help the prediction module to define and interpret prototypes. We will discuss prediction module next and then discuss how interpretation module can benefit from the prediction module in an iterative data re-weighting procedure.

3.1.2 Prediction Module: Neural Network. The prediction module contains a patient representation learning and a prototype learning network.

Data Representation via Neural Networks To encode data such as patient longitudinal clinical events, we first embed the event sequences using neural networks. Although we have flexible choices of neural networks, in this paper we chose the recurrent convolution neural networks (RCNN) [20] to learn distributed representations of each event. In particular, we added one dimension filter and a max-pooling layer in the CNN part, and used a bidirectional LSTM for RNN. This representation learning procedure for patient n is denoted as Eq. 2.

$$\mathrm{g}_n = \mathrm{RCNN}(X_n) = \mathrm{RCNN}([e_n^1, \tau_n^1], [e_n^2, \tau_n^2], \cdots, [e_n^t, \tau_n^t]), \quad (2)$$

where τ_n^k is the time difference between consecutive events, such that $\tau_n^k = t_n^k - t_n^{k-1}$ for $k > 1$ and $\tau_n^0 = 0$. By including τ_n^k as additional features, we incorporate the time information into patient representation learning. After RCNN we also use highway network [34] to alleviate the vanishing gradient issue in network training. A single layer of highway network is:

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W}_H) \odot T(\mathbf{x}, \mathbf{W}_T) + \mathbf{x} \odot (1 - T(\mathbf{x}, \mathbf{W}_T)), \quad (3)$$

where $\mathbf{x}$ and $\mathbf{y}$ are input and output for a single layer, respectively. Here $\odot$ is element-wise multiplication, T is the transform gate, and the dimensionality of $\mathbf{x}, \mathbf{y}, H(\mathbf{x}, \mathbf{W}_H)$, and $T(\mathbf{x}, \mathbf{W}_T)$ are the same. T and H use sigmoid and Relu as activation function, respectively. Multiple layers highway network are concatenated. Given g_n as input of the first layer of highway networks, after multiple layers of updating, we represent the output of the n-th sample as $\mathbf{h}(X_n)$, which can be simplified as

$$\mathbf{h}(X_n) = \text{Highway-Network}(\mathrm{g}_n). \quad (4)$$

Empirically we find the highway networks essential for prototype qualities.

Data representation learning step is not limited to the combination or RCNN and highway network. To generalize this representation learning step, we can write

$$\mathbf{h}(X_n) = \text{Encoder-NN}(X_n), \quad (5)$$

which is the composite of Equation 2 and 4.

Rule-guided Prototype Learning The embedded clinical events $\mathbf{h}(X_n)$ is then used in an iterative prototype learning procedure. Specifically, we first generate prototype vectors from $\mathbf{h}(X_n)$. Given a rule list R, $|R| = K$, for each rule $r_j \in R$, we can find all positive data samples for r_j, denoted as $\mathcal{X}^{(\mathrm{j})}$. Thus we can get a pseudo representation of r_j:

$$\mathbf{p}_j = \frac{1}{\sum_{X_i \in \mathcal{X}^{(\mathrm{j})}} w_i} \sum_{X_i \in \mathcal{X}^{(\mathrm{j})}} w_i \mathbf{h}(X_i), \quad \text{for } j = 1, \cdots, K. \quad (6)$$

Table 1: Notations used in PEARL.

Notation	Definition
E; $e_n^i \in \{1, 2, \cdots \lvert E\rvert\}$	All events; Event i of subject n
$t^i \lvert i \in \{1, \cdots, T\}$	Time stamp for event i
$R = (r_1, r_2, \cdots, r_K, r_0)$	Rule list comprised of K rules, r_0 is the default rule
$X_n = \{e_n^1, t_n^1; e_n^2, t_n^2; \cdots\}$	Event sequence of subject n
$\boldsymbol{y}; y_n$	Labels for all data $\mathcal{X}$; One label for sequence X_n
$\mathcal{L}_1; \mathcal{L}_2$	Loss for prototype similarity; Cross-entropy loss for classification
$\hat{P} = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_K\}, \mathbf{p}_i \in \mathbb{R}^c$	K prototype vectors, prototype layer in network.
$\mathbf{h}(X) \in \mathbb{R}^c$; $\mathbf{s}_R(X)$	Output of highway layer; Output of softmax layer
$\mathbf{o}_R(X) \in \mathbb{R}^K$	Output of prototype layer, subscript R mean it rely on rule list.
$r_i \rightarrow \mathbf{p}_i$	One prototype $\mathbf{p}_i$ corresponds to a rule r_i
$\mathcal{X}$; $\mathcal{X}^{(\mathrm{j})}$	Training subjects; Training subjects that satisfy rule r_j

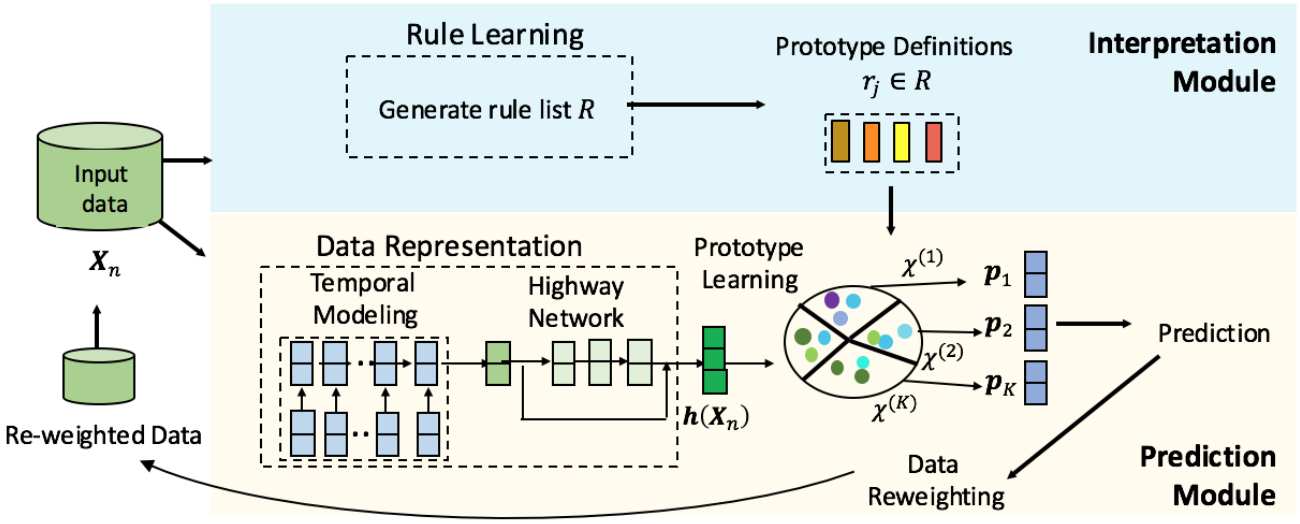

Figure 1: The PEARL architecture includes two modules: an interpretation module with a decisin rule learning procedure, and a prediction module with a prototype learning procedure. Two modules iteratively improves each other during training.

where $\mathcal{X}^{(\mathrm{j})} \subseteq \mathcal{X}$ represent all the data samples that satisfy the j-th rule r_j. $|X^{(\mathrm{j})}|$ represents its cardinality. The output of prototype learning network is a vector of one training subject's distance to all the prototypes, as given by Eq. 7.

$$\mathbf{o}_R(X_n) = \left[\mathrm{d}(\mathbf{h}(X_n), \mathbf{p}_1), \mathrm{d}(\mathbf{h}(X_n), \mathbf{p}_2), \cdots, \mathrm{d}(\mathbf{h}(X_n), \mathbf{p}_K)\right]^\top \in \mathbb{R}^K. \tag{7}$$

Here, $\mathrm{d}(\mathbf{v}_1, \mathbf{v}_2) = \|\mathbf{v}_1 - \mathbf{v}_2\|_2$, is the Euclidean distance of $\mathbf{v}_1$ and $\mathbf{v}_2$. The dimension of $\mathbf{o}(X)$ depends on the number of rules. Since these prototypes use rules as guidance, we also call them rule-prototypes, in contrast to non-rule prototypes in [19]. The subscript R means the function rely on rule list R.

Last, a fully-connected layer (with parameter $\mathbf{W} \in \mathbb{R}^{K \times L}$, where L is number of class) and a softmax activation are used to perform the final classification.

$$\mathbf{s}_R(X_n) = \mathrm{softmax}(\mathbf{W}^\top \mathbf{o}(X_n)), \tag{8}$$

where $\mathbf{s}_R(X_n)$ is the estimated probability. We then used the standard cross-entropy loss for training.

3.1.3 Iterative Data Reweighing. To enable the iterative learning of prototypes and rule list, we use a data re-weighting procedure based on results from the prediction module. We first provide some intuition and then describe the detailed method.

Intuition Since learned prototypes are trained to represent spatially close data samples from the new learned feature space $\mathbf{h}(X)$, prototypes can be more discriminative and can reveal more of the underlying data similarity relationships than the rules from the original feature space as shown in the 2nd diagram of Figure 2. With such a better similarity measure from the representation space, new representations of data samples are more easily separable. More importantly, the examples that are difficult to separate may often be noise or low probability examples, i.e., if $p(x, y)$ be the joint distribution of data, a hard-separable example x_i has low $p(x_i, y_i)$.

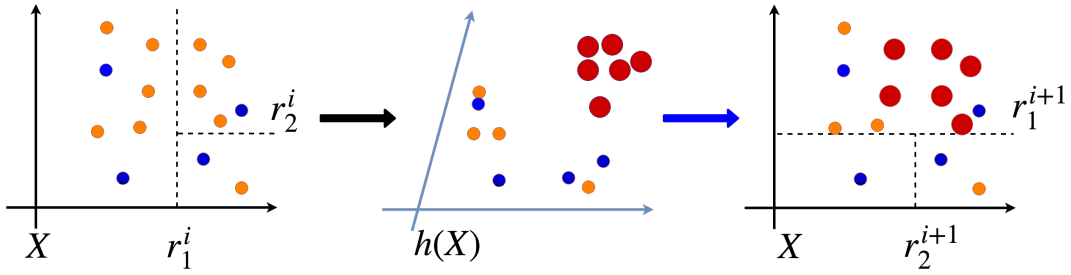

Figure 2: Illustration of the process in which prototypes and decision rule learning affect each other via up-weighting more discriminative samples (shown as bigger red dots). Best viewed in color.

Such a phenomenon has been observed previously in training simpler models [10]. If we up-weight simpler samples that are more separable, rule-list learning focuses these simpler samples more and lead to easier training and more separation later. For examples, the red dots shown in Figure 2 are the high-probability examples, which should be given higher weights. We will also empirically study data separation in experiments to justify this intuition.

Procedure The iterative learning and re-weighing procedure is based on the similarity between data sample (such as patient subjects) and prototypes. To start with, we measure the Euclidean distance between subject $\mathbf{h}(X_n)$ and each prototype vector $\mathbf{p}_k$ as depicted by Eq. 9.

$$s_{nk} = d(\mathbf{h}(X_n), \mathbf{p}_k), \tag{9}$$

where d is the Euclidean distance. We aim at boosting the prototypes that have fewer subjects within its proximal neighbors in the learned representation space, indicating these prototypes are far away from other subjects and hence more discriminative.

For n-th data sample, we use

$$s_n = \sum_k s_{nk} \tag{10}$$

to measure its similarity with prototype. Large s_n indicates n-th data is more separable, thus we should increase its weight.

Weight Normalization Here we set

$$w_n \propto s_n. \tag{11}$$

After that, we do weight normalization by letting its mean equal to 1 (by dividing its mean) and clipping the maximal and minimal weights to be 3 and 0.33, respectively.

Weight Smoothing If the weight changes greatly between neighboring iterations, the decision tree may also change greatly. We want to avoid this case. So after normalization, we smooth the weight by let

$$w_t = \xi w_t + (1 - \xi) w_{t-1}, \tag{12}$$

where ξ is a value between 0 to 1 that balance the new weight (w_t) and the weight in previous iteration (w_{t-1}). For example, we set $\xi = 0.8$ in experiment.

We summarize the procedure in Algorithm 1. We alternately optimize rule list R and prototype networks until convergence. The convergence criteria is when the loss of the current epoch is within a pre-specified threshold from the previous epoch.

Algorithm 1 `PEARL` Prototype Learning via Rule Learning

1: **Input:** Event sequence $X_n = \{e_n^1, t_n^1; e_n^2, t_n^2; \cdots, \}$, label y_n for $n = 1, \cdots, N$. Let $N = |\mathcal{X}|$. Hyperparameter η, maximal iteration number $T_{\max}$. Initial weights are equal to 1, i.e., $w_n = 1$ for $n = 1, \cdots, N$.
2: **for** iter = $1, \cdots, T_{\max}$ **do**
3: *A. weighted decision tree*: Find rule $R = \{r_1, r_2, \cdots, \}$ based on $\{X_1, \cdots, X_N\}$ and corresponding weight $\{w_1, \cdots, w_N\}$.
4: *B. prototype neural network*: Construct and train prototype neural network (Section 3.1), based on $\{X_1, \cdots, X_N\}$ and $R = \{r_1, r_2, \cdots, \}$. Prototype is built based R.
5: *C. Data Reweighing*: Reweigh data according to Eq. 11. Perform weight normalization and smoothing (Eq. 12).
6: **end for**
7: **Output:** rule list $R = \{r_1, r_2, \cdots, \}$, prototype based on R, trained neural network.

Inference Procedure for New Samples For a new subject $X_{\text{new}} = \{e_{\text{new}}^1, e_{\text{new}}^2, \cdots\}$, `PEARL` will generate two outputs. First is the predicted probability for classification, i.e., the output in softmax layer, $\mathbf{s}_R(X_{\text{new}})$ in Eq. 8. Second, we obtain the output of prototype layer, i.e., $o(X_{\text{new}})$. As it indicate the similarity between the current example and prototypes by their Euclidean distance, the new subject can be explained by the characteristics of its closest prototype.

4 EXPERIMENT

4.1 Experimental Setup

We evaluate `PEARL` model by comparing against other baselines on two tasks: heart failure (HF) detection and mortality prediction on two EHR datasets. Models are implemented by Pytorch. All methods are trained on a laptop with 8GB memory.

Dataset Description To evaluate the performance of `PEARL`, we conducted experiments using the following real world datasets. The statistics of the datasets are summarized in Table 2.

Table 2: Basic statistics of datasets. Heart Failure and MIMIC-III (mortality prediction).

Dataset	Heart Failure	MIMIC-III
# cases	2,268	2,825
# controls	14,526	4,712
# visits per patient	19.7	2.7
# clinical variables per patient	41.0	21.6
# unique clinical variables	1,865	942
# clinical variables per visit	2.1	11.6

Heart Failure (HF) Data: The HF dataset is extracted from a proprietary EHR warehouse [1] where subjects were generally monitored over 4 years. The HF cohort includes 2, 268 case patients and 14, 526 matching controls as defined by clinical experts. The criteria for being patients include 1) ICD-9 diagnosis of heart failure appeared in the EHR for two outpatient encounters, indicating consistency in clinical assessment, and 2) at least one medication was prescribed with an associated ICD-9 diagnosis of heart failure. The diagnosis date was defined as its first appearance in the record. These criteria have also been previously validated as part of Geisinger Clinical involvement in a Centers for Medicare and Medicaid Services (CMS) pay-for-performance pilot [28]. For matching controls, a primary care patient was eligible as a control patient if they are not in the case list, and had the same gender and age (within 5 years) and the same PCP as the case patient. More details could be found in [38].
MIMIC-III Data: We use the MIMIC III data for evaluation[2]. MIMIC III was collected on over 58,000 ICU patients at the Beth Israel Deaconess Medical Center from June 2001 to October 2012 [13]. We only included patients with at least two visits in our experiment, resulting in a total of 7,537 ICU patients.

Baselines We consider the following baseline algorithms.

- Decision Tree: The original feature contains temporal information, and decision tree cannot handle this case. Thus, we use *aggregate feature* as the input feature for decision tree. That is, for each patient, we have a fixed size vector that has binary value (0 or 1). The size is equal to number of unique clinical variables. If the i-th clinical variable occurs in any visit of the patient's history, the i-th value is equal to 1 and 0 otherwise. Thus, each clinical variable corresponds to a feature in decision tree and only has 2 states, "exists" or "don't exist". We use scikit [27] package in Python, which support decision tree with weighted samples.
- Prototype Learning (without rules) [19]: RCNN+prototype (without rule). Prototypes are randomly initialized.
- RNN (Doctor-AI) [8]: RNN+softmax. It concatenate multi-hot vector with a difference of time stamp as input feature. A softmax layer is added after bi-LSTM.
- RCNN: CNN+RNN+softmax. RCNN uses 1 dimensional filter, a max-pool layer and bi-LSTM, followed by a softmax layer.

Evaluation Strategies We randomly split dataset 5 times and repeat the experiments 3 independent times with different random seeds. For each split, we divide the dataset into training, validation and testing set in a $7:1:2$ ratio. Then we report the mean and standard deviation of results (both accuracy and run time). To measure the prediction accuracy, we used the area under the receiver operating characteristic curve (ROC-AUC). For rule learning, we report the average results of multiple runs. After tuning, we set $\lambda_1 = 1$ and $\lambda_2 = 1e{-}3$. To initialize embeddings, we use window size of 15 for word2vec [25] and train medical code vectors of 100 dimensions on each training data, following [23]. For prototype learning, we use the same number of prototypes with PEARL to make sure that the parameter numbers are the same. For the RNN model, we implemented a bidirectional-LSTM. For the RCNN model, the number of filters for CNN is 30, stride is 1, and the windows size is 1. We add a max-pooling layer following convolution with pool size $(5, 1)$. PEARL uses the parameter of RNN and CNN in RCNN model to get a warm start. In PEARL, RCNN of previous epoch can be used as warm start for current epoch. For the highway network, the number of layers of highway network is set to 2. Training is done through Adam [15] at learning rate 1e-1. The batch size is set to 256. The threshold in convergence criteria is set as 0.001. All the hyperparameters are chosen via tuning against validation set.

We fix the best model on the validation set within 5 epochs and report the performance in the test set.

4.2 Performance Comparison

Prediction accuracy: We first report the prediction performance. Table 3 shows PEARL has the highest AUC performance among all methods. As for the baseline models, the rule learning methods (decision tree) have the lowest AUC because it classifies based on the composition of simple logics. Prototype learning (RCNN + prototype, but no rules) is better than rule learning but worse than PEARL. It shows PEARL can improve upon both prototype and rule learning.

Training Time: All deep learning methods required similar training time, while PEARL required more time but the overhead is moderate. Decision tree method is fastest but the accuracy is much lower.

Interpretability-Accuracy Tradeoff: We study the relationship between accuracy and the interpretability in rule learning (i.e., decision tree) and the proposed PEARL model. Interpretability is measured by the number of rules of different methods. For PEARL, the number of rules is exactly as the number of leaf nodes in a decision tree. For the decision tree method, we can adjust its maximum depth to control the number of nodes. Figure 3 shows that our method can use a small number of prototypes to achieve better accuracy than the decision tree method. For example, 3 rule-prototypes can already explain more samples than rule lists with over 50 rules. Note that the prototype learning result in Figure 3 is the standard prototype learning without rule learning, which performs worse than PEARL.

Rule learning accuracy as a function of epochs: We also study the accuracy of rules at different iterations in Algorithm 1. We conduct 5 independent trials using different hyperparameters and report their average results, which are shown in Figure 4. We can find that the accuracy of rules increase with iterative learning and we conclude that the data reweighting scheme does improve the accuracy of rules as well.

[1] Data source is anonymized for blind review.
[2] https://mimic.physionet.org/

Table 3: Performance Comparison of Different Methods. More training time (runtime) is measured in terms of seconds. The numbers in parenthesis are the standard deviation.

Model	Heart Failure			MIMIC-III		
	ROC-AUC	Runtime	# param	ROC-AUC	Runtime	# param
Decision Tree	.612(.007)	2.87(.3)	0.4K	0.657(0.008)	0.92(0.09)	0.6K
RCNN + Prototype [19] (no rule)	.668(.003)	768.3(6.3)	18.4K	.761(.010)	178.0(5.7)	18.4K
RNN [8]	.667(.003)	643.0(45.6)	95.8K	.724(.011)	318.0(11.2)	49.7K
RCNN	.682(.009)	718.0(13.7)	8.4K	.766(.009)	144.0(3.4)	8.4K
PEARL with 1 epoch (no reweight)	.674(.004)	843.8(50)	18.8K	.761(.006)	237(8.9)	18.4K
PEARL	.679(0.004)	1012.0(71)		.763(.004)	543(12.3)	

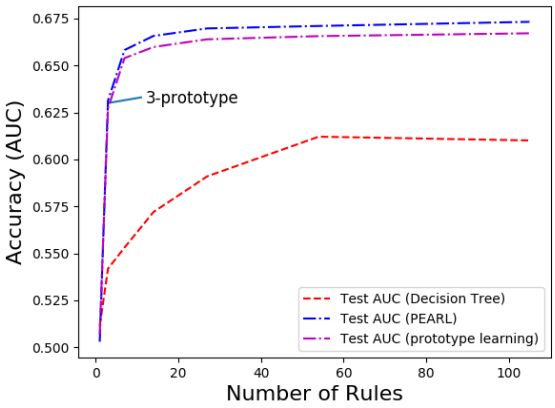

Figure 3: Tradeoff: Test Accuracy v.s. Interpretability (as the number of rules). PEARL is more robust to fewer rules. Conventional rule learning method may require lots of rule.

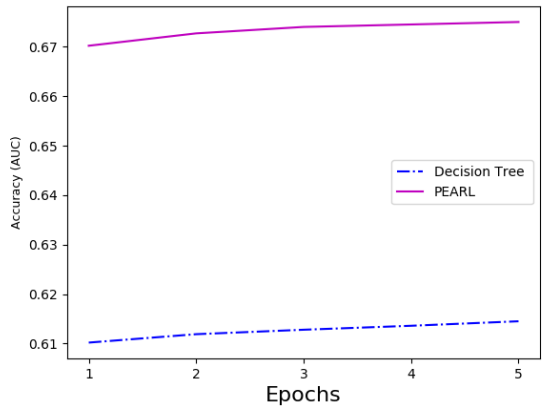

Figure 4: Average test accuracy of decision trees and PEARL in different iterations in Algorithm 1. It shows data reweighing can improve prediction accuracy.

Table 4: Stability of Decision Tree under different weight post-processing strategy. '-' means no any post-processing.

Method	$\text{Sim}(s_{D_k}, s_{D_{k+1}})$	Test Accuracy
-	0.68	0.662
Norm	0.79	0.673
Smooth	0.78	0.671
Norm&Smooth	0.87	0.675

4.3 Evaluation of Stability of Decision Tree

To evaluate the stability of rule lists across iterations of reweighting, we introduce the following quantitative metrics for stability, including **similarity** and **test accuracy**. The first measures if the rules learned from decision tree are stable across iterations, while the second evaluates whether the learned decision tree retains sufficient predictive power.

Similarity We transform the rule list of decision tree into a list of sequences $s = [s_1, \cdots]$. Each leaf node in decision tree corresponds to a sequence. The length of the sequence is the depth of the leaf node. The i-th element of the sequence is the feature of i-th node in the rule list. For example, the decision tree has 3 leaf nodes. And the list of sequences is $s = [d_1 \to d_2 \to d_4, d_1 \to d_2 \to d_5, d_1 \to d_3]$, each sequence corresponds to a path to a leaf node. Now given two decision trees D and E, the corresponding sequence list are $s_D = [s_1, \cdots, s_{D_k}]$ and $s_E = [s'_1, \cdots, s'_{E_k}]$. $\text{Length}(s_i)$ is the length of sequence s_i. $\text{LLCS}(s_i, s'_j)$ is the Length of Longest Common Subsequence between sequences s_i and s'_j. The similarity between two decision trees D (s_D is the corresponding sequence list) and E is defined as

$$\text{Sim}(s_D, s_E) = 0.5 * \Big(\frac{\sum_{s \in s_D} \max_{s' \in s_E} \text{LLCS}(s, s')}{\sum_{s \in s_D} \text{Length}(s)} + \frac{\sum_{s' \in s_E} \max_{s \in s_D} \text{LLCS}(s', s)}{\sum_{s' \in s_E} \text{Length}(s')} \Big) \tag{13}$$

In particular, the similarity will be 1 if two decision trees are identical.

To evaluate the stability of decision trees in PEARL (Algorithm 1), we evaluate the similarity of decision trees for consecutive iterations under different settings (different weight smoothing/normalization settings).

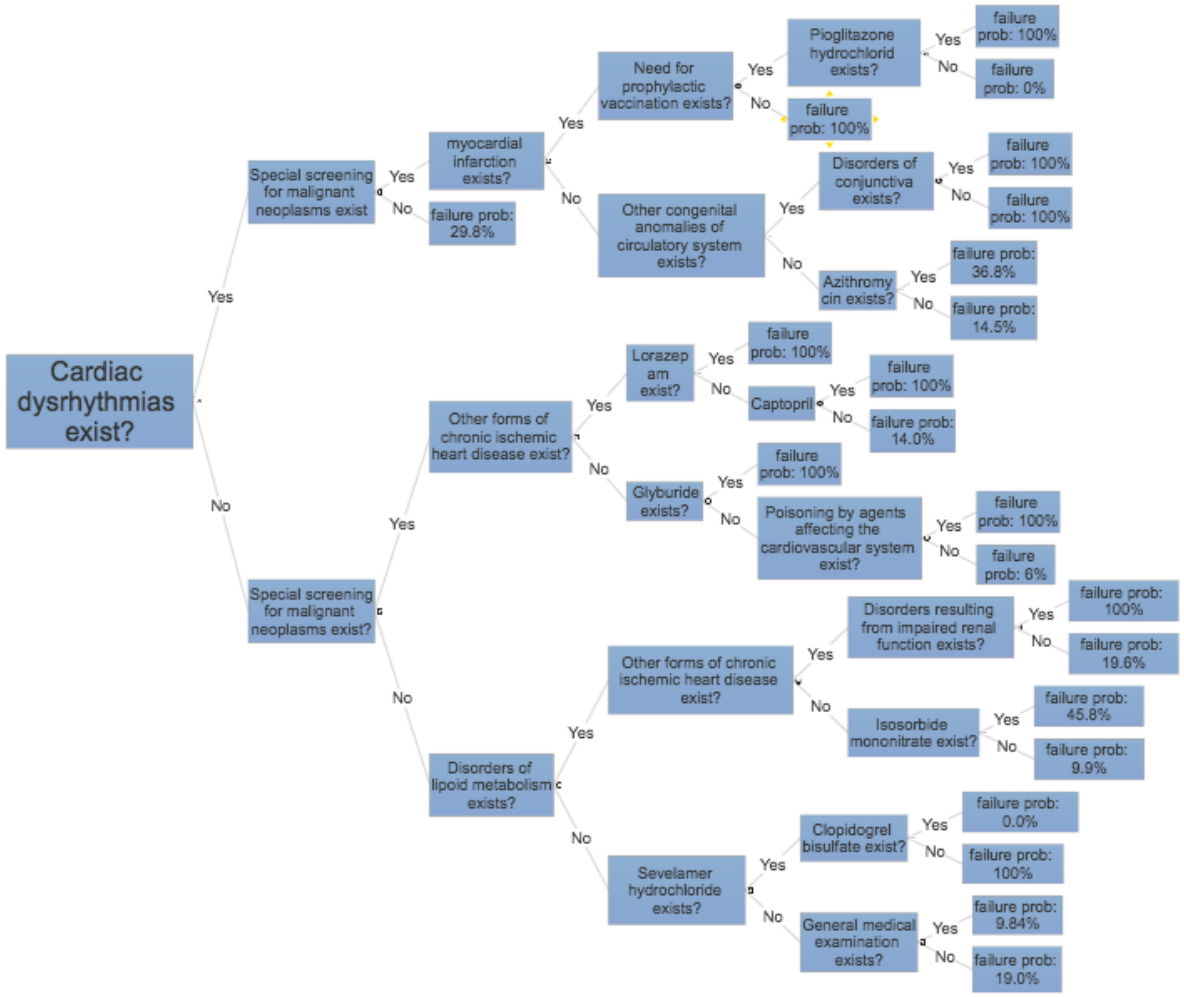

Figure 5: Decision Tree learned by PEARL. Each leaf node has a failure probability.

The results are reported in Table 4. It shows that most of the sequences between consecutive trees are highly overlapped. Especially, the final method with norm and smoothness constraints lead to high similarity hence high stability of the resulting rule sets across consecutive iterations. These results confirmed the consistency and robustness of the proposed method.

4.4 Case Study on Interpretability

We study whether PEARL can provide more interpretable diagnosis compared with conventional rule learning methods. In particular, we find the corresponding prototypes learned in PEARL for a sets of patients and retrieve the closest rule-prototypes, see Figure 6. For each prototype with multiple patients, we retrieve their high frequent events among the patients who satisfy the rule-prototype while the remaining events that only occur to one or two patients are discarded.

In general, the rule learning often yields complex rule lists that involve hundreds of clinical events, many of which are duplicated in multiple rules. As a contrast, PEARL only used ~ 10 rules to make correct diagnosis. Below we provide one example of prototype-rules from PEARL.

If a patient experience all following events, including chronic airways obstruction, malignant neoplasm of trachea, lung and bronchus, carcinoma in situ of respiratory system, Alprazolam, Eszopiclone, abnormal findings on radiological examination of body structure, acute bronchitis, Albuterol Sulfate, Hypertrophic conditions of skin, and diltiazem hydrochloride, then the patient has a high probability of experiencing heart failure.

The prototype-rules include 10 clinical events. Most of them concern severe conditions of lung and respiratory systems (a common symptom of HF patients), and the medications for treating HF, which are common comorbidities of heart failures. Patients belong to this prototype can be diagnosed based on the occurrence of these events on their EHR. For patients of this prototype, if using conventional rule learning, diagnosis would require a much more complex rule with more than 60 clinical events and rule depth for about 6.

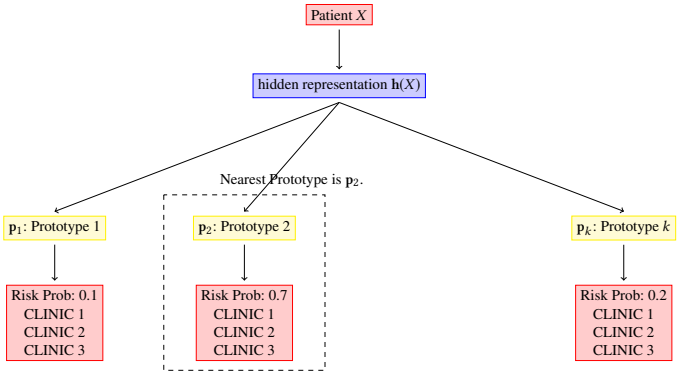

Figure 6: Interpretation Procedure. Given a patient sample, e.g., X, first we compute its hidden representation $\mathbf{h}(X)$. Then we find its nearest prototype, e.g., $\mathbf{p}_2$, so we can see the risk prob, e.g., 0.7 and some high-risk code for that prototype. We can see that the interpretation of our method is simpler than that of decision tree.

High Freq Clinical Code for each prototype

Now we list the high frequent Clinical Code for some prototype (10 codes for each) on Heart Failure Dataset.

- **Prototype 1**: risk prob: 0.19; essential hypertension, general symptoms, symptoms involving respiratory system and other chest symptoms, other and unspecified disorder of joint, diabetes mellitus, osteoarthrosis and allied disorders, need for prophylactic vaccination and inoculation against certain viral diseases, other disorders of soft tissues, other and unspecified disorders of back, special investigations and examinations.
- **Prototype 2**: risk prob: 0.09; general medical examination, essential hypertension, need for prophylactic vaccination and inoculation against certain viral diseases, general symptoms, special investigations and examinations, symptoms involving respiratory system and other chest symptoms, other dermatoses, cataract, other and unspecified disorder of joint, other and unspecified disorders of back.
- **Prototype 3**: risk prob: 0.09; disorders of lipoid metabolism, essential hypertension, diabetes mellitus, acquired hypothyroidism, general symptoms, need for prophylactic vaccination and inoculation against certain viral diseases, other and unspecified disorder of joint, other and unspecified anemias, symptoms involving respiratory system and other chest symptoms, osteoarthrosis and allied disorders.
- **Prototype 4**: risk prob: 0.19; other forms of chronic ischemic heart disease, disorders of lipoid metabolism, essential hypertension, diabetes mellitus, symptoms involving respiratory system and other chest symptoms, need for prophylactic vaccination and inoculation against certain viral diseases, general symptoms, simvastatin, other postsurgical states, clopidogrel bisulfate.
- **Prototype 5**: risk prob: 0.06; special screening for malignant neoplasms, disorders of lipoid metabolism, essential hypertension, general medical examination, special investigations and examinations, need for prophylactic vaccination and inoculation against certain viral diseases, acquired hypothyroidism, diabetes mellitus, other and unspecified disorder of joint, general symptoms.
- **Prototype 6**: risk prob: 0.14; special screening for malignant neoplasms, other forms of chronic ischemic heart disease, disorders of lipoid metabolism, essential hypertension, symptoms involving respiratory system and other chest symptoms, diabetes mellitus, general medical examination, acquired hypothyroidism, need for prophylactic vaccination and inoculation against certain viral diseases, general symptoms.
- **Prototype 7**: risk prob: 0.25; cardiac dysrhythmias, essential hypertension, disorders of lipoid metabolism, general symptoms, symptoms involving respiratory system and other chest symptoms, need for prophylactic vaccination and inoculation against certain viral diseases, other and unspecified aftercare, diabetes mellitus, acquired hypothyroidism, other disorders of urethra and urinary tract.
- **Prototype 8**: risk prob: 0.40; cardiac dysrhythmias, other forms of chronic ischemic heart disease, essential hypertension, disorders of lipoid metabolism, symptoms involving respiratory system and other chest symptoms, general symptoms, diabetes mellitus, other postsurgical states, other and unspecified anemias, need for prophylactic vaccination and inoculation against certain viral diseases.
- **Prototype 9**: risk prob: 0.42; cardiac dysrhythmias, ill-defined descriptions and complications of heart disease, disorders of lipoid metabolism, essential hypertension, symptoms involving respiratory system and other chest symptoms, general symptoms, other forms of chronic ischemic heart disease, other and unspecified aftercare, other postsurgical states, diabetes mellitus.
- **Prototype 10**: risk prob: 0.14; cardiac dysrhythmias, special screening for malignant neoplasms, essential hypertension, disorders of lipoid metabolism, general symptoms, need for prophylactic vaccination and inoculation against certain viral diseases, symptoms involving respiratory system and other chest symptoms, special investigations and examinations, general medical examination, acquired hypothyroidism.

5 CONCLUSION

In this paper, we proposed PEARL, an integrative prototype learning neural network that combines rule learning and prototype learning on deep neural networks to harness the benefits of these methods. We empirically demonstrated that PEARL is more accurate , thanks to an iterative data reweighing algorithm, and more interpretable than rule learning, since it explains diagnostic decisions using much fewer clinical variables. PEARL is an initial attempt to combine traditional rule learning with deep neural networks. In future research, we will try to extend PEARL to other interpretable models.

REFERENCES

[1] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. 2018. Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research* 18, 234 (2018), 1–78.
[2] Inci M Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K Jain, and Jiayu Zhou. 2017. Patient subtyping via time-aware lstm networks. In *SIGKDD*.
[3] Jacob Bien and Robert Tibshirani. 2011. Prototype selection for interpretable classification. *The Annals of Applied Statistics* (2011), 2403–2424.
[4] Leo Breiman. 2017. *Classification and regression trees*. Routledge.

[5] Chao Che, Cao Xiao, Jian Liang, Bo Jin, Jiayu Zho, and Fei Wang. 2017. An RNN Architecture with Dynamic Temporal Matching for Personalized Predictions of Parkinson's Disease. In *SIAM on Data Mining*. SDM.

[6] Zhengping Che, Sanjay Purushotham, Robinder G. Khemani, and Yan Liu. 2016. Interpretable Deep Models for ICU Outcome Prediction. *AMIA Annual Symposium proceedings. AMIA Symposium* 2016 (2016), 371–380.

[7] Chaofan Chen and Cynthia Rudin. 2017. An optimization approach to learning falling rule lists. *CoRR* abs/1710.02572 (2017). arXiv:1710.02572 http://arxiv.org/abs/1710.02572

[8] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. 2016. Doctor ai: Predicting clinical events via recurrent neural networks. In *MLHC*.

[9] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. 2016. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*. 3504–3512.

[10] Amit Dhurandhar, Karthikeyan Shanmugam, Ronny Luss, and Peder Olsen. 2018. Improving Simple Models with Confidence Profiles. *arXiv preprint arXiv:1807.07506* (2018).

[11] Joseph Futoma, Jonathan Morris, and Joseph Lucas. 2015. A Comparison of Models for Predicting Early Hospital Readmissions. *JBI* (2015).

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

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

[14] Been Kim, Cynthia Rudin, and Julie A Shah. 2014. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *Advances in Neural Information Processing Systems*. 1952–1960.

[15] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980

[16] Janet L Kolodner. 1992. An introduction to case-based reasoning. *Artificial intelligence review* 6, 1 (1992), 3–34.

[17] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. 2016. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 1675–1684.

[18] Hung Le, Truyen Tran, and Svetha Venkatesh. 2018. Dual Memory Neural Computer for Asynchronous Two-view Sequential Learning. In *Proceedings of the 24rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1637–1645.

[19] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. 2017. Deep Learning for Case-based Reasoning through Prototypes: A Neural Network that Explains its Predictions. *arXiv preprint arXiv:1710.04806* (2017).

[20] Ming Liang and Xiaolin Hu. 2015. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3367–3375.

[21] Zachary Chase Lipton. 2016. The Mythos of Model Interpretability. *CoRR* abs/1606.03490 (2016). arXiv:1606.03490 http://arxiv.org/abs/1606.03490

[22] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzell. 2016. Learning to diagnose with LSTM recurrent neural networks. In *ICLR*.

[23] Tengfei Ma, Cao Xiao, and Fei Wang. 2018. Health-ATM: A Deep Architecture for Multifaceted Patient Health Record Representation and Risk Prediction. In *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 261–269.

[24] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. 2013. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence* 35, 11 (2013), 2624–2637.

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13)*. Curran Associates Inc., USA, 3111–3119. http://dl.acm.org/citation.cfm?id=2999792.2999959

[26] Detlef Nauck and Rudolf Kruse. 1999. Obtaining interpretable fuzzy classification rules from medical data. *Artificial intelligence in medicine* 16, 2 (1999), 149–169.

[27] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* 12 (Nov. 2011), 2825–2830. http://dl.acm.org/citation.cfm?id=1953048.2078195

[28] M. Pfisterer, P. Buser, H. Rickli, M. Gutmann, P. Erne, and P. Rickenbacher. 2009. BNP-guided vs symptom-guided heart failure therapy. *JAMA: the journal of the American Medical Association.* 301 (2009), 383–392.

[29] Carey E Priebe, David J Marchette, Jason G DeVinney, and Diego A Socolinsky. 2003. Classification using class cover catch digraphs. *Journal of classification* 20, 1 (2003), 003–023.

[30] Oren Rippel, Manohar Paluri, Piotr Dollar, and Lubomir Bourdev. 2015. Metric learning with adaptive density discrimination. *arXiv preprint arXiv:1511.05939* (2015).

[31] Ronald L Rivest. 1987. Learning decision lists. *Machine learning* 2, 3 (1987), 229–246.

[32] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. 2017. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems* 28, 11 (2017), 2660–2673.

[33] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*. 4077–4087.

[34] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387* (2015).

[35] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*. 3630–3638.

[36] Fulton Wang and Cynthia Rudin. 2014. Falling Rule Lists. *CoRR* abs/1411.5899 (2014). arXiv:1411.5899 http://arxiv.org/abs/1411.5899

[37] Chenyue Wu and Esteban G Tabak. 2017. Prototypal analysis and prototypal regression. *arXiv preprint arXiv:1701.08916* (2017).

[38] J. Wu, J. Roy, and WF. Stewart. 2010. Prediction modeling using EHR data: challenges, strategies, and a comparison of machine learning approaches. *Medical Care.* 48 (2010), S106–113.

[39] Cao Xiao, Edward Choi, and Jimeng Sun. 2018. Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. *Journal of the American Medical Informatics Association* (2018).

[40] Cao Xiao, Tengfei Ma, Adji B. Dieng, David M. Blei, and Fei Wang. 2018. Readmission prediction via deep contextual embedding of clinical concepts. *PLOS ONE* 13, 4 (04 2018), 1–15. https://doi.org/10.1371/journal.pone.0195024

[41] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*. ACM, 2048–2057.

[42] Yutao Zhang, Robert Chen, Jie Tang, Walter F Stewart, and Jimeng Sun. 2017. Leap: Learning to prescribe effective and safe treatment combinations for multimorbidity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1315–1324.