# PEARL: PROTOTYPE LEARNING VIA RULE LISTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Deep neural networks have demonstrated promising prediction and classification performance on many healthcare applications. However, the interpretability of those models are often lacking. On the other hand, classical interpretable models such as rule lists or decision trees do not lead to the same level of accuracy as deep neural networks and can often be too complex to interpret (due to the potentially large depth of rule lists). In this work, we present PEARL, Prototype lEArning via Rule Lists, which iteratively uses rule lists to guide a neural network to learn representative data prototypes. The resulting prototype neural network provides accurate prediction, and the prediction can be easily explained by prototype and its guiding rule lists. Thanks to the prediction power of neural networks, the rule lists defining prototypes are more concise and hence provide better interpretability. On two real-world electronic healthcare records (EHR) datasets, PEARL consistently outperforms all baselines across both datasets, especially achieving performance improvement over conventional rule learning by up to $28\%$ and over prototype learning by up to $3\%$. Experimental results also show the resulting interpretation of PEARL is simpler than the standard rule learning.

## 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 and disease detection (Lipton et al., 2016; Choi et al., 2016b; Xiao et al., 2018a), medication prediction (Zhang et al., 2017; Le et al., 2018), risk prediction (Futoma et al., 2015; Xiao et al., 2018b), and patient subtyping (Baytas et al., 2017; Che et al., 2017). 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 (Lipton, 2016). This is a critical problem because clinicians are often unwilling to accept algorithm recommendations without clarity as to the underlying reasoning.

Recently, there have been great efforts of trying to explain black-box deep models, including via attention mechanism (Choi et al., 2016b; Xu et al., 2015), visualization (Samek et al., 2017), explanation by examples or prototypes (Li et al., 2017). To bring deep models into 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. Rule learning and prototype learning are two promising directions to achieve for clinical model interpretability.

Rule learning generates a set of rules from training data, in which its prediction is done at a leaf level via simple models such as majority vote or regression. For example, the results of rule learning are rule lists composed of multiple if-then statements (Angelino et al., 2018). Those rules can be interpretable to domain experts as they are expressed in simple logical forms (Rivest, 1987; Breiman, 2017). 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 for human with tens or hundreds of levels of the rules.

Prototype learning is another interpretable model inspired by case-based reasoning (Kolodner, 1992), where observations are classified based on their proximity to a prototype point in the dataset. Many machine learning models have incorporated prototype concepts (Priebe et al., 2003; Bien & Tibshirani, 2011; Kim et al., 2014), and learn to compute prototypes (as actual data points or syn-

thetic points) that can represent a set of similar points. However prototypes alone may not lead to interpretable models as we still need an intuitive way to represent and explain what a prototype is, especially given recent deep prototype works (Li et al., 2017).

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, each probability associated with a set of rules as criteria (Wang & Rudin, 2014; Chen & Rudin, 2017). While prototype could be selected from actual patients and genes for clinicians to make sense of large patient cohort or gene data (Bien & Tibshirani, 2011). However, there are still open challenges: How to construct simple rules with more accurate prediction and classification performance? How to produce accurate and intuitive definitions of prototypes?

In this work, we propose Prototype lEArning via Rule List (PEARL), which combines 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 rule lists, via a data reweighing procedure using prototypes, and then update prototypes via neural networks with learned rules. PEARL not only generates simple and interpretable rule lists and prototypes, but also provides neural network models which can infer the similarity of a query datum to all the prototypes. As training progresses, we can improve both the predictive power of the learned prototypes and rule list accuracy. To summarize, we make the following contributions in this paper.

1. By combining rule list and prototype learning, PEARL harnesses the power of these methods and is capable to achieve more accurate classification results than both rule list and prototype learning.
2. It automatically learns prototypes corresponding to rules in a rule list, which are more concise than traditional rule list 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 performance and simple interpretation.

## 2 TECHNICAL BACKGROUND

A prototype is an object that is 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 a type of case-based reasoning (Kolodner, 1992) and aims to find some prototypes (Priebe et al., 2003; Bien & Tibshirani, 2011; Kim et al., 2014) Prototypes can be seen as an alternative approach to learn centroids of clusters, and have been applied to few shot learning (Mensink et al., 2013; Rippel et al., 2015; Vinyals et al., 2016; Snell et al., 2017). Let $X = \{x_i\}_{i=1}^n$ be the data set, to learn a prototype existing works choose some $p_j \in X$ Bien & Tibshirani (2011), compute a linear combination $p_j = \sum_i^n b_{ij} x_i$ Wu & Tabak (2017), or form a Bayesian generative model Kim et al. (2014). In this work, we follow Li et al. (2017) and use a general representation of $p_j = f_j(X)$, where $f_j$ is automatically learned via deep neural networks. Current prototype selection methods typically select one prototype at a time, and provide limited higher-level abstraction on the reasoning side of diagnosis.

Rule lists are logic statements over original features. A rule list $R = (r_1, r_2, ..., r_K, r_0)$ of length $K$ is a $(K+1)$-tuple consisting of $K$ distinct association rules, $r_k := z_k \Rightarrow q_k$ for $k = 1, ..., K$ with an additional default rule $r_0$. 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 build on existing rule list learning method Angelino et al. (2018) to iteratively guide the prototype learning via neural networks.

## 3 PEARL: METHODOLOGY

Let $\mathcal{X} = \{X_1, \cdots, X_N\}$ be $N$ data samples, where each $X_n$ consists of a sequences of $\{e_n^i, t_n^i\}$, $e_n^i \in \mathcal{E}$ is the $i$-th event label in $X_n$ and $t_n^i$ is the occurrence time of $e_n^i$. For example, in electronic health records, each $e_n^i$ represents a medical event such as diagnoses and medications of a patient $n$,

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 $r_i$ |
| $\mathcal{X}$; $\mathcal{X}^{(j)}$ | Training subjects; Training subjects that satisfy rule $r_j$ |

and $t_n^i$ is the time stamp of the event $e_n^i$. For each $X_n$, there is a class label $y_n$. We focus on health applications, so $\mathcal{X}$ are training subjects and $\boldsymbol{y}$ are the diagnosis such as the onset of heart failure, the type of pancreatic cancer, or different (sub)-types of diabetes. The goal is to accurately predict $\boldsymbol{y} = \{y_1, \cdots, y_N\}$ as well as providing explanation for such predictions.

In this work, we aim to do so by providing an interpretable representation of data with a deep neural network. The outputs of the network include the class label $y$ and a set of interpretable prototypes corresponding to a rule list. The neural network is used to performing accurate classification, under the guidance of prototypes defined by the rule lists. Formally, the overall objective of PEARL is:

$$\begin{aligned} \arg\min_{\theta} \quad & \underbrace{\lambda_1 \mathcal{L}_1(\mathbf{h}(\mathcal{X}), \hat{P})}_{\text{distance of data to prototypes}} + \underbrace{\lambda_2 \mathcal{L}_2(\mathbf{s}_R(\mathbf{h}(\mathcal{X})), \boldsymbol{y})}_{\text{classification error}}, \\ & \text{where} \quad \mathcal{L}_1(\mathbf{h}(\mathcal{X}), \hat{P}) = \sum_{X_n \in \mathcal{X}} \min_{k \in \{1, \cdots, K\}} \|\mathbf{h}(X_n) - \mathbf{p}_k\|_2^2. \end{aligned} \tag{1}$$

where $\boldsymbol{h}(\mathcal{X})$ is the learned representation of $\mathcal{X}$, $R$ 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 $K$ representation of patients, which serve as prototypes. Each $\mathbf{p}_k$ lies in the same space as $\mathbf{h}(\mathcal{X})$, and should correspond to some rules in $R$. The second term $\mathcal{L}_2$ is the Cross Entropy loss for the final prediction target, where $\mathbf{s}_R(\mathbf{h}(\mathcal{X}))$ represents the predicted label for $\mathcal{X}$ and $\boldsymbol{y}$ is the ground-truth label. Here $\theta$ represents all the model parameters for patient representation learning $\mathbf{h}(\mathcal{X})$ and classification model $\mathbf{s}_R(\cdot)$. 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 Li et al. (2017). However, we do not use other terms from (Li et al., 2017) and instead introduce rule lists as the guidance for prototype learning. Note that relative weights $\lambda_1$ and $\lambda_2$ values are chosen via hyperparameter tuning. In general we chose $\lambda_2 > \lambda_1$ to emphasize the classification performance. We will discuss the network structures in details next.

### 3.1 Model Architecture

The network architecture of PEARL mainly comprises two modules: an interpretable module with a rule list learning procedure, and a prediction module with a prototype learning procedure. 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 illustrated in Fig. 1 and discussed in more details below.

#### 3.1.1 Interpretation Module: Rule Learning

The interpretable module employs a rule list classifier to provide interpretable prototype definitions. Given patient data $X_n$, 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 one recent state-of-the-art COREL (Angelino et al., 2018). $R$ is then used to help the prediction module to define and interpret

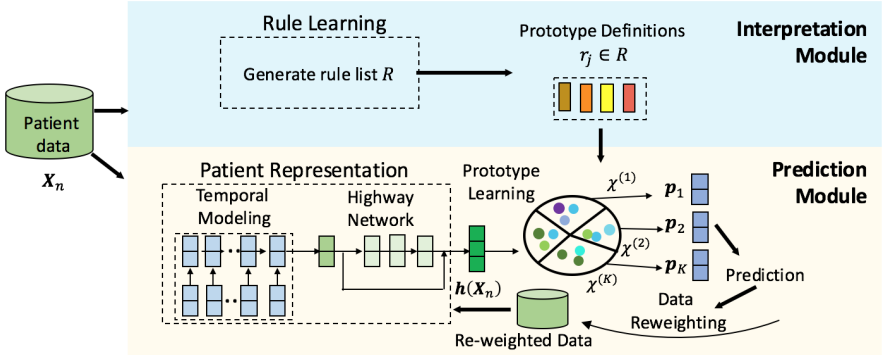


Figure 1: The PEARL architecture includes two modules: an interpretable module with a rule list learning procedure, and a prediction module with a prototype learning procedure. Two modules iteratively improves each other during training.

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.

**Patient Representation via Neural Networks** To encode 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) (Liang & Hu, 2015) to learn the 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.

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

where $\tau_n^k$ 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 $\tau_n^k$ as additional features, we incorporate the time information into patient representation learning. After RCNN we also use highway network (Srivastava et al., 2015) to alleviate the vanishing gradient issue in network training. A single layer of highway network is:

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

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 $\mathrm{g}_n$ 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

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

Empirically we find the highway networks are essential for prototype qualities.

Patient representation learning step is not limited to the combination or RCNN and highway network. To generalize this representation learning step, we can write $\mathbf{h}(X_n) = \text{Encoder-NN}(X_n)$, 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 subjects for $r_j$, denoted as $\mathcal{X}^{(j)}$. Thus we can get a pseudo representation of $r_j$:

$$\mathbf{p}_j = \frac{1}{|\mathcal{X}^{(j)}|} \sum_{X_i \in \mathcal{X}^{(j)}} \mathbf{h}(X_i), \quad \text{for } j = 1, \cdots, K. \tag{5}$$

where $\mathcal{X}^{(j)} \subseteq \mathcal{X}$ represent all the data samples that satisfy the $j$-th rule $r_j$. $|X^{(j)}|$ 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. 6.

$$\mathbf{o}_R(X_n) = [\|\mathbf{h}(X_n) - \mathbf{p}_1\|_2^2, \|\mathbf{h}(X_n) - \mathbf{p}_2\|_2^2, \cdots, \|\mathbf{h}(X_n) - \mathbf{p}_K\|_2^2]^\top \in \mathbb{R}^K. \tag{6}$$

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 contract to non-rule prototypes in (Li et al., 2017). 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.

$$\mathbf{s}_R(X_n) = \text{softmax}(\mathbf{W}^\top \mathbf{o}(X_n)), \tag{7}$$

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 data re-weighting based on results from the prediction module. We first provide some intuition and then describe the detailed method.

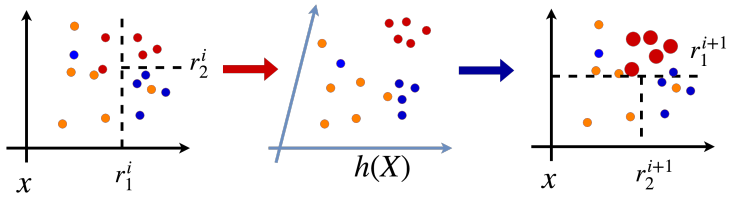


Figure 2: Illustration of the process in which prototypes and rule list learning affect each other via re-weighting red-colored samples. Best viewed in color.

**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 noises or low probability examples, i.e., if $p(x, y)$ be the joint distribution of data, a hard-separable example $x_i$ has low $p(x_i, y_i)$. Such a phenomenon has been observed previously in training simpler models (Dhurandhar et al., 2018). 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 subjects and prototypes. To start with, we measure the cosine similarity between subject $\mathbf{h}(X_n)$ and each prototype vector $\mathbf{p}_k$ as depicted by Eq. 8.

$$s_{nk} = \frac{\mathbf{h}(X_n)^\top \cdot \mathbf{p}_k}{\|\mathbf{h}(X_n)\|_2 \cdot \|\mathbf{p}_k\|_2} \tag{8}$$

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. Thus, for each prototype $k$, we calculate its average similarity with all subjects as $s^k = \frac{1}{N}\sum_{n=1}^{N} s_{nk}$, where $N$ is the size of the current dataset. Then we collect those prototypes, denoted as $\mathcal{K}'$, of which $s^k$ is less than a pre-specified threshold $\eta$ and their corresponding data subjects. We concatenate these samples to the original data $X$.

$$\mathcal{X} \leftarrow [\mathcal{X}, \mathcal{X}^{(j)}] \;\; \forall\, j \in \mathcal{K}', \tag{9}$$

where $\mathcal{X}^{(j)} \subseteq \mathcal{X}$ represent all the data samples that satisfy the $j$-th rule. We summarize the procedure in Algorithm 1. We alternately optimize rule list $R$ and neural network until convergence. The convergence criteria is when the loss of the current epoch is within a pre-specified threshold from the previous epoch. Data augmentation is equivalent to data weighting. For practical purposes where the rule list cannot directly handle data weights, data augmentation can achieve desired results.

**Algorithm 1** PEARL Prototype Learning via Rule lists

**Input:** Event sequence $X_n = \{e_n^1, t_n^1; e_n^2, t_n^2; \cdots, \}$, label $y_n$, corresponding binary vector $\mathbf{c}_n = [0, 0, 1, \cdots]^\top$ for $n = 1, \cdots, N$. Let $N = |\mathcal{X}|$. Hyperparameter $\eta$, maximal iteration number $T_{\text{max}}$.
**Output:** rule list, network classifier.

1: **for** iter = $1, \cdots, T_{\text{max}}$ **do**
2: **A. Rule Learning**:
3: Feature selection to reduce dimension. $\mathbf{c}_n$ is transformed into low-dim $\tilde{\mathbf{c}}_n = [0, 1, \cdots]^\top$.
4: Find rule $R = \{r_1, r_2, \cdots, \}$ based on $\tilde{\mathbf{c}}$. $\mathcal{X}^{(j)} \subseteq \mathcal{X}$ is set of all samples fit the rule $r_j$.
5: **B. prototype + NN training**: Construct and train the neural network (Section 3.1).
6: **C. Data Reweighing**:
7: Compute all $s_{nk}$, i.e., similarity between $n$-th data and $k$-th prototype.
8: Collect all prototypes $k \in \{1, \cdots, K\}$ that have less corresponding subjects ($s^k = \frac{1}{N}\sum_n s_{nk} < \eta$) into $\mathcal{K}'$.
9: Reweigh data according to Eq. 9: $\mathcal{X} \leftarrow [\mathcal{X}, \mathcal{X}^{(j)}]\ \forall\, j \in \mathcal{K}'$. $N = |\mathcal{X}|$.
10: **end for**

**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. 7. 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 $L_2$ 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. All methods are implemented in PyTorch (Paszke et al., 2017) and trained on an MacOS 1.12 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.

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. Subject inclusion criteria is in Appendix.

MIMIC-III Data: We use the MIMIC III (Medical Information Mart for Intensive Care) data for evaluation[2]. MIMIC III is collected on over 58,000 ICU patients at the Beth Israel Deaconess Medical Center from June 2001 to October 2012 (Johnson et al., 2016). 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.

- Rule learning: in this work we used the certifiably optimal rule lists in Angelino et al. (2018).
- Decision Tree: we directly use scikit (Pedregosa et al., 2011) package in Python.
- Prototype Learning (without rules) (Li et al., 2017): RCNN+prototype (without rule). Prototype is randomly initialized. The result is very sensitive to the initialization.

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

Table 2: Basic statistics of datasets

| Dataset | Heart Failure Dataset | MIMIC-III (mortality prediction) |
|---|---|---|
| # 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 |

- RNN (Doctor-AI) (Choi et al., 2016a): 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. All RCNN use 1 dimensional filter, a max-pool layer and bi-LSTM. It is followed by a softmax layer.

**Evaluation Strategies** We randomly split dataset 5 times and repeat the experiments 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 5 trials. After tuning, we set $\lambda_1 = 1$ and $\lambda_2 = 1e{-}3$. To initialize embeddings, we use window size of 15 for word2vec (Mikolov et al., 2013) and train medical code vectors of 100 dimensions on each training data, following (Ma et al., 2018). 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 with hidden layer size 3. 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 $(5, 1)$. For the highway network, the number of layers of highway network is set to 2. Training is done through Adam (Kingma & Ba, 2014) at learning rate 1e-1. The batch size is set to 256. Data weighting threshold $\eta$ is set to values between $.45$ and $.55$. The threshold in convergence criteria is set as $0.001$. We fix the best model on the validation set within 5 epochs and report the performance in the test set.

### 4.2 RESULTS

**Accuracy Comparison** Table 3 shows PEARL has the highest AUC performance among all methods. As for the baseline models, the rule learning has the lowest AUC due to it makes classification based on composition of simple logics. Prototype learning is better than rule learning and RNN models but worse than PEARL. It shows PEARL can improve upon both prototype and rule learning.

Table 3: Performance Comparison of Different Methods. 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 |
| Rule List(Angelino et al., 2018) | .533(.001) | 78.8(.5) | $0.1K$ | .485(.015) | 47.5(.7) | $0.07K$ |
| Decision Tree | .530(.003) | 2.87(.3) | $0.4K$ | 0.657(0.008) | 0.92(0.09) | $0.6K$ |
| Prototype Learning (Li et al., 2017) | .668(.003) | 768.3(6.3) | $18.4K$ | .761(.010) | 178.0(5.7) | $18.4K$ |
| RNN (Choi et al., 2016a) | .636(.008) | 1405.0(27.0) | $95.8K$ | .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 | .672(.006) | 892.0(45.0) | $18.4K$ | .762(.007) | 196.0(6.5) | $18.4K$ |
| PEARL | **.688(.003)** | 1486.0(57.0) | | **.769(.019)** | 289.0(38.4) | |

**Interpretability-Accuracy Tradeoff** We study the relationship between accuracy and the interpretability in rule list learning and the proposed PEARL model. Interpretability is measured by the number of rules (also the number of prototypes) of different methods. Figure 3 shows that our method can use a small number of prototypes to achieve better accuracy than the rule list learning. In fact, 3 rule-prototypes can already explain more samples than rule lists with over $50$ rules.

**Separating distance of Prototypes over Training Iterations** To justify reweighing data points lead to more separable prototypes, we study on whether the mean distances $\frac{\sum_{k=1}^{K}\sum_{n=1}^{N} s_{nk}}{NK}$ between pro-

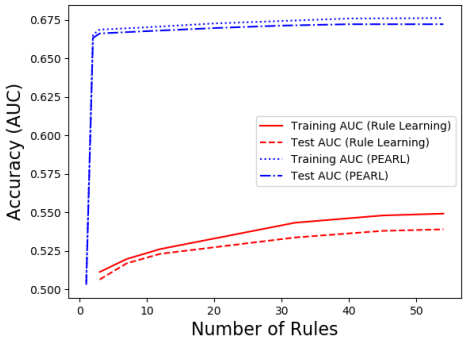


Figure 3: Tradeoff: AUC v.s. Interpretability. PEARL is more robust to number of rules. Conventional rule learning method may require lots of rule and generalize poorly on test dataset.

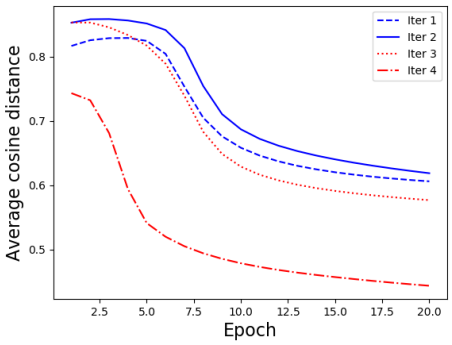


Figure 4: Average Distance between data and prototype during training iterations.

totypes and data decreases over training iteration $T_i$. As shown in Figure 4, by using data weighing, the average distance decreases across iteration $T_i$. Interestingly, even with each iteration, the average distance also decreases with training epochs of neural networks, suggesting that such a reduction in average distances also leads to lower training loss.

We conducted further experiments to test the accuracy change of rule lists during training, which shows that data augmentation helps improving rule list accuracy, along with more hyper-parameter tuning results. All the results are shown in the appendix.

### 4.3 Case Study

We study whether PEARL can provide more interpretable diagnosis compared with conventional rule learning. In particular, we find the corresponding prototypes learned in PEARL for a sets of patients and retrieve the closest rule-prototypes. 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 $\sim 10$ rules to make correct diagnosis. Below we provide one example of prototype-rules from PEARL.

```
If a patient experience the following events,
(1) chronic airways obstruction (2) malignant neoplasm of trachea, lung
 and bronchus (3) carcinoma in situ of respiratory system (4) Alprazolam
(5) Eszopiclone (6) abnormal findings on radiological examination of body
structure (7) acute bronchitis (8) Albuterol Sulfate
(9) Hypertrophic conditions of skin (10) 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 40 clinical events and rule depth for about 50. We provide one example in A.2 in Appendix.

## 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 all 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 less 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


Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research*, 18(234): 1–78, 2018.

Inci M Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K Jain, and Jiayu Zhou. Patient subtyping via time-aware lstm networks. In *SIGKDD*, 2017.

Jacob Bien and Robert Tibshirani. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, pp. 2403–2424, 2011.

Leo Breiman. *Classification and regression trees*. Routledge, 2017.

Chao Che, Cao Xiao, Jian Liang, Bo Jin, Jiayu Zho, and Fei Wang. An rnn architecture with dynamic temporal matching for personalized predictions of parkinson's disease. In *SIAM on Data Mining*, 2017.

Chaofan Chen and Cynthia Rudin. An optimization approach to learning falling rule lists. *CoRR*, abs/1710.02572, 2017. URL http://arxiv.org/abs/1710.02572.

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

Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *NIPS*, 2016b.

Amit Dhurandhar, Karthikeyan Shanmugam, Ronny Luss, and Peder Olsen. Improving simple models with confidence profiles. *arXiv preprint arXiv:1807.07506*, 2018.

Joseph Futoma, Jonathan Morris, and Joseph Lucas. A comparison of models for predicting early hospital readmissions. *JBI*, 2015.

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. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3, 2016.

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

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

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

Hung Le, Truyen Tran, and Svetha Venkatesh. 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*, pp. 1637–1645. ACM, 2018.

Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. *arXiv preprint arXiv:1710.04806*, 2017.

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

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

Zachary Chase Lipton. The mythos of model interpretability. *CoRR*, abs/1606.03490, 2016. URL http://arxiv.org/abs/1606.03490.

Tengfei Ma, Cao Xiao, and Fei Wang. 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*, pp. 261–269. SIAM, 2018.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 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, pp. 3111–3119, USA, 2013. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999792.2999959.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

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. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2078195.

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

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

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

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

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

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

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

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

Fulton Wang and Cynthia Rudin. Falling rule lists. *CoRR*, abs/1411.5899, 2014. URL http://arxiv.org/abs/1411.5899.

Chenyue Wu and Esteban G Tabak. Prototypal analysis and prototypal regression. *arXiv preprint arXiv:1701.08916*, 2017.

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

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

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

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

Yutao Zhang, Robert Chen, Jie Tang, Walter F Stewart, and Jimeng Sun. 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*, pp. 1315–1324. ACM, 2017.

# A DETAILS OF DATASETS

## A.1 INCLUSION CRITERIA FOR HEART FAILURE DATA

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 (Pfisterer et al., 2009). 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 (Wu et al., 2010).

## A.2 EXAMPLE RESULT OF RULE LEARNING

For the diagnosis of patients in the prototype mentioned in Sec.4.3, we need to check all rules below.

```
if (RETINAL DISORDERS=yes,NONSUPPURATIVE OTITIS MEDIA AND EUSTACHIAN TUBE DISORDERS=yes) then (Yes)

else if (WARFARIN SODIUM=yes, CONDUCTION DISORDERS=yes) then (Yes)

else if (DILTIAZEM HYDROCHLORIDE=yes, PULMONARY CONGESTION AND HYPOSTASIS=yes) then (Yes)

else if (AMIODARONE HYDROCHLORIDE=yes) then (Yes)

else if (DIGOXIN=yes,OTHER AND UNSPECIFIED ANEMIAS=yes) then (Yes)

else if (ATHEROSCLEROSIS=yes,CODEINE PHOSPHATE=yes) then (Yes)

else if (ABNORMAL FINDINGS ON EXAMINATION OF BODY STRUCTURE=yes, DIGOXIN=yes) then (Yes)

else if (ASSAULT BY SUBMERSION [DROWNING]=yes,CONDUCTION DISORDERS=yes) then (Yes)

else if (OTHER AND UNSPECIFIED ANEMIAS=yes,CODEINE PHOSPHATE=yes) then (Yes)

else if (ALBUTEROL SULFATE=yes,THEOPHYLLINE ANHYDROUS=yes) then (Yes)

else if (ATHEROSCLEROSIS=yes,CONDUCTION DISORDERS=yes) then (Yes)

else if (WARFARIN SODIUM=yes,ALPRAZOLAM=yes) then (Yes)

else if (ANGINA PECTORIS=yes,ALPRAZOLAM=yes}) then (Yes)

else if (ACUTE BRONCHITIS AND BRONCHIOLITIS=yes,PULMONARY CONGESTION AND HYPOSTASIS=yes) then (Yes)

else if (WARFARIN SODIUM=yes,THEOPHYLLINE ANHYDROUS=yes) then (Yes)

else if (HEARING LOSS=yes,CODEINE PHOSPHATE=yes) then (Yes)

else if (DISEASES OF PERICARDIUM=yes,OTHER BACTERIAL PNEUMONIA=yes) then (Yes)

else if (DISEASES DUE TO VIRUSES AND CHLAMYDIAE=yes,ANGINA PECTORIS=yes}) then (Yes)

else if (DIABETES MELLITUS=yes,PHYSIOLOGICAL MALFUNCTION ARISING FROM MENTAL FACTORS=yes) then (Yes)

else if (CHRONIC AIRWAYS OBSTRUCTION=yes,CODEINE PHOSPHATE=yes) then (Yes)

else if (DISEASES OF PERICARDIUM=yes,OTHER BACTERIAL PNEUMONIA=yes) then (Yes)

else if (DISEASES DUE TO VIRUSES AND CHLAMYDIAE=yes,ANGINA PECTORIS=yes) then (Yes)

else if (DIABETES MELLITUS=yes,PHYSIOLOGICAL MALFUNCTION ARISING FROM MENTAL FACTORS=yes) then (Yes)

else if (CHRONIC AIRWAYS OBSTRUCTION=yes,CODEINE PHOSPHATE=yes) then (Yes)

else if (ALPRAZOLAM=yes,CONDUCTION DISORDERS=yes) then (Yes)

else if (ANGINA PECTORIS=yes,ESZOPICLONE=yes) then (Yes)

else if (DISORDERS OF REFRACTION AND ACCOMMODATION=yes,DISORDERS OF PLASMA PROTEIN METABOLISM=yes) then (Yes)

else if (OLD MYOCARDIAL INFARCTION=yes,MALIGNANT NEOPLASM OF TRACHEA, BRONCHUS, AND LUNG=yes}) then (Yes)

else if (NEOPLASM OF UNCERTAIN BEHAVIOR OF OTHER AND UNSPECIFIED SITES AND TISSUES=yes,OTHER BACTERIAL PNEUMONIA=yes) then (Yes)

else if (OTHER HYPERTROPHIC AND ATROPHIC CONDITIONS OF SKIN=yes,OTHER BACTERIAL PNEUMONIA=yes) then (Yes)

else if (VISUAL DISTURBANCES=yes,PULMONARY CONGESTION AND HYPOSTASIS=yes) then (Yes)

else if (HYDROCORTISONE=yes,CODEINE PHOSPHATE=yes) then (Yes)

else if (NEOMYCIN SULFATE=yes,OTHER HYPERTROPHIC AND ATROPHIC CONDITIONS OF SKIN=yes) then (Yes)
```

```
else if (RETINAL DISORDERS=yes,ALPRAZOLAM=yes) then (Yes)
else if (DIABETES MELLITUS=yes,OTHER POSTSURGICAL STATES=yes) then (Yes)
else if (ANGINA PECTORIS=yes,CONDUCTION DISORDERS=yes) then (Yes)
else if (ANGINA PECTORIS=yes,ATHEROSCLEROSIS=yes) then (Yes)
else if (OTHER DISORDERS OF KIDNEY AND URETER=yes,PULMONARY CONGESTION AND HYPOSTASIS=yes) then (Yes)
else if (OTHER AND UNSPECIFIED ANEMIAS=yes,CONDUCTION DISORDERS=yes) then (Yes)
else if (MALIGNANT NEOPLASM OF TRACHEA, BRONCHUS, AND LUNG=yes,CONDUCTION DISORDERS=yes) then (Yes)
else if (ANGINA PECTORIS=yes,OTHER DISEASES OF PERICARDIUM=yes) then (Yes)
else if (CODEINE PHOSPHATE=yes) then (Yes)
else if (OTHER RETINAL DISORDERS=yes,PHYSIOLOGICAL MALFUNCTION ARISING FROM MENTAL FACTORS=yes) then (Yes)
else if (ESZOPICLONE=yes) then (Yes)
else if (DIABETES MELLITUS=yes,SPECIAL INVESTIGATIONS AND EXAMINATIONS=yes) then (Yes)
else if (DIABETES MELLITUS=yes,OTHER AND UNSPECIFIED DISORDERS OF BACK=yes) then (Yes)
else if (HEARING LOSS=yes,OTHER DERMATOSES=yes) then (Yes)
else if (DISORDERS OF REFRACTION AND ACCOMMODATION=yes,OTHER AND UNSPECIFIED DISORDERS OF BACK=yes) then (Yes)
else if (MALIGNANT NEOPLASM OF TRACHEA, BRONCHUS, AND LUNG=yes,ESZOPICLONE=yes) then (Yes)
else if (DILTIAZEM HYDROCHLORIDE=yes,DIABETES MELLITUS=yes) then (Yes)
else (No)
```

### A.3 RULE LEARNING ACCURACY AS A FUNCTION OF EPOCHS

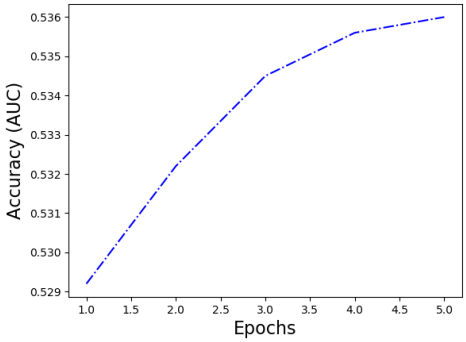


Figure 5: Average rule accuracy on test dataset for different epochs

Here, we study the accuracy of rule during different epochs in Algorithm 1. We conduct 5 independent trials using different hyperparameter and report their average results. The results are shown in Figure 5. We can find that the accuracy of rules increase with iterative learning and we conclude that the data augmentation does improve the accuracy of rule list learning as well.

### A.4 RULE PROTOTYPE VS. NON-RULE PROTOTYPES ON ACCURACY

To study the performance improvement of prototype learning due to impacts of rules, we compare the empirical effect of non-rule prototypes and rule prototypes. As shown in Figure 7, we found that more rule-prototypes can yield better accuracy in general, which shows learned rule-prototypes are better than non-rule prototypes.

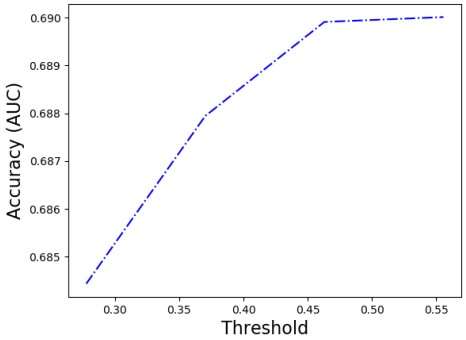


Figure 6: Effect of different threshold $\eta$ in Algorithm 1.

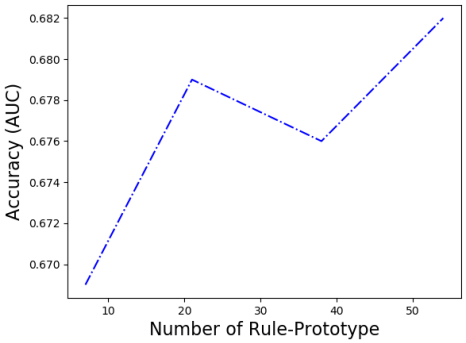


Figure 7: Effect of Fractions of Rule prototype suggest for the iterative learning, more rules can help prototype learning achieve better results.

### A.5 Prototype Rareness v.s. Accuracy

We then study the empirical effect of data reweighing threshold $\eta$ in Algorithm 1, where $\eta$ controls the number of prototypes from which the corresponding subjects are up-weighted. From Figure 6, we find that higher threshold usually corresponds to better accuracy.