Interpretable and Steerable Sequence Learning via Prototypes

Yao Ming*
Hong Kong University of Science and Technology
ymingaa@ust.hk

Panpan Xu
Bosch Research North America
panpan.xu@us.bosch.com

Huamin Qu
Hong Kong University of Science and Technology
huamin@cse.ust.hk

Liu Ren
Bosch Research North America
liu.ren@us.bosch.com

ABSTRACT

One of the major challenges in machine learning nowadays is to provide predictions with not only high accuracy but also user-friendly explanations. Although in recent years we have witnessed increasingly popular use of deep neural networks for sequence modeling, it is still challenging to explain the rationales behind the model outputs, which is essential for building trust and supporting the domain experts to validate, critique and refine the model.

We propose ProSeNet, an interpretable and steerable deep sequence model with natural explanations derived from case-based reasoning. The prediction is obtained by comparing the inputs to a few *prototypes*, which are exemplar cases in the problem domain. For better interpretability, we define several criteria for constructing the prototypes, including *simplicity*, *diversity*, and *sparsity* and propose the learning objective and the optimization procedure. ProSeNet also provides a *user-friendly* approach to *model steering*: domain experts without any knowledge on the underlying model or parameters can easily incorporate their intuition and experience by manually refining the prototypes.

We conduct experiments on a wide range of real-world applications, including predictive diagnostics for automobiles, ECG, and protein sequence classification and sentiment analysis on texts. The result shows that ProSeNet can achieve accuracy on par with state-of-the-art deep learning models. We also evaluate the interpretability of the results with concrete case studies. Finally, through user study on Amazon Mechanical Turk (MTurk), we demonstrate that the model selects high-quality prototypes which align well with human knowledge and can be interactively refined for better interpretability without loss of performance.

CCS CONCEPTS

• **Computing methodologies** → *Neural networks*; *Instance-based learning*;

*This work was done during his internship at Bosch Research North America.

KEYWORDS

Sequence learning; Deep Neural Network; Interpretability

ACM Reference Format:
Yao Ming, Panpan Xu, Huamin Qu, and Liu Ren. 2019. Interpretable and Steerable Sequence Learning via Prototypes. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330908

1 INTRODUCTION

Event sequence data is becoming pervasive in a variety of domains, *e.g.*, electronic health records (EHR) in health care [10, 15], click streams in software applications and vehicle fault logs in automobiles. In general, an event sequence is a series of temporally ordered events. With the advances of machine learning, especially deep learning, we have seen a growing trend of research that applies sequence learning to assist decision-making in these domains. For example, by modeling fault sequences collected from vehicle fleets, we can predict errors that are likely to occur in the future, and thus enable predictive maintenance for car manufacturers and repair workshops, which eventually could improve customer experience and reduce warranty costs [7].

The most widely adopted method for modeling sequential data nowadays is Recurrent Neural Networks (RNNs) and its variants, such as Long Short-Term Memory networks (LSTMs). RNNs have achieved remarkable performance in various sequence modeling applications, *e.g.*, document/text classification [34], machine translation [33] and speech recognition [12]. Despite their superior performance, RNNs are usually considered as "black-boxes" which lack transparency, limiting their application in many critical decision-making scenarios [4]. The demand for more transparent and intelligible machine learning systems is becoming even more urgent as recent regulations in the European Union require "the right to explanation" for algorithms used in individual level predictions [24].

To address this challenge, a variety of methods have been developed to unveil the inner-workings of deep sequence models through visualizing the changes in hidden states [17, 31], extracting feature importance [1, 22, 23] and constructing rules that mimic the behavior of RNNs [5]. However, post-hoc explanations can be incomplete or inaccurate in capturing the reasoning process of the original model. Therefore it is often desirable to have models with inherent interpretability in many application scenarios [29].

We leverage the concept of prototype learning to construct deep sequence model with built-in interpretability. Prototype learning is a form of case-based reasoning [18, 30], which draws conclusions for new inputs by comparing them with a few exemplar cases (i.e.

prototypes) in the problem domain [6, 19]. It is a natural practice in our day-to-day problem-solving process. For example, physicians perform diagnosis and make prescriptions based on their experience with past patients and mechanics predict potential malfunctions by recalling vehicles exhibiting similar symptoms. Prototype learning imitates such human problem-solving process for better interpretability. Recently the concept has been incorporated in convolutional neural networks to build interpretable image classifiers [6, 19]. However, so far prototype learning is not yet explored for modeling sequential data.

We propose prototype sequence network (ProSeNet), which combines prototype learning with variants of RNN to achieve both interpretability and high accuracy for sequence modeling. The RNN as the backbone captures the latent structure of the temporal development. Prediction on a new input sequence is performed based on its similarity to the prototypes in the latent space. For better interpretability, we consider the following criteria in constructing prototypes for explanation:

Simplicity. It is possible to directly use the original sequences in the data as prototypes, but these sequences may contain irrelevant noises. In our approach, the prototypes can be subsequences of the original training data and contain only the key events determining the output. Shorter prototypes are preferred for presenting the explanation in a more succinct form.

Diversity. Redundant prototypes should be avoided since they add to the complexity of the explanation but do not bring extra performance. Therefore we encourage using a set of prototypes that are sufficiently distinct from each other. The prototypes also give a high-level overview of the original data which can be several magnitudes larger.

Sparsity. For each input it is desirable that only a few prototypes are "activated" such that people are not overwhelmed with long and redundant explanations.

We introduce a novel learning objective which takes the above criteria into consideration and propose a training procedure which iteratively performs gradient descent and prototype projection. For steerable learning, we consider a constrained training process with a number of user-specified prototypes which reflect the experts' intuition and experience in the domain.

ProSeNet is evaluated on several real-world datasets and it is able to achieve comparable performance with state-of-the-art deep learning techniques. The experiments cover a diverse range of applications including predictive maintenance of automotives, classification of protein sequences, annotation of electrocardiography (ECG) signals and sentiment analysis on customer reviews, demonstrating the general applicability of the method. In each experiment we not only report classification accuracy on training and test data, but also demonstrate intuitive interpretations of the result through concrete case studies and visualizations. We further study the effect of the number of prototypes k and provide guidelines for selecting k. Besides that, we also perform studies to explore the effect of including the diversity and the simplicity criteria in the model.

To further evaluate the interpretability of the prototypes, we conduct a user study on Amazon Mechanical Turk (MTurk) for a sentiment analysis task on customer reviews, the result shows that ProSeNet is able to select high quality prototypes that are well-aligned with human knowledge on natural languages for sentiment classification. Finally, we demonstrate that through learning under constraints with user-specified prototypes, the model can be steered to obtain comparable performance with better interpretability.

The main contribution of this paper is summarized as follows:

- A sequence model that learns interpretable representations via sequence prototypes for predictive tasks.
- An interaction scheme which allows human experts to incorporate their domain knowledge by validating and updating the learned sequence prototypes.
- Experiments on real-world datasets show that ProSeNet achieves comparable performance with the state-of-the-arts while providing analogy based interpretability.

The rest of the paper is organized as follows: Section 2 summarizes related work; Section 3 introduces the architecture of ProSeNet, the learning objective and the training process; Section 4 presents experimental results; Section 5 concludes the paper.

2 RELATED WORK

Recently, variants of RNNs including Long-Short Term Memory networks (LSTMs) [21] have been proven to be very effective in modeling sequence data. They have been successfully applied to sentiment analysis [34], ECG signal classification [16], mortality and disease risk prediction using EHR data [8, 10, 13], and etc..

Despite their impressive performance, these deep learning models consist of complex nonlinear transformations and are often used as "black boxes", which leads to trust and fairness issues in many applications [4, 24]. Therefore recently we can observe fast expanding literature in interpretable machine learning. In particular, we review two major approaches to interpretable sequence modeling: 1) post-hoc methods, which derive explanations by looking into existing models 2) sequence models with built-in interpretability.

Post-hoc explanation. Post-hoc methods unveil the underlying mechanisms of a pre-existing "black-box" model. Karpathy *et al.* [17] and Strobelt *et al.* [31] visualize the changes of the internal states in RNNs to understand the roles of the hidden units in retaining temporal information. Another popular approach extracts the importance of each input token in determining the final result [1, 22, 23]. Recently, model distillation is also used to explain deep sequence models [5, 23, 26, 27]. The basic idea is to build surrogate models that mimic the behavior of the original one. The surrogate models are usually easier to interpret, shedding light into the inner-workings of more complex models.

Sequence models with inherent interpretability. Built-in interpretability is sometimes desirable since post-hoc explanations usually do not fit the original model precisely[29]. Traditional techniques such as decision trees and logistic regression are considered inherently interpretable. However, they lack the capability of modeling complex temporal dependencies in sequential data, thus the performance is often sub-optimal. State-of-the-art researches focus on building models with both interpretability and high accuracy. One example of such models is RETAIN [9], which uses LSTM with attention mechanism for predictive analysis on patient data. The built-in attention highlights the clinic visits and the diagnostic codes that are most critical for the predictions.

ProSeNet mimics our day-to-day problem-solving process by matching inputs with historical data and producing solutions accordingly. Different from nearest neighbor classifiers used in typical case-based reasoning systems [30, 32], in our approach only a few selected prototypes are memorized, simplified and used for reasoning. There are several benefits in bringing such sparsity: 1) for different inputs it is easier to compare the predictions as well as their interpretations 2) the learned prototypes give a concise overview of the original data, which can be several magnitudes larger 3) it becomes possible to involve human-in-the-loop to update the prototype interactively such that they can incorporate their domain knowledge to further improve the interpretability of the model. Combining prototype-based reasoning with DNNs is first explored for image classification by Li *et al.* and Chen *et al.*[6, 19]. In this paper, we incorporate the concept for predictive analysis on sequential data for the first time.

Evaluating interpretability. There is no universally applicable method to evaluate the interpretability of machine learning models and it is usually use case and model dependent [11]. Quantitative approaches measure the sparsity of the features or the complexity of the model (e.g. number of rules in decision trees). However how these metrics are correlated with human interpretability is still unknown. In this work we evaluate how good the prototypes explain the prediction results based on user studies conducted on MTurk.

3 METHODOLOGY

We introduce the architecture of ProSeNet, formulate the learning objective and describe the training procedure in this section.

3.1 ProSeNet Architecture

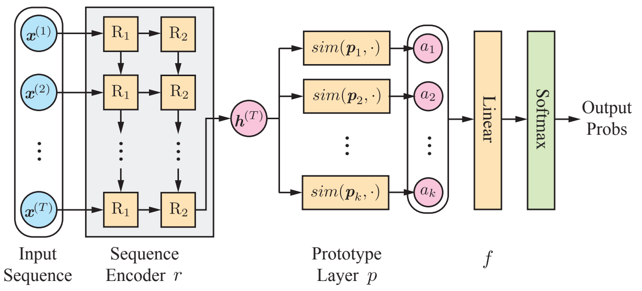


Figure 1: The architecture of our proposed ProSeNet model. The model consists of three parts, the recurrent sequence encoder network r, the prototype layer p that contains k prototypes, the fully connected layer f, and a softmax layer for output probabilities in multi-class classification tasks.

Let $\mathcal{D} = \{((x^{(t)})_{t=1}^T, y)\}$ be a labeled sequence dataset, where T is the sequence length, $x^{(t)} \in \mathbb{R}^n$ is the input vector at step t, and $y \in \{1, \ldots, C\}$ is the label of the sequence. We aim to learn representative prototype sequences (not necessarily exist in the training data) that can be used as classification references and analogical explanations. For a new input sequence, its similarities with each representative sequences are measured in the learned latent space. Then, the prediction of the new instance can be derived and explained by its similar prototype sequences.

The basic architecture of ProSeNet is similar to the one proposed by Li *et al.*[19]. As shown in Figure 1, the model consists of three components: a sequence encoder r, a prototype layer p, and a fully connected layer f.

For a given input sequence $(x^{(t)})_{t=1}^T$, the sequence encoder r maps the entire sequence into a single embedding vector with fixed length $e = r((x^{(t)})_{t=1}^T)$, $e \in \mathbb{R}^m$. The encoder could be any backbone sequence learning models *e.g.*, LSTM, Bidirectional LSTM (Bi-LSTM) or GRU. In our experiments, the hidden state at the last step, $h^{(T)}$, is used as the embedding vector.

The prototype layer p contains k prototype vectors $p_i \in \mathbb{R}^m$, which have the same length as e. The layer scores the similarity between e and each prototype p_i. In previous work [19], the squared L_2 distance, $d_i^2 = \|e - p_i\|_2^2$, is directly used as the output of the layer. To improve interpretability, we compute the similarity using:

$$a_i = \exp(-d_i^2),$$

which converts the distance to a score between 0 and 1. Zero can be interpreted as the sequence embedding e being completely different from the prototype vector p_i, and one means they are identical.

With the computed similarity vector $a = p(e)$, the fully connected layer computes $z = Wa$, where W is a $C \times k$ weight matrix and C is the output size (*i.e.*, the number of classes in classification tasks). To enhance interpretability, we constrain W to be non-negative. For multi-class classification tasks, a softmax layer is used to compute the predicted probability: $\hat{y}_i = \exp(z_i)/\sum_{j=1}^C \exp(z_j)$.

3.2 Learning Objective

Our goal is to learn a ProSeNet that is both accurate and interpretable. For accuracy, we minimize the cross-entropy loss on training set: $CE(\Theta, \mathcal{D}) = \sum_{((x^{(t)})_{t=1}^T, y) \in \mathcal{D}} y \log(\hat{y}) + (1-y)\log(1-\hat{y})$, where Θ is the set of all trainable parameters of the model.

Diversity. In our experiments, we found that when the number of prototypes k is large (*i.e.*, over two or three times the number of classes), the training would often result in a number of similar or even duplicate prototypes (*i.e.*, some prototypes are very close to each other in the latent space). It would be confusing to have multiple similar prototypes in the explanations and also inefficient in utilizing model parameters. We prevent such phenomenon through a diversity regularization term that penalizes on prototypes that are close to each other:

$$R_d(\Theta) = \sum_{i=1}^{k} \sum_{j=i+1}^{k} \max\left(0, d_{min} - \|p_i - p_j\|_2\right)^2,$$

where d_{min} is a threshold that classifies whether two prototypes are close or not. We set d_{min} to 1.0 or 2.0 in our experiments. R_d is a soft regularization that exerts a larger penalty on smaller pairwise distances. By keeping prototypes distributed in the latent space, it also helps produce a sparser similarity vector a.

Sparsity and non-negativity. In addition, to further enhance interpretability, we add L_1 penalty on the fully connected layer f, and constrain the weight matrix W to be non-negative. The L_1 sparsity penalty and non-negative constraints on f help to learn sequence prototypes that have more unitary and additive semantics for classification.

Clustering and evidence regularization. To improve interpretability, Li *et al.*[19] also proposed two regularization terms to be jointly minimized, the clustering regularization R_c and the evidence regularization R_e. R_c encourages a clustering structure in the latent space by minimizing the squared distance between an encoded instance and its closest prototype:

$$R_c(\Theta, \mathcal{D}) = \sum_{(\boldsymbol{x}^{(t)})_{t=1}^{T} \in \mathcal{X}} \min_{i=1}^{k} \left\| r\left((\boldsymbol{x}^{(t)})_{t=1}^{T}\right) - \boldsymbol{p}_i \right\|_2^2,$$

where $\mathcal{X}$ is the set of all sequences in the training set $\mathcal{D}$. The evidence regularization R_e encourages each prototype vector to be as close to an encoded instance as possible:

$$R_e(\Theta, \mathcal{D}) = \sum_{i=1}^{k} \min_{(\boldsymbol{x}^{(t)})_{t=1}^{T} \in \mathcal{X}} \left\| \boldsymbol{p}_i - r\left((\boldsymbol{x}^{(t)})_{t=1}^{T}\right) \right\|_2^2.$$

Full objective. To summarize, the loss that we are minimizing is:

$$\begin{aligned} Loss(\Theta, \mathcal{D}) = CE(\Theta, \mathcal{D}) + \lambda_c R_c(\Theta, \mathcal{D}) + \lambda_e R_e(\Theta, \mathcal{D}) \\ + \lambda_d R_d(\Theta, \mathcal{D}) + \lambda_{l_1} \|\boldsymbol{W}\|_1, \end{aligned} \tag{1}$$

where λ_c, λ_e, λ_d and λ_{l_1} are hyperparameters that control the strength of the regularizations. The configuration of these hyperparameters largely depends on the nature of the data and can be selected through cross-validation. For each experiment in Section 4, we provide the hyperparameter settings.

3.3 Optimizing the Objective

We use stochastic gradient descent (SGD) with mini-batch to minimize the loss function on training data. In this section, we mainly discuss the prototype projection and simplification techniques that we used to learn simple and interpretable prototypes.

Prototype projection. Since the prototype vectors $\boldsymbol{p}_i$ are representations in the latent space, they are not readily interpretable. Li *et al.*[19] proposed to jointly train a decoder that translates the latent space to the original input space to make prototypes interpretable. However, the decoder may not necessarily decode prototypes to meaningful sequences. Instead of using a decoder, we design a projection step during training that assigns $\boldsymbol{p}_i$ with their closest sequence embedding in the training set:

$$\boldsymbol{p}_i \leftarrow \arg\min_{\boldsymbol{e} \in r(\mathcal{X})} \|\boldsymbol{e} - \boldsymbol{p}_i\|_2. \tag{2}$$

Each prototype vector $\boldsymbol{p}_i$ is then associated with a *prototype sequence* in the input space. The projection step is only performed every few training epochs (we set to 4 in our experiments) to reduce computational cost. Compared with the original prototype network [19], the projection step saves the efforts of jointly training a sequence auto-encoder, which is computationally expensive. It also assures each prototype to be an observed sequence, which guarantees that the prototypes are meaningful in the real world.

Interpretation with prototypes. ProSeNet is readily explainable by consulting the most similar prototypes. When making predictions based on a new input sequence, the explanation can be generated along with the inference procedure. A prediction could be explained by a weighted addition of the contribution of the most similar prototypes:

Input:	pizza is good but service is extremely slow
Prediction:	Negative
Explanation:	0.69 * good food but worst service (Negative 2.1)
	+ 0.30 * service is really slow (Negative 1.1)

The factors in front of the prototype sequences are the similarities between the input and the prototypes. At the end of each prototype shows its associated weights $\boldsymbol{w}_i$. The weights can be interpreted as the model's confidence on the possible labels of the prototype.

Prototype simplification. Although the prototypes are already readable after projecting to observed sequences in the training data, it may still be difficult to comprehend a prototype sequence if it contains insignificant or irrelevant noisy events.

Next, we introduce a procedure to simplify the projected prototype sequences. That is, instead of projecting a prototype to a complete observed sequence, we project it to a subsequence containing the critical events. The projection step (Equation 2) now becomes:

$$\begin{aligned} \boldsymbol{p}_i &\leftarrow r(seq_i), \\ seq_i &= \arg\min_{seq \in sub(\mathcal{X})} \left(\|r(seq) - \boldsymbol{p}_i\|_2 \right), \end{aligned} \tag{3}$$

where $sub(\mathcal{X})$ is the set of all possible subsequences of the data in $\mathcal{X}$, $|\cdot|$ computes the effective length of the subsequence. Note that the complexity of the above operation is $O(2^T N)$, where N is the size of training set and T is the maximum length of the sequences in $\mathcal{X}$. The cost of the brute-force computation grows exponentially with T, which is unacceptable even for relatively short sequences.

We use beam search to find an approximate solution [28]. Beam search is a greedy breadth-first search algorithm which only keeps w best candidates in each iteration. w is called the beam width. The algorithm first selects w closest candidate sequences to prototype $\boldsymbol{p}_i$. Then it generates all the possible subsequences which can be obtained by removing one event from any of the w candidates. The score in Equation 3 is calculated for each subsequence. The w subsequences with the minimum scores are then kept as candidates to continue the search in the next iteration. The subsequence with the minimum score is the output. The complexity of the algorithm is now $O(w \cdot T^2 N)$. We use $w = 3$ in our experiments.

3.4 Refining ProSeNet with User Knowledge

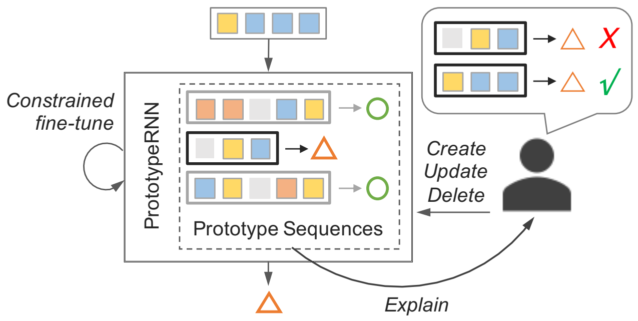


Figure 2: A user verifying and refining a ProSeNet with his/her knowledge and observation on the model output.

Next, as illustrated in Figure 2, we discuss how users can refine a ProSeNet for better interpretability and performance by validating and updating the prototypes, esp. when they have certain expertise or knowledge in the problem domain. Allowing users to validate and interact with the prototypes can also increase their understanding of the model and the data, which is the foundation of user trust [26].

We assume that the knowledge of a user can be explicitly expressed in the form of input-output patterns which the user recognizes as significant or typical in the domain (*e.g.*, "food is good" is typically a review with "positive" sentiment). These patterns can be regarded as the "prototypes" that the user learned from his/her past experiences. The refinement can thus be done by incorporating user-specified prototypes as constraints in the model.

Based on the users' past knowledge and observation on the model outputs, there are three types of possible operations that they can apply to the model: *create* new prototypes, *revise* or *delete* existing prototypes. After changes are committed, the model is fine-tuned on the training data to reflect the change.

When fine-tuning the model the prototypes should be fixed to reflect the users' constraints. Therefore we make the following revisions to the optimization process described in Section 3.3: 1) instead of updating the latent prototype vectors $\boldsymbol{p}_i$ in the gradient descent step, we use the updated sequence encoder r in each iteration to directly set $\boldsymbol{p}_i = r(seq_i)$; 2) the prototype projection step is skipped. After fine-tuning, the sequence encoder r learns better representations of the data. The user can verify the updated results and repeat the process until he/she is satisfied with the result.

4 EXPERIMENTAL EVALUATION

In this section, we evaluate ProSeNet for classification tasks on four real-world sequence datasets. Besides performance metrics, we also evaluate the interpretability of ProSeNet with both qualitative case studies and quantitative experiments with human users. We also perform ablation studies to understand how the following factors affect the performance: the prototype number k, the diversity regularization term and the prototype simplification step.

We implemented ProSeNet[1] using PyTorch[2]. We use stochastic gradient descent (SGD) for the training of all models. We clip the L_2 norm of the gradients at 5 to prevent exploding gradient during the training [25]. The learning rate is set to 1.0 for the first 10 epochs, and is decayed with a factor of 0.85 for each epoch afterwards.

4.1 Case Study 1: Predictive Diagnostics based on Vehicle Fault Log Data

Today's vehicles have complex interconnected modules and the faults usually have a significant history of development over a vehicle's lifetime. Fault logs collected from cars can therefore be used to understand the typical development paths of the problems and support predictive diagnostics. The fault log of each vehicle can be modeled as a sequence of events. Each event corresponds to one or multiple faults that happen at the same time. Each fault is described with a five-digit Diagnostic Trouble Code (DTC) which is standard across different makes and models. With ProSeNet, we aim to predict the risk of faults (*i.e.* DTCs) for a vehicle in the future using its historical DTC logs. We encode an event as a multi-hot vector since multiple faults could occur at the same time. The input at each step is therefore a binary vector $\boldsymbol{x}^{(t)} \in \{0,1\}^n$ and each element in the vector indicates if a particular fault has occurred. The problem is formulated as multi-label classification to predict the risk of different DTCs. The softmax layer is replaced with a sigmoid layer to compute the output probabilities.

In total there are 12k vehicle fault sequences containing 393 different types of DTCs. We train the classifier to predict the top 92 DTCs which have occurred more than 100 times in the dataset. The sequences have an average length of 2.31. The dataset is split into 7.2k training, 2.4k validation, and 2.4k test set. We train a ProSeNet with an LSTM encoder (1 layer, 50 hidden units) and 100 prototypes. We set $\lambda_{l_1} = 1.0, \lambda_e = 0.1, \lambda_c = 0.01, \lambda_d = 0.01, d_{min} = 1.0$ during the training. For prototype simplification, we set the beam width $w = 3$. We use recall at 5 (Recall@5) and mean average precision at 5 (MAP@5) as performance measures.

We compare the performance of ProSeNet with a standard LSTM with the same number of layers and hidden units (Table 1). Both models are trained for 24 epochs with a batch size of 64.

Table 1: Performance on vehicle fault risk prediction.

Model	Recall@5 (%)	MAP@5
ProSeNet	0.473	0.759
LSTM	0.479	0.751

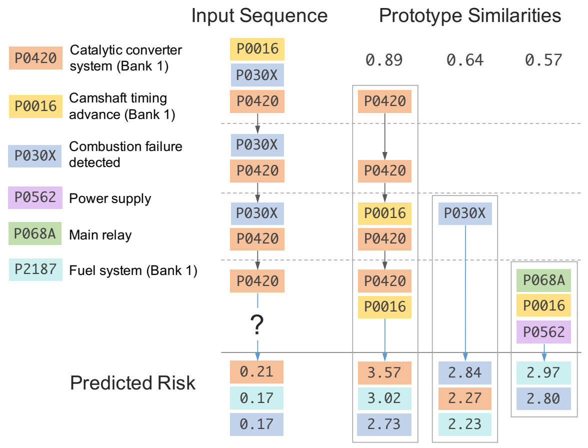


Figure 3: An input sequence and the similarity scores with its closest prototypes. The weights w_i are visualized at the bottom as the outcome of the prototype sequences.

An example prediction of the model on an input fault log sequence is shown in Figure 3. The input sequence shows a recurring sequence consisting of "P030X" and "P0420", while the model predicts a relatively high risk (0.17) of "P2187" which has not occurred before. "P2187" indicates a problem of the fuel system in the engine at Bank 1. With the given explanation, we can see that there are three prototypes that match different aspects of the input sequence. All of the three prototypes indicate a high risk of "P2187", which explains the reasons of the prediction. A mechanic could utilize the model to predict potential future problems and ground the predictions on exemplar cases. The entire set of prototypes in the model provides an overview of all the fault development paths, which can help manufacturers identify systematic issues and develop preventive maintenance strategies.

[1]Code available for research purposes at `https://github.com/myaooo/ProSeNet`
[2]https://pytorch.org/

4.2 Case Study 2: Sentiment Analysis

We also evaluate ProSeNet using a sentiment classification task on text data. We use the reviews of restaurants in the Yelp Open Dataset[3]. Each review is tokenized into a sequence of words using NLTK[4]. We only use reviews that are less than 25 words in the experiments (106k reviews in total) since in later user study (Section 4.6) shorter sentences are easier for humans to read and compare. We use the stars (one to five) given with the reviews as labels and conduct experiments on both *fine-grained* (5-class) and *binary* (positive=rating ≥ 3) classifications. The dataset is split into 60% training, 20% validation, and 20% test set.

Table 2: Average accuracy (%) of LSTM, ProSeNet and ResNet on 10 random train-test splits of Yelp Reviews. Numbers in parentheses are standard deviations.

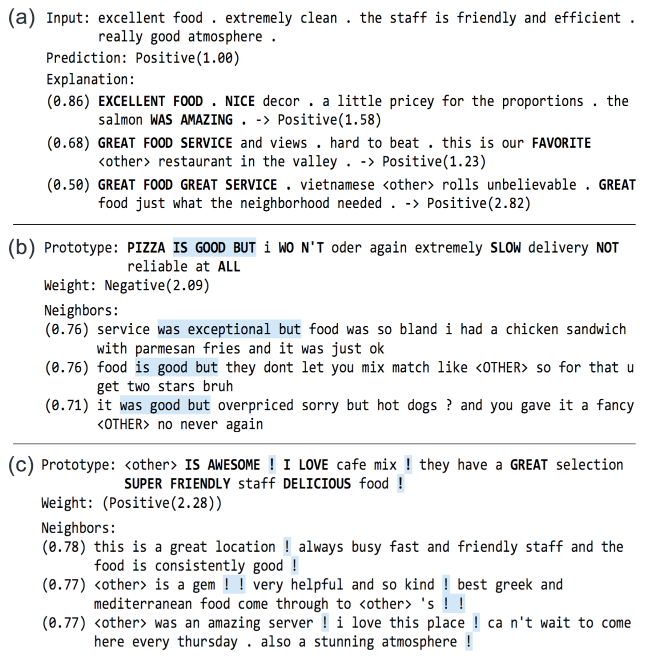
Model	Binary	Fine-grained
ProSeNet	95.5 (0.1)	60.7 (0.2)
ProSeNet$_{\text{Bi-LSTM}}$	95.5 (0.2)	61.0 (0.3)
LSTM	95.6 (0.1)	60.7 (0.3)
Bi-LSTM	95.7 (0.1)	61.1 (0.3)
ResNet	95.3 (0.3)	60.6 (0.4)

As shown in Table 2, we report the accuracy of ProSeNet, ProSeNet with Bi-LSTM encoder, LSTM, Bi-LSTM, and ResNet on both validation and test set. All LSTMs have 2 layers, with 100 hidden units per layer. The ResNet contains 7 residual blocks similar to the architecture mentioned in [14]. We apply a dropout rate of 0.8 during training. The initial number of prototypes is set to 100 and 200 in binary and fine-grained classification tasks respectively. The result shows that our model can learn interpretable representations while achieving similar, though slightly lower performance compared with the state-of-the-art bi-directional LSTMs.

Figure 4 (a) shows the explanation of an example sentiment analysis result. The referenced prototypes show the different aspects of a good restaurant — good food and service. Some other prototype sequences and their neighboring sequences are presented in Figure 4 (b)(c). We can see that some prototypes represent frequent short phrases that are sentimentally significant. Some prototypes capture long-range semantics, such as the transition of sentiments via contrastive conjunctions (*e.g.*, but in Figure 4 (b)). We also discovered some interesting sequential "patterns" of how people express their sentiments. For example, a typical way of expressing positive sentiment is through multiple short compliments ended with exclamation marks (Figure 4 (c)). Note that the input sequences and the prototype sequences are matched through a learned distance measure through an LSTM rather than strict pattern matching.

4.3 Case Study 3: UniProtKB Protein Sequence Classification

We evaluate ProSeNet in the biology domain using the UniProtKB database. The database contains 558,898 protein sequences manually annotated and reviewed. Protein sequences are composed of 20 standard amino acids and can be grouped into families. Proteins in a family descend from a common ancestor and typically have similar

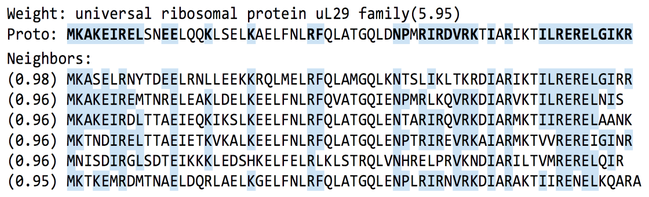


```
(a) Input: excellent food . extremely clean . the staff is friendly and efficient .
           really good atmosphere .
    Prediction: Positive(1.00)
    Explanation:
    (0.86) EXCELLENT FOOD . NICE decor . a little pricey for the proportions . the
           salmon WAS AMAZING . -> Positive(1.58)
    (0.68) GREAT FOOD SERVICE and views . hard to beat . this is our FAVORITE
           <other> restaurant in the valley . -> Positive(1.23)
    (0.50) GREAT FOOD GREAT SERVICE . vietnamese <other> rolls unbelievable . GREAT
           food just what the neighborhood needed . -> Positive(2.82)

(b) Prototype: PIZZA IS GOOD BUT i WO N'T oder again extremely SLOW delivery NOT
               reliable at ALL
    Weight: Negative(2.09)
    Neighbors:
    (0.76) service was exceptional but food was so bland i had a chicken sandwich
           with parmesan fries and it was just ok
    (0.76) food is good but they dont let you mix match like <OTHER> so for that u
           get two stars bruh
    (0.71) it was good but overpriced sorry but hot dogs ? and you gave it a fancy
           <OTHER> no never again

(c) Prototype: <other> IS AWESOME ! I LOVE cafe mix ! they have a GREAT selection
               SUPER FRIENDLY staff DELICIOUS food !
    Weight: (Positive(2.28))
    Neighbors:
    (0.78) this is a great location ! always busy fast and friendly staff and the
           food is consistently good !
    (0.77) <other> is a gem ! ! very helpful and so kind ! best greek and
           mediterranean food come through to <other> 's ! !
    (0.77) <other> was an amazing server ! i love this place ! ca n't wait to come
           here every thursday . also a stunning atmosphere !
```

Figure 4: Examples of binary sentiment classification on Yelp Reviews. (a) The generated explanation of a prediction. (b) and (c): Prototypes and their neighboring sentences. Similar patterns between prototypes and neighboring sequences are manually highlighted in blue. The numbers show the similarities between the input and prototypes. Bold and uppercase texts show the simplified prototype sequences.



```
Weight: universal ribosomal protein uL29 family(5.95)
Proto: MKAKEIRELSNEELQQKLSELKAELFNLRFQLATGQLDNPMRIRDVRKTIARIKTILRERELGIKR
Neighbors:
(0.98) MKASELRNYTDEELRNLLEEKKRQLMELRFQLAMGQLKNTSLIKLTKRDIARIKTILRERELGIRR
(0.96) MKAKEIREMTNRELEAKLDELKEELFNLRFQVATGQIENPMRLKQVRKDIARVKTILRERELNIS
(0.96) MKAKEIRDLTTAEIEQKIKSLKEELFNLRFQLATGQLENTARIRQVRKDIARMKTIIRERELAANK
(0.96) MKTNDIRELTTAEIETKVKALKEELFNLRFQLATGQLENPTRIREVRKAIARMKTVVREREIGINR
(0.96) MNISDIRGLSDTEIKKKLEDSHKELFELRLKLSTRQLVNHRELPRVKNDIARILTVMRERELQIR
(0.95) MKTKEMRDMTNAELDQRLAELKGELFNLRFQLATGQLENPLRIRNVRKDIARAKTIIRENELKQARA
```

Figure 5: A prototype protein sequence and its neighboring sequences in the test data. The bold and highlighted characters show the simplified prototype subsequence. The matching subsequences in the neighbors are highlighted in blue.

functions and 3D structure. We investigate whether ProSeNet can learn the sequential similarity within families.

We clip the sequences with a maximum length of 512 and aim to classify the top 100 families ranked by their size. The sequences are split into 62k train and 19k test set. We set $\lambda_{l_1} = 1.0, \lambda_e = 0.1, \lambda_c = 0, \lambda_d = 0.01, d_{min} = 1.0$ and train a ProSeNet consists of a Bi-LSTM (2 layer $\times$ 50 hidden units) and 200 prototypes. We trained the ProSeNet for 40 epochs with a batch size of 64. In 10 train-test splits, the model scores an average accuracy of 97.0% ($SD = 0.2\%$) on the test set. Its accuracy is slightly lower than a Bi-LSTM (97.4%, $SD = 0.2\%$) and slightly higher than a ResNet with seven residual blocks (96.7%, $SD = 0.3\%$). However, the ProSeNet learns interpretable representations that reveal the significant subsequences for a family of proteins. An example is shown in Figure 5.

[3]http://www.yelp.com/dataset
[4]https://www.nltk.org/

4.4 Case Study 4: ECG Signal Classification

We investigate whether ProSeNet can be extended to learn meaningful prototypes in real-valued time series using the MIT-BIH Arrhythmia ECG dataset[5]. ECG is widely used in medical practices to monitor cardiac health. Correct categorization of the waveforms is critical for proper diagnosis and treatment. In the dataset, each signal consists of heartbeats annotated by at least two cardiologists. We downsample the ECG signals to 125Hz and split them into annotated heartbeats according to the protocol proposed by Kachuee *et al.*[16]. The annotations are mapped into five groups as suggested by AAMI[2]: Normal (N), Supraventricular Ectopic Beat (SVEB), Ventricular Ectopic Beat (VEB), Fusion Beat (F) and Unknown Beat (Q). The training and test set contain 87k and 21k sequences respectively.

Instead of discretizing the time series data into event sequences [3], we directly use LSTM to encode the real-valued sequence. We set $\lambda_{l_1} = 0.1$, $\lambda_e = 1.0$, $\lambda_d = 0.01$, $d_{min} = 2.0$, dropout rate to 0.1, and train a ProSeNet with a Bi-LSTM encoder (32 hidden units × 3 layers) and 30 prototypes. The training runs for 36 epochs with a batch size of 128 and prototype simplification is not applied. After removing prototypes with small weight ($\max(\boldsymbol{w}_i) < 0.1 \max(\boldsymbol{W})$), we obtain a model with 23 prototypes.

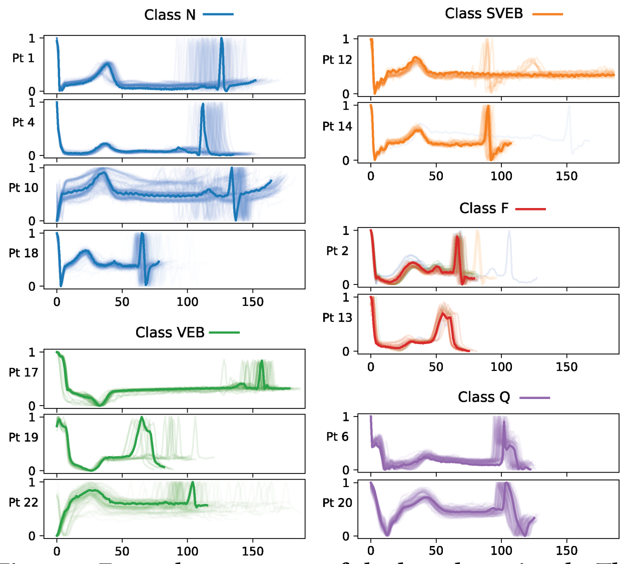


Figure 6: Example prototypes of the heartbeat signals. The prototypes are shown as bold lines and colored with their associated label. The color encodes the label of the signals. Transparent lines represent heartbeats in the test set, displayed along with their closest prototypes.

A few selected prototypes are shown in Figure 6. We can see that the ProSeNet successfully learned a few prototypes for each class. Prototype 12 shows a characteristic junctional escape beat (belonging to the SVEB group), which shows a long flat line corresponding to the dropped beat. Prototype 17 shows a premature ventricular contraction beat with a strong contraction and a long pause between ventricular contraction. This demonstrates the capability of ProSeNet in learning meaningful representations on ECG time series data, which has also been verified by two independent cardiologists.

[5]https://www.physionet.org/physiobank/database/mitdb/

We also compared our model with the state-of-the-art models for classification of ECG heartbeats. The result is summarized in Table 3. We can see that ProSeNet has comparable performance to LSTM, and even slightly better accuracy than Residual CNN [16]. Our model can present verifiable and understandable prototypes which are very useful in the healthcare domain. In practice, the most similar prototypes can be presented side-by-side with the automatic annotations of the ECG signals for explanation.

Table 3: Performance on MIT-BIH Arrythmia ECG heartbeats classification.

Model	Acc. (%)	Avg. Precision	Avg. Recall
ProSeNet$_{\text{Bi-LSTM}}$	97.7	85.0	92.6
Bi-LSTM	98.0	90.0	89.3
Residual CNN [16]	97.5	86.3	90.0

4.5 Ablation Studies

4.5.1 Choosing the Number of Prototypes k. We investigate how would the number of prototypes, k, influence the performance of ProSeNet using UniProtKB and Yelp Review data. Using the same hyperparameter configuration as in Section 4.2 and Section 4.3, we train a series of ProSeNets with different k. As shown in the blue lines in Figure 7, the accuracy first improves dramatically as k increases. Then the increasing slope quickly flattens after k exceeds 100 for UniProtKB and 40 for Yelp Reviews.

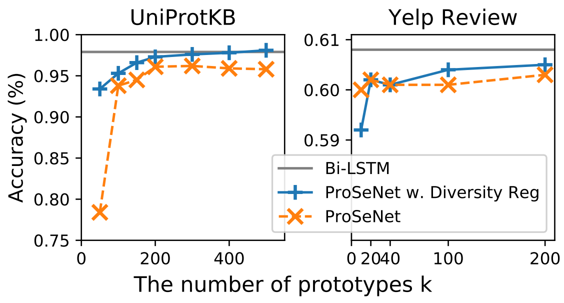


Figure 7: The influence of the prototype number and the diversity regularization R_d ($\lambda_d = 0.01$) on the performance.

An immediate question is: which k should we use? As k increases, the accuracy improves, however it will become more difficult to comprehend and differentiate such a large number of prototypes. Thus, there is a trade-off between accuracy and interpretability. In practice, since increasing k after a certain threshold only brings marginal improvement to the performance, one possible strategy is to first start from a small k (*e.g.*, $k = C$ to assume one prototype per class) and gradually increase k until the performance improvement falls below a certain threshold.

4.5.2 Effect of the Diversity Regularization Term R_d. To study the effect of the diversity regularization term, we removed the term by setting $\lambda_d = 0$ and run another set of experiments with varying prototype numbers. The result is also plotted in Figure 7. We can observe that the performance on UniProtKB is consistently lower without R_d for different settings of k. R_d also positively affects the performance on Yelp Reviews for larger ks.

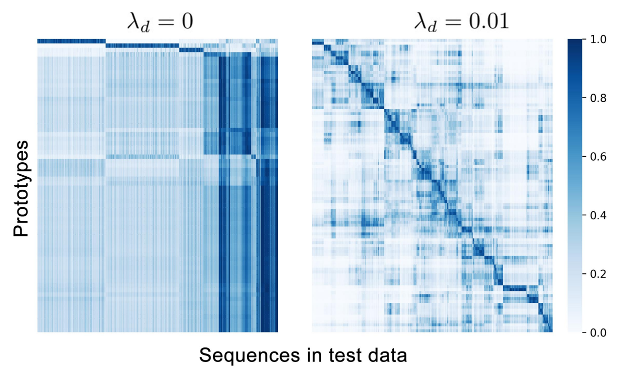


Figure 8: The influence of the diversity regularization term R_d on the diversity and sparsity of ProSeNet. The heatmaps show the similarities between prototypes and test sequences on the Yelp Reviews.

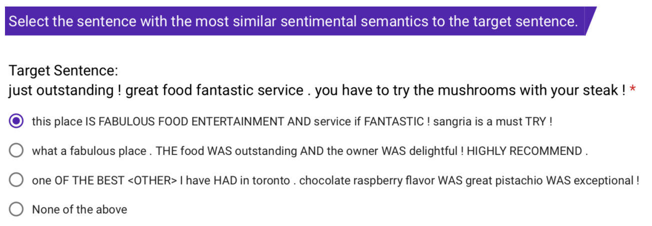
We further examine the impact of R_d by plotting the similarity scores between the prototypes and test sequences as heatmaps for two ProSeNets with 100 prototypes (Figure 8) on Yelp Review data. Without diversity regularization ($\lambda_d = 0$), most of the rows have similar horizontal patterns in the heatmap (Figure 8 left), indicating near-duplicate prototypes. With $\lambda_d = 0.01$ the similarity heatmap is much sparser and more diagonal, showing that the prototypes are more diverse and evenly distributed in the latent space.

4.5.3 Performance of Prototype Simplification. We examine the influence of prototype simplification on performance and subsequence lengths. We use the previous settings with $\lambda_d = 0.01$ on the UniProtKB and Yelp Reviews. There is no significant difference in accuracy on both datasets. However, with simplification applied, the average prototype (sub)sequence lengths are decreased from 20.1 to 15.1 on Yelp Reviews and 274.5 to 130.7 on UniProtKB dataset.

4.6 Human Evaluation of Interpretability

The interpretability of a machine learning model is a subjective concept, which is often regarded to be difficult to evaluate computationally [11, 20]. Thus, we conduct a quantitative evaluation of the interpretability of ProSeNet through experiments with human subjects. With the prototype learning structure, we aim to answer the following questions: 1) How understandable and accurate are the prototypes in explaining the predictions of the input sequences? 2) How would the incorporation of human knowledge (Section 3.4) influence the performance and interpretability of ProSeNet? We use the ProSeNet trained on Yelp Reviews for binary sentiment classification (Section 4.2) for evaluation. The model has 80 effective prototypes (*i.e.*, the associated weight $\max(\boldsymbol{w}_i) > 0.1\max(\boldsymbol{W})$).

Experiment Setup. To evaluate the interpretability of the explanations, we recruit human participants on Amazon Mechanical Turk, who are non-experts in machine learning. Directly asking whether an explanation is interpretable or accurate is very subjective and varies for different people. Thus, we adopt a relative measure by asking the participants to select one of three prototype sentences that expresses the most similar sentiment to a given input sentence. The prototype in the model that has the largest similarity score to the input sentence is regarded as the proposed answer by the model and is presented as one of the options. The other options are randomly selected from the rest of the prototypes. We also include a "None of the above" as the fourth option. The input sentences are selected from the validation set with stratified sampling. That is, we divide the sequences into groups according to their most similar prototypes, and use the groups as the strata for sampling. An example question is shown in Figure 9.


Select the sentence with the most similar sentimental semantics to the target sentence.

Target Sentence:
just outstanding ! great food fantastic service . you have to try the mushrooms with your steak ! *

[✓ this place IS FABULOUS FOOD ENTERTAINMENT AND service if FANTASTIC ! sangria is a must TRY !]
what a fabulous place . THE food WAS outstanding AND the owner WAS delightful ! HIGHLY RECOMMEND .
one OF THE BEST <OTHER> I have HAD in toronto . chocolate raspberry flavor WAS great pistachio WAS exceptional !
None of the above

Figure 9: Example question of the user experiment. The uppercase words represent simplified prototype sequences.

Table 4: Average accuracy of human subjects and ProSeNet before and after updating the model via interaction. The most voted option by human subjects is used as the correct answer. The accuracy is calculated over questions that are not most voted as "None of the above".

	Model acc.	Human acc.	None of the above
Before	0.618	0.667 (SD: 0.113)	0.131
After	**0.682**	0.698 (SD: 0.142)	0.131

We sample 70 questions and split them into four questionnaires. We gather 20 responses from different human subjects for each of the questionnaires. After filtering the responses that failed quality check (*e.g.*, consistency check of the answers of duplicate questions), each question has 12.5 valid responses on average. We use the most voted options by human subjects as the correct answer of each question and computes the accuracy of human and the model. The result is summarized in the first row in Table 4.

Interacting with sequence prototypes. To study how the input of human knowledge would affect the interpretability of the model, we use the feedback from the user study as a source of human knowledge to update the model (as described in Section 3.4) and then run a second round of experiment on MTurk. Based on the result of the first round user experiment, we update the model to improve the quality of the prototypes. The update protocol is as follows. For each of the wrongly answered question, we check the prototype sequence that is proposed as the answer by the model, as well as its neighboring sequences in the validation set. If the neighboring sequences do not have consistent sentiment (with subjective judgment), we would delete this prototype. If the neighboring sequences do have consistent sentiment, but the provided prototype is not representative enough (*e.g.*, part of the sentence has misleading meaning), a new sentence is selected from the neighboring sentences to replace the old prototype.

Following the above protocol, we updated 13 prototypes and removed 5 prototypes. After the incremental training completes, the performance of the model on the test set is basically unchanged (slightly increased by 0.1%). Then we run the second user experiments with the same procedure. An average of 12.3 valid responses is collected for each question.

Result. As shown in the second row in Table 4, the accuracy of the model's proposed answers increased significantly from 61.8% to **68.2%**, which is only 1.6% lower than human accuracy. The result of the first experiment indicates that there is still a gap between the quality of the model's generated explanations and the human standard. However, the second experiment shows that the incorporation of human knowledge via our proposed interaction scheme could be very helpful in improving the interpretability of the model.

5 CONCLUSION AND FUTURE WORK

In this paper, we presented an interpretable and steerable deep sequence modeling technique called ProSeNet. The technique combines prototype learning and RNNs to achieve both interpretability and high accuracy. Experiments and case studies on four different real-world sequence prediction/classification tasks demonstrated that ProSeNet is not only as accurate as other state-of-the-art machine learning techniques but also much more interpretable. In addition, large scale user study on Amazon Mechanical Turk demonstrated that for familiar domains like sentiment analysis on texts, ProSeNet is able to select high quality prototypes that are well-aligned with human knowledge for prediction and interpretation. Furthermore, ProSeNet obtained better interpretability without loss of performance by incorporating the feedback from the user study to update the prototypes, demonstrating the benefits of involving human-in-the-loop for interpretable machine learning. Future works include applying the technique to other sequence data and developing interactive user interface for updating the prototypes.

ACKNOWLEDGMENTS

This research was partially supported by Hong Kong TRS grant T41-709/17N.

REFERENCES

[1] Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic Text Scoring using Neural Networks. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (2016), 715–725.

[2] ANSI/AAMI. 2008. *Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms.* Standard. American National Standards Institute, Inc. (ANSI), Association for the Advancement of Medical Instrumentation (AAMI).

[3] Roel Bertens, Jilles Vreeken, and Arno Siebes. 2016. Keeping it short and simple: Summarising complex event sequences with multivariate patterns. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 735–744.

[4] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 1721–1730.

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

[6] Chaofan Chen, Oscar Li, Alina Barnett, Jonathan Su, and Cynthia Rudin. 2018. This looks like that: deep learning for interpretable image recognition. *arXiv preprint arXiv:1806.10574* (2018).

[7] Yuanzhe Chen, Panpan Xu, and Liu Ren. 2018. Sequence Synopsis: Optimize Visual Summary of Temporal Event Data. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan 2018), 45–55. https://doi.org/10.1109/TVCG.2017.2745083

[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 *Machine Learning for Healthcare Conference.* 301–318.

[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] Edward Choi, Andy Schuetz, Walter F Stewart, and Jimeng Sun. 2017. Using recurrent neural network models for early detection of heart failure onset. *Journal of the American Medical Informatics Association* 24, 2 (2017), 361–370.

[11] Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608* (2017).

[12] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Int. Conf. Acoustics, Speech and Signal Processing.* IEEE, 6645–6649.

[13] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, and Aram Galstyan. 2017. Multitask learning and benchmarking with clinical time series data. *arXiv preprint arXiv:1703.07771* (2017).

[14] 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.* 770–778.

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

[16] Mohammad Kachuee, Shayan Fazeli, and Majid Sarrafzadeh. 2018. ECG Heartbeat Classification: A Deep Transferable Representation. *arXiv preprint arXiv:1805.00794* (2018).

[17] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078* (2015).

[18] Janet L Kolodner. 1992. An Introduction to Case-based Reasoning. *Artificial Intelligence Review* 6, 1 (1992), 3–34.

[19] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. 2018. Deep Learning for Case-based Reasoning through Prototypes: A Neural Network that Explains its Predictions. In *AAAI Conference on Artificial Intelligence.*

[20] Zachary C. Lipton. 2018. The Mythos of Model Interpretability. *Commun. ACM* 61, 10 (Sept. 2018), 36–43.

[21] Zachary C Lipton, John Berkowitz, and Charles Elkan. 2015. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019* (2015).

[22] W. James Murdoch, Peter J. Liu, and Bin Yu. 2018. Beyond Word Importance: Contextual Decomposition to Extract Interactions from LSTMs. In *International Conference on Learning Representations.*

[23] W James Murdoch and Arthur Szlam. 2017. Automatic Rule Extraction from Long Short Term Memory Networks. (2017).

[24] Parliament and Council of the European Union. 2016. The General Data Protection Regulation. (2016).

[25] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the Difficulty of Training Recurrent Neural Networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28 (ICML'13).* 1310–1318.

[26] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why Should I Trust You?: Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 1135–1144.

[27] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence.*

[28] Elaine Rich and Kevin Knight. 1991. *Artificial intelligence.* Tata McGraw-Hill.

[29] Cynthia Rudin. 2018. Please Stop Explaining Black Box Models for High Stakes Decisions. *NeurIPS 2018 Workshop on Critiquing and Correcting Trends in Machine Learning* (2018).

[30] Rainer Schmidt, Stefania Montani, Riccardo Bellazzi, Luigi Portinale, and Lothar Gierl. 2001. Cased-based reasoning for medical knowledge-based systems. *International Journal of Medical Informatics* 64, 2-3 (2001), 355–367.

[31] Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M Rush. 2018. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 667–676.

[32] Jimeng Sun, Fei Wang, Jianying Hu, and Shahram Edabollahi. 2012. Supervised patient similarity measure of heterogeneous patient records. *ACM SIGKDD Explorations Newsletter* 14, 1 (2012), 16–24.

[33] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems (NIPS'14).* 3104–3112.

[34] Duyu Tang, Bing Qin, and Ting Liu. 2015. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. In *Proc. Conf. Empirical Methods in Natural Language Processing.* 1422–1432.

A PROTOTYPE SIMPLIFICATION VIA BEAM SEARCH

The beam search algorithm that we used for prototype simplification is shown in Algorithm 1. The BestCandidates($\mathcal{S}, w$) takes a set of sequences $\mathcal{S}$, computes the score using Equation 3 for each sequence, and returns w sequences with the lowest scores. The algorithm terminates when the subsequences are not reducible, or there is no better remove-one subsequences than the existing sequences in the set of candidates $\mathcal{S}$.

```
Input: encoder $r$, training data $\mathcal{X}$, prototype $\boldsymbol{p}_i$
Parameters: beam width $w$
Output: projected prototype $\hat{\boldsymbol{p}}_i$, subsequence $\boldsymbol{s}$
/* Find $w$ sequences with minimum distance to $\mathbf{p}_i$ */
$\mathcal{S} \leftarrow$ BestCandidates($\mathcal{X}, w$);
$s_{opt} \leftarrow NULL$;
while $\mathcal{S} \neq \emptyset$ do
    $\hat{\mathcal{S}} \leftarrow \emptyset$;
    for $\mathbf{s} \in \mathcal{S}$ do
        $s_{opt} \leftarrow$ BestCandidates($\{s_{opt}, \boldsymbol{s}\}, 1$);
        if $|\mathbf{s}| \geq 1$ then
            /* remove-one sub-sequences */
            $\hat{\mathcal{S}} \leftarrow \hat{\mathcal{S}} \cup \{\boldsymbol{s}/e_i \mid e_i \in \boldsymbol{s}\}$;
    end
    /* Find $w$ subsequences with the lowest scores
       using Equation 3, return $\hat{\mathcal{S}}$ if there are less
       than $w$ sequences in it. */
    $\mathcal{S} \leftarrow$ BestCandidates($\hat{\mathcal{S}}, w$);
end
$\boldsymbol{p}_i \leftarrow r(s_{opt})$;
```

Algorithm 1: Beam Search

B EXPERIMENT DETAILS

B.1 Data Processing of Yelp Reviews

For sentiment classification tasks on Yelp Reviews, we first filtered the reviews to contain only "Restuarant" reviews according to "category" field of the business being reviewed. We then tokenize the review texts into sequences of words using NLTK. For human evaluation purpose, we filtered reviews with length (number of words) over 25. For both binary classification and fine-grained classification, we balanced the classes by down-sampling. The size of the largest class is no more than twice the size of the smallest class. The vocabulary sizes are 6287 and 6792. We using word embedding of size 100 for all the models. The embedding are jointly trained with the models.

B.2 Post-processing of Human Evaluation Data

We partition the 70 questions evenly to 4 questionnaires (each with 17 and 18 questions) to prevent participants becoming overwhelming. We also add three additional quality check questions (*e.g.*, duplicate questions with options in a different order, or questions with obvious correct answer).

We filtered the responses that fails more than 1 quality-check questions. Then we compute the correct answer of each question using the most voted option. We further filter responses that have have accuracy lower than 50%. Then we finally compute the human accuracy and the model accuracy using the most voted options as the correct answers.

C SUPPLEMENT EXAMPLES

C.1 Sentiment Classification on Yelp Reviews

```
Proto: REALLY REALLY GOOD ! I had half meat and half cheese and a sub .
       would DEFINITELY VISIT AGAIN .
Weight: Positive(3.10)

Neighbors:
(0.88) really good 10+ ! ! its fresh and yummy nothing but the best ! something
       different from <OTHER> and there nice people .
(0.87) really good sandwiches ! extremely flavourful and a good selection of
       meats that you ca n't get everywhere .
(0.87) really good N second steak ! creative and tasty ! service could come with
       a smile but the food is good .
(0.86) really good comfort food with a homey feel . great customer service we
       will be going back .
```

```
Proto: I REALLY LIKE THE FOOD here . you can tell it 's prepared with a lot
       of care . LOVE EATING there .
Weight: (Positive(1.03))

Neighbors:
(0.90) i love the atmosphere and the vibes . it is kind of traditional modern .
       besides they are nice and very welcoming .
(0.89) i really liked the made to order crepes and breakfast potatoes . good
       breakfast variety and friendly staff .
(0.88) i love everything about chipotle . nothing new but their <OTHER> and
       their <OTHER> is awesome for a vegetarian .
(0.87) i enjoyed the <OTHER> burger and friendly staff . the atmosphere is not
       bad and pricing is in line with similar establishments .
```

```
Proto: SLOW SERVICE . loud music . overly GREASY noodles . BLAND sushi .
       probably wo n't EVER GO BACK .
Weight: (Negative(4.64))

Neighbors:
(0.93) service was terrible . place smells like grease . food tastes bad . rude
       hostess . manager was not helpful .
(0.92) terrible service . waited forever for a rude server . ended up leaving .
       not worth the stop .
(0.88) very disappointed . <OTHER> does n't speak english well and service is
       awful . will not come again .
(0.86) service sucks tiny portions way over priced ! will not return .
```

Figure 10: Additional examples of learned prototype sequences on Yelp Reviews. The bold uppercase texts are the simplified subsequences.

C.2 ECG Heartbeat Classification

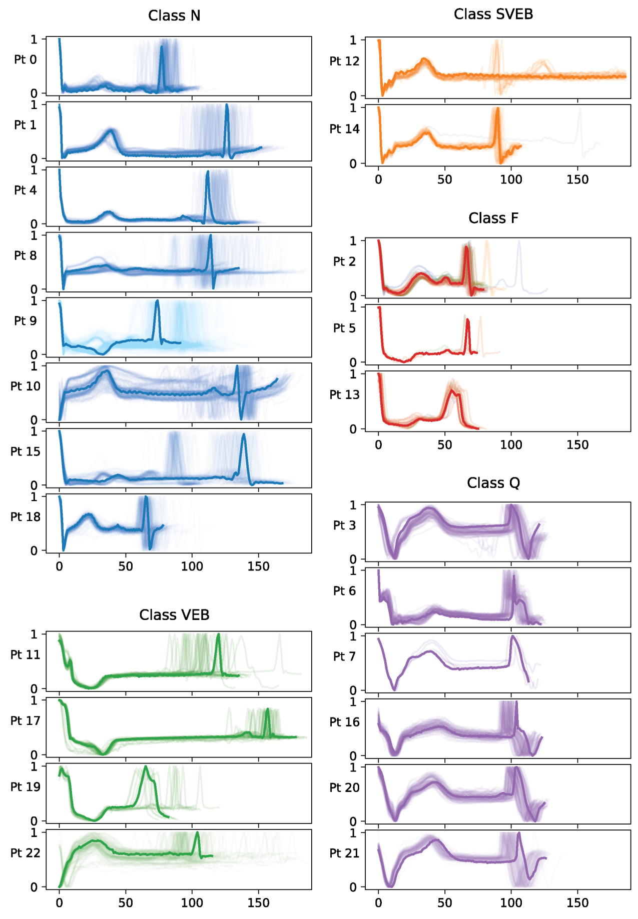


Figure 11: Complete list of time series prototypes learned form the ECG heartbeats. Bold lines represent the prototype signals, and transparent lines shows are test signals close to the prototypes.