STONE: Signal Temporal Logic Neural Network for Time Series Classification

Ruixuan Yan,* Agung Julius,* Maria Chang,† Achille Fokoue,† Tengfei Ma,† Rosario Uceda-Sosa†

* *Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA*
{*yanr5, juliua2*}@*rpi.edu*

† *IBM T.J. Watson Research Center, IBM Research, Yorktown Heights, NY, USA*
{*maria.chang, tengfei.ma1*}@*ibm.com*,{*achille, rosariou*}@*us.ibm.com*

2021 International Conference on Data Mining Workshops (ICDMW) | 978-1-6654-2427-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICDMW53433.2021.00101

Abstract*—In this paper, we propose a neuro-symbolic framework called signal temporal logic neural network (STONE) that combines the characteristics of neural networks and temporal logics. Weighted Signal Temporal Logic (wSTL) formulas are recursively composed of subformulas connected using logical and temporal operators. The quantitative semantics of wSTL is defined such that the quantitative satisfaction of subformulas with higher weights have a more significant influence on the quantitative satisfaction of a wSTL formula. In the STONE, each neuron represents a component of a wSTL formula, and the output of STONE corresponds to the quantitative satisfaction of a wSTL formula. We use STONE to represent wSTL formulas and classify time-series data. WSTL formulas are more interpretable and human-readable than classical time series classification models. The STONE is end-to-end differentiable, which allows learning of wSTL formulas to be done using backpropagation. Experiments on benchmark time-series datasets show that STONE is comparable to the state-of-the-art time series classification models and the wSTL learning algorithm is faster than the traditional STL learning algorithm.

I. Introduction

Time series classification (TSC) has been considered as one of the most challenging tasks in machine learning (ML). Many supervised ML algorithms have been proposed to solve TSC problems [2]–[8]. However, these classification models are often not human-readable, for example, hyperplanes in a higher-dimensional space [3]. Temporal logics are formal languages that can express specifications about the temporal properties of systems. Compared with traditional ML models, temporal logic formulas can express temporal and logical properties in a human-readable and interpretable form. Human readability and interpretability are important because they can give non-expert users insights into the model. With these benefits, temporal logics have been exploited to model time-series data [9]–[12] and explore temporal properties in time-series data.

Signal Temporal Logic (STL), a branch of temporal logic, can express temporal specifications on cyber-physical systems [13]–[16]. STL has been a popular tool for analyzing time-series data by learning STL formulas from data. The learning task is based on a notion of quantitative satisfaction of STL formulas and is posed as an optimization problem with the quantitative satisfaction in the objective function [11], [12]. For example, an STL formula that can classify the "Coffee" dataset [17] in Fig. 1 is expressed as $\phi = \Diamond_{[154,165]} s \leq -0.8017$ (details will be discussed in Section III), which reads as "The signal s is smaller than or equal to -0.8017 at some time point between 154 and 165." The quantitative satisfaction of STL, as proposed in [18], uses $\max$ and $\min$ functions, which are non-smooth functions. As pointed out in [19], the traditional quantitative satisfaction has a limitation that the satisfaction of parts of the formula other than the most "significant" part has no contribution to the overall quantitative satisfaction. The authors in [19] then proposed a weighted STL (wSTL) that allows the expression of user preference on STL specifications by encoding the preference of satisfaction with weights. However, the quantitative satisfaction proposed in [19] has two limitations: 1) subformulas with zero weights are still influential in the overall quantitative satisfaction; 2) it does not reflect that quantitative satisfaction of subformulas with larger weights has a greater influence on the overall quantitative satisfaction.

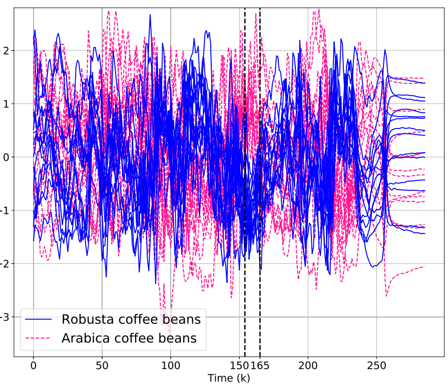


Fig. 1. Time-series data of the "Coffee" dataset with annotated time interval learned by an STL formula $\phi = \Diamond_{[154,165]} s \leq -0.8017$.

Combining neural networks and symbolic logic to perform learning tasks has attracted lots of attention [1], [20]–[22] in recent years, which produces modeling representations that

This work was supported by the National Science Foundation under Grant CNS-1618369 and Grant CNS-1936578. The authors would like to thank Dr. Alexander Gray from IBM Research for the introduction of Logical Neural Network (LNN) [1].

978-1-6654-2427-1/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDMW53433.2021.00101

are interpretable. Activation functions corresponding to truth functions of logical operators in first-order logic have been proposed in [1] such that the truth value bounds of formulas can be learned from the neural network. Most of the existing STL learning algorithms solve a non-convex optimization problem to find the parameters in the formula [12], [23], [24], where the loss function is not differentiable everywhere with respect to the parameters which results in the learning process being slow. Also, the learned STL formula cannot reflect the importance of data at different time points in deciding the property of interest. For instance, the STL formula learned from the "Coffee" dataset, $\phi = \Diamond_{[154,165]} s \leq -0.8017$, cannot manifest which time points between 154 and 165 play a more important role in classifying the two classes of data.

The problem of using neural networks to perform signal temporal logic learning tasks has not been well studied. The contributions of this paper are: 1) We define a novel quantitative satisfaction for wSTL that has the properties of non-influence of zero weights, ordering of influence, and monotonicity (see Section IV for more details). 2) We construct a novel neuro-symbolic framework called weighted Signal Temporal lOgic Neural nEtwork (STONE) that combines the characteristics of neural networks and wSTL to perform TSC tasks and express the models as interpretable and human-readable formulas. Each neuron in STONE represents a predicate, a temporal or a logical operator, with weights on the edges connecting the neurons. 3) We perform extensive experiments on multiple benchmark time-series datasets to compare the proposed STONE with the state-of-the-art TSC models. We show that STONE is highly competitive to the other models, and we compare STONE with the traditional STL learning algorithm and show STONE has the benefit of high computational efficiency.

II. Related Work

Many existing time series classifiers can be generally divided into distance-based, interval-based, dictionary-based, frequency-based, Shapelet-based, and ensemble-based models. Distance-based models, such as the K-nearest neighbors (KNN) algorithm have been used to perform TSC tasks by replacing Euclidean distance metric with dynamic time warping (DTW) metric [5], which compares the similarity of two time-evolving sequences of data. Interval-based approaches, such as time series forest (TSF) [6], classify time-series data using a random forest classifier, which splits the data into random intervals and extracts statistical characteristics to complete the classification task. TSF has been demonstrated to perform better than KNN with DTW [6]. Supervised Time Series Forest (STSF) is another interval-based approach that classifies time-series data by examining groups of relevant intervals using a tree-based structure [25]. Dictionary-based models, such as Bag of Symbolic Fourier Approximation Symbols (BOSS) [26], extract words (sub-series) from time-series data and create features based on their frequency. The collected features can then be used with any classifier. Frequency-based models, such as random interval spectral ensemble (RISE) [7], extract frequency-domain characteristics. RISE is a variant of TSF in which the features taken from the original time-series data are spectral rather than statistical. Shapelet-based methods are designed to explore shapelets, which are subsequences of time-series data used to find similarities between series within the same class [27]. Shapelet Transform (ST) [28] is one shapelet-based model that classifies time-series data based on the similarity between the data and the extracted shapelets. Ensemble-based models are ensembles of multiple TSC models. Hierarchical vote collective of transformation-based ensembles (HIVE-COTE) is one popular ensemble model for TSC [7], whose prediction is a weighted average of predictions from its base models. Nonetheless, these models cannot express the temporal and logical properties of time-series data as a natural-language form formula that is human-readable and interpretable.

With the advancement of deep neural networks, end-to-end neural network architectures also emerged for TSC. Fully convolutional neural networks (FCN), residual network (ResNet), and Multi-scale Convolutional Neural Network (MCNN) have been developed to classify time-series data [29], [30]. These models are shown to achieve competitive results as the models discussed earlier. However, these models also have the limitations of interpretability.

STL has been applied to infer knowledge from time-series data [12], [24] due to its expressivity of temporal properties in a logical statement. STL inference refers to learning an STL formula from a set of labeled data by performing a classification task. The STL inference approach has the limitations of a long training period and a formula that cannot describe the importance of data at various time points in determining quantitative satisfaction. The wSTL proposed in this paper can tackle the above issues, and the STONE can learn wSTL formulas efficiently due to its end-to-end differentiability.

III. Preliminaries

A discrete l-dimensional time-series data is denoted as $s = s(0), s(1), ..., s(K)$, where $l \in \mathbb{Z}_{>0}$, $s(k) \in \mathbb{R}^l$, $k, K \in \mathbb{Z}_{\geq 0}$ and $k \leq K$. A time interval between k_1 and k_2 is denoted as $I = [k_1, k_2] = \{k' | k_1 \leq k' \leq k_2, k_1, k_2 \in \mathbb{Z}_{\geq 0}\}$, and $k + I$ denotes the time interval $[k + k_1, k + k_2]$.

A. Signal Temporal Logic (STL)

In this paper, we consider a fragment of Signal Temporal Logic (STL) proposed in [31], whose syntax is defined recursively as follows:

$$\phi := \top | \pi | \neg \phi | \phi_1 \wedge \phi_2 | \phi_1 \vee \phi_2 | \Box_I \phi | \Diamond_I \phi, \tag{1}$$

where ϕ, ϕ_1, ϕ_2 are STL formulas, $\top$ is Boolean *True*, $\pi := f(s)$ is a predicate defined over s, and $f(s) = \boldsymbol{a}^T s \leq c$, $\boldsymbol{a} \in \mathbb{R}^l$, $\|\boldsymbol{a}\|_2 = 1, c \in \mathbb{R}$, $\neg, \wedge, \vee$ are logical *negation, conjunction,* and *disjunction* operators. Temporal operators $\Box, \Diamond$ read as "*always*" and "*eventually*", respectively. I is a time interval, and $\Box_I \phi$ is satisfied at k if ϕ is satisfied at all $k' \in k + I$, $\Diamond_I \phi$ is satisfied at k if ϕ is satisfied at least one $k' \in k + I$. The Boolean semantics of STL measures whether s

satisfies ϕ at k qualitatively, for example, $(s,k) \models \phi$ reads as "s satisfies ϕ at k" and $(s,k) \not\models \phi$ reads as "s violates ϕ at k". The quantitative semantics of STL measures the degree of satisfaction or violation of ϕ over s at k.

Definition 1 *The quantitative semantics (satisfaction) of STL is defined as [18]:*

$$\begin{aligned} r(s,\pi,k) &= c - \boldsymbol{a}^T s(k), \\ r(s,\neg\phi,k) &= -r(s,\phi,k), \\ r(s,\phi_1 \wedge \phi_2, k) &= \min(r(s,\phi_1,k), r(s,\phi_2,k)), \\ r(s,\phi_1 \vee \phi_2, k) &= \max(r(s,\phi_1,k), r(s,\phi_2,k)), \\ r(s,\Box_I \phi, k) &= \min_{k' \in k+I} r(s,\phi,k'), \\ r(s,\Diamond_I \phi, k) &= \max_{k' \in k+I} r(s,\phi,k'). \end{aligned} \tag{2}$$

The quantitative satisfaction at $k=0$ is simplified as $r(s,\phi)$.

B. Weighted Signal Temporal Logic (wSTL)

The quantitative satisfaction of the $\wedge, \Box, \vee, \Diamond$ operators in **Definition 1** is the minimum or maximum of the quantitative satisfaction of subformulas. This means the overall quantitative satisfaction is determined by the quantitative satisfaction of a single subformula. Furthermore, the traditional STL quantitative satisfaction cannot express importance over different subformulas. In many circumstances, we need a quantitative satisfaction that can account for the effects of different subformulas.

Example 1 *Consider the "Coffee" dataset presented in the Introduction section, the STL formula learned by classifying this dataset is expressed as $\phi = \Diamond_{[154,165]} s \leq -0.8017$, which means there is at least one time point between 154 and 165 such that s is smaller than or equal to -0.8017. Specifically, the time-series data between $k=154$ and $k=165$ of the "Coffee" dataset is shown in Fig. 2. The STL formula ϕ has the drawback of not being informative enough to reflect which time points in the interval $[154,165]$ play a more important role in classifying these two classes of data. This is due to the fact that the quantitative satisfaction of ϕ is determined by the data at a single time point. We seek a more informative formula that can reflect the significance of data at various time points in classifying the dataset.*

We introduce the notion of importance weights into STL, and the novel STL is called weighted STL (wSTL) [19].

Definition 2 *The syntax of wSTL is defined over s as [19]*

$$\tilde{\phi} := \top | \pi | \neg \tilde{\phi} | {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2 | {}^{w_1}\tilde{\phi}_1 \vee {}^{w_2}\tilde{\phi}_2 | \Box_I^{\boldsymbol{w}} \tilde{\phi} | \Diamond_I^{\boldsymbol{w}} \tilde{\phi}, \tag{3}$$

where $\top, \pi$, and the logical and temporal operators are the same as the ones in STL, w_1 and w_2 are non-negative weights on the subformulas $\tilde{\phi}_1$ and $\tilde{\phi}_2$, respectively, and $\boldsymbol{w} = [w_{k_1}, w_{k_1+1}, ..., w_{k_2}]^T \in \mathbb{R}_{\geq 0}^{k_2-k_1+1}$ assigns a non-negative weight $w_{k'}$ to $k' \in [k_1, k_2]$ in the temporal operators.

With importance weights, wSTL can express more informative specifications. Throughout the paper, we use $\boldsymbol{w}$ to denote weights associated with an operator and $\boldsymbol{w}^{\tilde{\phi}}$ to denote weights associated with a wSTL formula $\tilde{\phi}$ that may have multiple operators.

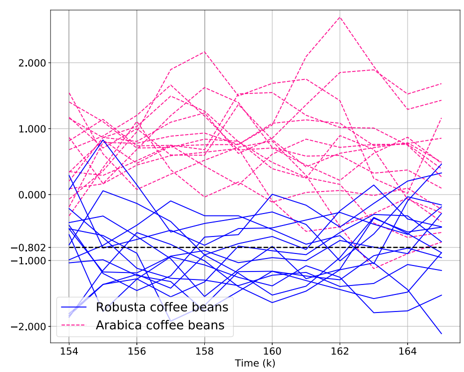


Fig. 2. Time-series data between 154 and 165 of "Coffee" dataset.

Definition 3 *The quantitative semantics (satisfaction) of wSTL is defined as [19]*

$$\begin{aligned} r^w(s,\pi,k) &= c - \boldsymbol{a}^T s(k), \\ r^w(s,\neg\tilde{\phi},k) &= -r^w(s,\tilde{\phi},k), \\ r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2, k) &= \otimes^{\wedge}([w_i, r^w(s,\tilde{\phi}_i,k)]_{i=1,2}), \\ r^w(s, {}^{w_1}\tilde{\phi}_1 \vee {}^{w_2}\tilde{\phi}_2, k) &= \otimes^{\vee}([w_i, r^w(s,\tilde{\phi}_i,k)]_{i=1,2}), \\ r^w(s, \Box_I^{\boldsymbol{w}}\tilde{\phi}, k) &= \otimes^{\Box}(\boldsymbol{w}, [r^w(s,\tilde{\phi},k+k')]_{k' \in [k_1,k_2]}), \\ r^w(s, \Diamond_I^{\boldsymbol{w}}\tilde{\phi}, k) &= \otimes^{\Diamond}(\boldsymbol{w}, [r^w(s,\tilde{\phi},k+k')]_{k' \in [k_1,k_2]}), \end{aligned} \tag{4}$$

where $\otimes^{\wedge}, \otimes^{\vee} : \mathbb{R}_{\geq 0}^2 \times \mathbb{R}^2 \to \mathbb{R}$, $\otimes^{\Box}, \otimes^{\Diamond} : \mathbb{R}_{\geq 0}^{k_2-k_1+1} \times \mathbb{R}^{k_2-k_1+1} \to \mathbb{R}$ are activation functions corresponding to the $\wedge, \vee, \Box, \Diamond$ operators, respectively. The concrete-form activation functions for the $\wedge$, $\vee$, $\Box$, and $\Diamond$ operators will be presented in Section IV.

Remark 1 *The double negation property holds for the activation functions in (4) because $r^w(s,\neg\neg\tilde{\phi},k) = -r^w(s,\neg\tilde{\phi},k) = r^w(s,\tilde{\phi},k)$.*

Remark 2 *The quantitative satisfaction in (4) satisfies Demorgan's law if and only if the activation functions for the $\wedge, \vee, \Box, \Diamond$ operators satisfy Demorgan's law. For example, we can show that Demorgan's law holds for the $\wedge$ and $\vee$ operators, and the validity of Demorgan's law for the $\Box$ and $\Diamond$ operators can be proved similarly. It follows that*

$$r(s, \neg({}^{w_1}\neg\tilde{\phi}_1 \wedge {}^{w_2}\neg\tilde{\phi}_2), k) = -\otimes^{\wedge}([w_i, -r^w(s,\tilde{\phi}_i,k)]_{i=1,2}). \tag{5}$$

As the aggregation function $\otimes^{\wedge}$ satisfies Demorgan's law, we have

$$\begin{aligned} &-\otimes^{\wedge}([w_i, -r^w(s,\tilde{\phi}_i,k)]_{i=1,2}) = \otimes^{\vee}([w_i, r^w(s,\tilde{\phi}_i,k)]_{i=1,2}) \\ &= r^w(s, {}^{w_1}\tilde{\phi}_1 \vee {}^{w_1}\tilde{\phi}_2, k). \end{aligned} \tag{6}$$

C. Problem Statement

In this paper, we consider solving the following problems: 1) Design activation functions for the $\wedge, \vee, \Box, \Diamond$ operators in (4) such that subformulas with zero weights are not influential in the overall quantitative satisfaction of a wSTL formula, and subformulas with larger weights have a greater influence on the overall quantitative satisfaction. 2) Given two sets of time-series data D_P and D_N, where $D_P = \{D_P^i\}_{i=1}^{\tilde{p}}$ is a set of positive data with label 1 and $D_N = \{D_N^i\}_{i=1}^{\tilde{n}}$ is a set of negative data with label -1, we aim to build a neuro-symbolic model according to the design of the activation functions for the $\wedge, \vee, \Box, \Diamond$ operators. The neuro-symbolic model embodies the characteristics of neural networks and wSTL, and can classify D_P and D_N and produce a wSTL formula $\tilde{\phi}$ that is human-readable and interpretable. In the meantime, $\tilde{\phi}$ is satisfied by the data in D_P and violated by the data in D_N.

IV. Our Approach

In this section, we define a new quantitative satisfaction for wSTL such that subformulas with zero weights are not influential in the overall quantitative satisfaction and subformulas with larger weights have a greater influence on the overall quantitative satisfaction. In this manner, we can build a neural network model in which a neuron has not only a corresponding activation function but also a logical meaning. The proposed neuro-symbolic model is called a weighted Signal Temporal lOgic Neural nEtwork (STONE). Structurally, a STONE is a graph composed of neurons representing predicates and operators connected in the way determined by a wSTL formula. Neurons representing predicates accept s at single time points as inputs and have activation functions corresponding to $r^w(s, \pi, k)$, whose outputs are quantitative satisfaction of predicates. Neurons representing logical or temporal operators accept quantitative satisfaction of subformulas as inputs and have activation functions corresponding to the operators.

A. Activation Functions for Logical and Temporal Operators

Gradient-based neural learning of importance weights $\boldsymbol{w}^{\tilde{\phi}}$ requires the activation functions to be smooth and differentiable with respect to $\boldsymbol{w}^{\tilde{\phi}}$, where $\boldsymbol{w}^{\tilde{\phi}}$ denotes all the weights in $\tilde{\phi}$. Activation functions for the $\vee, \Box, \Diamond$ operators are derived from the activation function for the $\wedge$ operator using DeMorgan's law. A weighted version of the quantitative satisfaction in (2) was proposed in [19], where the activation function for the $\wedge$ operator is expressed as

$$\otimes^{\wedge}([w_i, r_i]_{i=1:N}) = \min_{i=1:N} \left\{ \left((\frac{1}{2} - \bar{w}_i) sign(r_i) + \frac{1}{2} \right) r_i \right\}, \tag{7}$$

where $r_i = r^w(s, \tilde{\phi}_i, k)$, and $\bar{w}_i = w_i / \sum_{j=1}^{N} w_j$ is the normalized weight, and $sign$ denotes the sign function. The activation function in (7) has two limitations: 1) If $w_i = 0$, then r_i is still possible to be the one deciding the quantitative satisfaction, for example, if $N = 3$, $r_1 = 1, r_2 = r_3 = 3$, and $\bar{w}_1 = 0, \bar{w}_2 = 0.5, \bar{w}_3 = 0.5$, then the overall quantitative satisfaction is $\otimes^{\wedge}([w_i, r_i]_{i=1:3}) = r_1$. This means even if the importance weight associated with $\tilde{\phi}_1$ is 0, it can still determine the overall quantitative satisfaction. 2) The quantitative satisfaction of $\tilde{\phi}$ is still determined by the data at a single time point. To address these limitations, we propose several principles of the activation functions such that subformulas with higher weights have a greater influence on the overall quantitative satisfaction and the weights can be learned by a neuro-symbolic framework. As the activation functions for the $\vee$, $\Box$, and $\Diamond$ operators are derived from the activation functions for the $\wedge$ operator using DeMorgan's law, the principles designed for the $\wedge$ operator also hold for the activation functions for the $\vee$, $\Box$, and $\Diamond$ operators. For simplicity, we only discuss the design principles for the $\wedge$ operator as follows.

The choice of the activation functions for the $\wedge$ operator should obey the following principles:

- **Non-influence of zero weights:** $r^w(s, \tilde{\phi}_i, k)$ has no influence on $r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2, k)$ if $w_i = 0$, $i = 1, 2$, where w_1 and w_2 are the weights defined in (4).
- **Ordering of influence:** if $r^w(s, \tilde{\phi}_1, k) = r^w(s, \tilde{\phi}_2, k)$ and $w_1 > w_2$, then we have
$$\frac{\partial r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2, k)}{\partial r^w(s, \tilde{\phi}_1, k)} > \frac{\partial r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2, k)}{\partial r^w(s, \tilde{\phi}_2, k)},$$
where w_1 and w_2 are the weights defined in (4).
- **Monotonicity:** $r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2, k)$ increases monotonically over $r^w(s, \tilde{\phi}_i, k)$, $i = 1, 2$, i.e.
$$\otimes^{\wedge}([w_i, r^w(s, \tilde{\phi}_i, k)]_{i=1,2}) \leq \otimes^{\wedge}([w_i, r^w(s, \tilde{\phi}_i, k) + d]_{i=1,2}),$$
where $d \geq 0$.

This paper develops a concrete activation function for the $\wedge$ operator by introducing another variable σ, which is used to define a *softmin* function that replaces the $\min$ function in the traditional STL quantitative semantics. The activation functions for the other operators are derived using DeMorgan's law.

Definition 4 *The activation functions for the $\wedge, \vee, \Box, \Diamond$ operators in (4) are defined as*

$$\begin{aligned}
&\otimes^{\wedge}([w_i, r^w(s, \tilde{\phi}_i, k)]_{i=1,2}, \sigma) = \frac{\sum_{i=1}^{2} \bar{w}_i s_i r_i}{\sum_{i=1}^{2} \bar{w}_i s_i}, \\
&\otimes^{\vee}([w_i, r^w(s, \tilde{\phi}_i, k)]_{i=1,2}, \sigma) = -\frac{\sum_{i=1}^{2} \bar{w}_i s_i r_i}{\sum_{i=1}^{2} \bar{w}_i s_i}, \\
&\otimes^{\Box}(\boldsymbol{w}, [r^w(s, \tilde{\phi}, k+i)]_{i \in I}, \sigma) = \frac{\sum_{i=k_1}^{k_2} \bar{w}_i s_i r_i}{\sum_{i=k_1}^{k_2} \bar{w}_i s_i}, \\
&\otimes^{\Diamond}(\boldsymbol{w}, [r^w(s, \tilde{\phi}, k+i)]_{i \in I}, \sigma) = -\frac{\sum_{i=k_1}^{k_2} \bar{w}_i s_i r_i}{\sum_{i=k_1}^{k_2} \bar{w}_i s_i},
\end{aligned} \tag{8}$$

where $\bar{w}_i$ is the normalized weight ($\bar{w}_i = w_i / \sum_{j=1}^{2} w_j$ in the $\wedge, \vee$ operators, $\bar{w}_i = w_i / \sum_{j=k_1}^{k_2} w_j$ in the $\Box, \Diamond$ operators), $I = [k_1, k_2]$, $r_i = r^w(s, \tilde{\phi}_i, k)$ in the $\wedge$ operator,

$r_i = -r^w(s, \tilde{\phi}_i, k)$ in the $\vee$ operator. In the activation function for the $\Box$ operator, r_i is defined as

$$r_i = \begin{cases} r^w(s, \tilde{\phi}, k+i) & \text{if } k+i \leq K, \\ \infty & \text{otherwise.} \end{cases}$$

In the activation function for the $\Diamond$ operator, r_i is defined as

$$r_i = \begin{cases} -r^w(s, \tilde{\phi}, k+i) & \text{if } k+i \leq K, \\ \infty & \text{otherwise.} \end{cases}$$

Also,

$$s_i = e^{-\frac{r_i}{\sigma}} / \sum_{j=1}^{2} e^{-\frac{r_j}{\sigma}}$$

in the $\wedge$ and $\vee$ operators, and

$$s_i = e^{-\frac{r_i}{\sigma}} / \sum_{j=k_1}^{k_2} e^{-\frac{r_j}{\sigma}}$$

in the $\Box, \Diamond$ operators, $\sigma > 0$. Note that the activation function for the $\vee$ operator is derived from the activation function for the $\wedge$ operator using DeMorgan's law. To clarify the notation, we use $\boldsymbol{w}$ to denote weights of an operator before normalization and $\bar{\boldsymbol{w}}$ to denote weights of the same operator after normalization. The activation functions for the $\Box$ and $\Diamond$ operators are designed such that if $k+i > K$, i.e., the length of data is smaller than the time interval of a wSTL formula, then the quantitative satisfaction beyond K does not affect the overall quantitative satisfaction by setting $r_i = \infty$.

Proposition 1 *The activation functions defined in (8) satisfy the principles of non-influence of zero weights, ordering of influence, and monotonicity.*

Proof *The proof of Proposition 1 is described as follows.*

- ***Non-influence of zero weights***
  Without loss of generality, let $w_1 = 0$, and $w_2 = 1$, then we have

$$r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2, k) = \frac{s_2 r_2}{s_2} = r_2,$$

  which shows r_1 is not influential in $r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2, k)$.
- ***Ordering of influence***
  Taking the derivative of $r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2, k)$ with respect to $r^w(s, \tilde{\phi}_1, k)$, we have

$$\frac{\partial r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2, k)}{\partial r^w(s, \tilde{\phi}_1, k)} = \frac{(\bar{w}_1 s_1 + \bar{w}_1 r_1 \frac{\partial s_1}{\partial r_1})}{(\bar{w}_1 s_1 + \bar{w}_2 s_2)^2} (\bar{w}_1 s_1 + \bar{w}_2 s_2) - \frac{(\bar{w}_1 s_1 r_1 + \bar{w}_2 s_2 r_2)\bar{w}_1 \frac{\partial s_1}{\partial r_1}}{(\bar{w}_1 s_1 + \bar{w}_2 s_2)^2}. \tag{9}$$

  Taking the derivative of $r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2, k)$ with respect to $r^w(s, \tilde{\phi}_2, k)$, we have

$$\frac{\partial r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2, k)}{\partial r^w(s, \tilde{\phi}_2, k)} = \frac{(\bar{w}_2 s_2 + \bar{w}_2 r_2 \frac{\partial s_2}{\partial r_2})}{(\bar{w}_1 s_1 + \bar{w}_2 s_2)^2} (\bar{w}_1 s_1 + \bar{w}_2 s_2) - \frac{(\bar{w}_1 s_1 r_1 + \bar{w}_2 s_2 r_2)\bar{w}_2 \frac{\partial s_2}{\partial r_2}}{(\bar{w}_1 s_1 + \bar{w}_2 s_2)^2}. \tag{10}$$

  As the denominator is positive and is the same in (9) and (10), we only need to compare the numerator of (9) and (10). If $r^w(s, \tilde{\phi}_1, k) = r^w(s, \tilde{\phi}_2, k)$, $w_1 > w_2$, i.e. $r_1 = r_2$ and $\bar{w}_1 > \bar{w}_2$, the difference between the numerator is

$$\begin{aligned}
&(\bar{w}_1 s_1 + \bar{w}_1 r_1 \frac{\partial s_1}{\partial r_1})(\bar{w}_1 s_1 + \bar{w}_2 s_2) - (\bar{w}_1 s_1 r_1 + \bar{w}_2 s_2 r_2) \\
&\bar{w}_1 \frac{\partial s_1}{\partial r_1} - (\bar{w}_2 s_2 + \bar{w}_2 r_2 \frac{\partial s_2}{\partial r_2})(\bar{w}_1 s_1 + \bar{w}_2 s_2) + (\bar{w}_1 s_1 r_1 \\
&+ \bar{w}_2 s_2 r_2)\bar{w}_2 \frac{\partial s_2}{\partial r_2} \\
&= (s_1(\bar{w}_1 - \bar{w}_2) + r_1 \frac{\partial s_1}{r_1}(\bar{w}_1 - \bar{w}_2))(\bar{w}_1 s_1 + \bar{w}_2 s_1) \\
&+ (\bar{w}_1 s_1 r_1 + \bar{w}_2 s_1 r_1)\frac{\partial s_1}{\partial r_1}(\bar{w}_2 - \bar{w}_1) \\
&= s_1(\bar{w}_1 - \bar{w}_2)(\bar{w}_1 s_1 + \bar{w}_2 s_1) + \bar{w}_1 s_1 r_1 \frac{\partial s_1}{\partial r_1}(\bar{w}_1 - \bar{w}_2) \\
&+ \bar{w}_2 s_1 r_1 \frac{\partial s_1}{\partial r_1}(\bar{w}_1 - \bar{w}_2) + \bar{w}_1 s_1 r_1 \frac{\partial s_1}{\partial r_1}(\bar{w}_2 - \bar{w}_1) \\
&+ \bar{w}_2 s_1 r_1 \frac{\partial s_1}{\partial r_1}(\bar{w}_2 - \bar{w}_1) \\
&= s_1^2(\bar{w}_1^2 - \bar{w}_2^2) > 0,
\end{aligned} \tag{11}$$

  which proves that if $r^w(s, \tilde{\phi}_1, k) = r^w(s, \tilde{\phi}_2, k)$ and $w_1 > w_2$, the following holds:

$$\frac{\partial r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2, k)}{\partial r^w(s, \tilde{\phi}_1, k)} > \frac{\partial r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2, k)}{\partial r^w(s, \tilde{\phi}_2, k)}.$$

- ***Monotonicity***
  For $\otimes^\wedge([w_i, r^w(s, \tilde{\phi}_i, k)]_{i=1,2})$, we have

$$\otimes^\wedge([w_i, r^w(s, \tilde{\phi}_i, k)]_{i=1,2}) = \frac{\bar{w}_1 s_1 r_1 + \bar{w}_2 s_2 r_2}{\bar{w}_1 s_1 + \bar{w}_2 s_2},$$

  and for $\otimes^\wedge([w_i, r^w(s, \tilde{\phi}_i, k) + d]_{i=1,2})$, we have

$$\otimes^\wedge([w_i, r^w(s, \tilde{\phi}_i, k) + d]_{i=1,2}) = \frac{\bar{w}_1 s_1' r_1' + \bar{w}_2 s_2' r_2'}{\bar{w}_1 s_1' + \bar{w}_2 s_2'},$$

  where $r_1' = r_1 + d, r_2' = r_2 + d$. We can show that

$$\begin{aligned}
s_i' &= \frac{e^{-(r_i+d)}}{\sum_{j=1}^2 e^{-(r_j+d)}} \\
&= \frac{e^{-d} e^{-r_i}}{e^{-d} \sum_{j=1}^2 e^{-r_j}}, \\
&= \frac{e^{-r_i}}{\sum_{j=1}^2 e^{-r_j}} = s_i, i = 1, 2.
\end{aligned}$$

  As $r_1 + d \geq r_1$, and $r_2 + d \geq r_2$, we have

$$\begin{aligned}
&\otimes^\wedge([w_i, r^w(s, \tilde{\phi}_i, k) + d]_{i=1,2}) \\
&= \frac{\bar{w}_1 s_1 (r_1 + d) + \bar{w}_2 s_2 (r_2 + d)}{\bar{w}_1 s_1 + \bar{w}_2 s_2} \\
&\geq \frac{\bar{w}_1 s_1 r_1 + \bar{w}_2 s_2 r_2}{\bar{w}_1 s_1 + \bar{w}_2 s_2} \\
&= \otimes^\wedge([w_i, r^w(s, \tilde{\phi}_i, k)]_{i=1,2}),
\end{aligned}$$

  which proves the monotonicity holds.

The connection between the quantitative satisfactions of wSTL and STL is discussed in the following proposition.

Proposition 2 *For sufficiently small σ and $w_1 = w_2$, $r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge^{w_2} \tilde{\phi}_2, k)$ can be arbitrarily close to the traditional STL quantitative satisfaction, i.e.*

$$\lim_{\sigma \to 0} r^w(s, {}^{w_1}\tilde{\phi}_1 \wedge^{w_2} \tilde{\phi}_2, k) = \min\{r^w(s, \tilde{\phi}_1, k), r^w(s, \tilde{\phi}_2, k)\}.$$

Proof *We know that $r_1 = r(s, \tilde{\phi}_1, k), r_2 = r(s, \tilde{\phi}_2, k)$, and*

$$s_i = \frac{e^{-\frac{r_i}{\sigma}}}{\sum_{j=1}^{2} e^{-\frac{r_j}{\sigma}}}. \tag{12}$$

Without loss of generality, let $r_1 < r_2$, then for sufficiently small σ, s_1 will approximate 1, and s_2 will approximate 0. Hence the robustness becomes

$$\lim_{\sigma \to 0} r^w(s, {}^{w_1}\phi_1 \wedge {}^{w_2}\phi_2, k) = \frac{\bar{w}_1(s_1 \uparrow^1) r_1 + \bar{w}_2(s_2 \uparrow^0) r_2}{\bar{w}_1(s_1 \uparrow^1) + \bar{w}_2(s_2 \uparrow^0)},$$
$$= r_1 = \min\{r(s, \tilde{\phi}_1, k), r(s, \tilde{\phi}_2, k)\}.$$

With the activation functions defined in (8), we can design neural networks for wSTL formulas by encoding the temporal and logical operators as neurons. A particular STONE for $\tilde{\phi} = \Diamond^{\boldsymbol{w}^1}_{[1,2]} \Box^{\boldsymbol{w}^2}_{[0,3]} \pi$ is shown in Fig 3.

B. Learning of wSTL Formulas with STONE

In this paper, wSTL learning for a given set of M formula structure candidates refers to optimizing the parameters for each formula candidate and selecting the wSTL formula that can best classify the two sets of time-series data D_P and D_N. The learned wSTL formula is satisfied by the data in D_P and violated by the data in D_N. Suppose $D_C = D_P \cup D_N$ that has $\mathcal{T} = \tilde{p} + \tilde{n}$ data. The overall learning process is shown in Fig. 4.

The loss function should satisfy the requirements that the loss is small when the learned formula is satisfied by the positive data or violated by the negative data, and the loss is large when these two conditions are not satisfied. A candidate loss function that satisfies the above requirements is [24]

$$J(\tilde{\phi}) = \sum_{j=1}^{\mathcal{T}} h(l_j, r^w(D_C^j, \tilde{\phi})), \tag{13}$$

where D_C^j is the j-th data in D_C, l_j is the label of D_C^j, and

$$h(l_j, r^w(D_C^j, \tilde{\phi})) = \begin{cases} \zeta l_j r^w(D_C^j, \tilde{\phi}) & \text{if } l_j r^w(D_C^j, \tilde{\phi}) > 0, \\ \gamma & \text{else,} \end{cases} \tag{14}$$

where $\zeta > 0$ is a tuning parameter, and γ is a positive large number that penalizes the cases when $\tilde{\phi}$ is violated by positive data or satisfied by negative data. However, this loss function is not differentiable everywhere with respect to $\boldsymbol{w}^{\tilde{\phi}}$ and other parameters in the STONE. An alternative loss function that is differentiable is

$$J(\tilde{\phi}) = \sum_{j=1}^{\mathcal{T}} \exp(-\zeta l_j r^w(D_C^j, \tilde{\phi})). \tag{15}$$

Algorithm 1 wSTL Learning Algorithm

Input: Time-series data D_C, a wSTL formula with specified structure $\tilde{\phi}$, number of iterations K, number of subformulas $\mathcal{J}$

Output: Learned wSTL formula $\tilde{\phi}$

```
Construct a STONE based on the structure of φ̃, and initialize w^φ̃, a, c.
for k = 1, 2, ..., K do
    Select a mini-batch data D_C^k from D_C.
    for j = 1, 2, ..., J do
        Perform forward-propagation to compute the quantitative satisfaction of the j-th subformula using (8).
    end for
    Compute the loss at the current iteration using (15).
    Perform back-propagation to update the parameters in the STONE.
end for
return φ̃
```

$J(\tilde{\phi})$ is small when $\tilde{\phi}$ is satisfied by positive data or violated by negative data and grows exponentially when $\tilde{\phi}$ is violated by positive data or satisfied by negative data. Another issue of (14) is that for a correctly predicted positive (negative) example, the loss increases as the quantitative satisfaction increases (decreases), but for very large (small) quantitative satisfaction the loss can potentially be greater than the penalty term γ. The loss function in (15) addresses this issue by taking the negation. We use Pytorch to build the STONE and perform the back-propagation to learn the formula. The wSTL learning algorithm with STONE is illustrated in **Algorithm 1**.

C. Formula Structure Selection

In this paper, we consider six wSTL formula structures that correspond to six common properties, which are also commonly used in the learning of STL formulas. The formula structures are described as follows.

(1) Multiple conjunctive patterns:
$$\tilde{\phi} = {}^{w_0}\tilde{\phi}_0 \wedge {}^{w_1}\tilde{\phi}_1 \wedge {}^{w_2}\tilde{\phi}_2 ... \wedge {}^{w_K}\tilde{\phi}_K,$$
where $\tilde{\phi}_0, \tilde{\phi}_1, ..., \tilde{\phi}_K$ are subformulas that can be predicates or wSTL formulas.

(2) Multiple disjunctive patterns:
$$\tilde{\phi} = {}^{w_0}\tilde{\phi}_0 \vee {}^{w_1}\tilde{\phi}_1 \vee {}^{w_2}\tilde{\phi}_2 ... \vee {}^{w_K}\tilde{\phi}_K.$$

(3) Consistent pattern:
$$\tilde{\phi} = \Box^{\boldsymbol{w}}_{[0,K]}\tilde{\phi}_0.$$

(4) Alternative pattern:
$$\tilde{\phi} = \Diamond^{\boldsymbol{w}}_{[0,K]}\tilde{\phi}_0.$$

(5) Consistently alternative pattern:
$$\tilde{\phi} = \Box^{\boldsymbol{w}^1}_{[0,K]}\Diamond^{\boldsymbol{w}^2}_{[0,K]}\tilde{\phi}_0.$$

(6) Alternatively consistent pattern:
$$\tilde{\phi} = \Diamond^{\boldsymbol{w}^1}_{[0,K]}\Box^{\boldsymbol{w}^2}_{[0,K]}\tilde{\phi}_0.$$

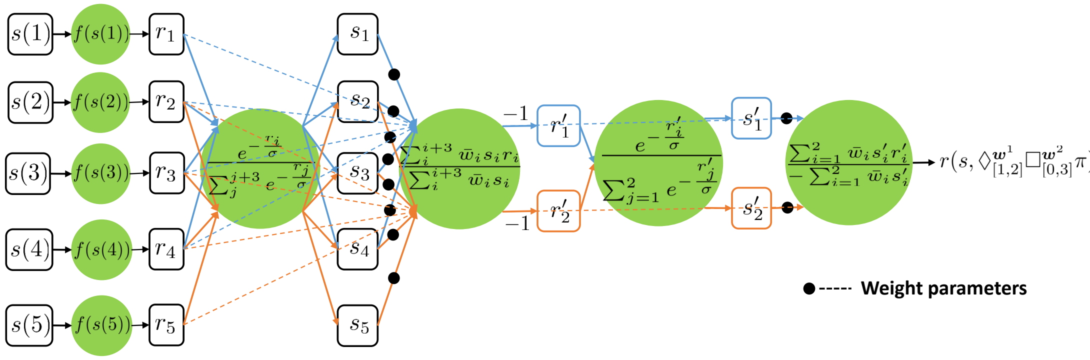


Fig. 3. The neural network structure for $\tilde{\phi} = \Diamond_{[1,2]}^{\boldsymbol{w}^1} \Box_{[0,3]}^{\boldsymbol{w}^2} \pi$.

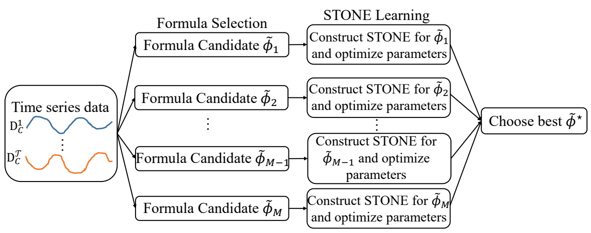


Fig. 4. Overview of the learning process of STONE.

V. Experiments

In this section, we use two experiments to evaluate the performance of the wSTL learning algorithm with STONE. In the first experiment, we compare the classification accuracy of STONE and other popular TSC models on 16 benchmark datasets. In the second experiment, we compare the training time of STONE and a traditional STL inference algorithm [24]. For each experiment, we describe the experimental setup and show the results. The experiments in the paper are implemented on a Windows laptop with a 1.9GHz Intel i7-8665U CPU and an 8 GB RAM.

Through the experiments, we aim to answer the following questions: 1) Can STONE achieve a competitive classification accuracy compared to the state-of-the-art models, and is there any improvement in the accuracy? 2) Can the learned wSTL formula provide insights into the temporal and logical properties that determine the problem of interest? 3) Can STONE be more computationally efficient compared with the STL learning algorithm?

A. Classification Accuracy

Experiment Setup: The datasets used in this experiment are selected from the UCR Time Series Classification Archive [17], which contains time-series data from different domains. We select 16 binary classification datasets and present the information about each dataset in Table I. The training and test data are split according to the default split specified in the UCR archive. For each dataset, we compare our model with 1NN-DTW [5], TSF [6], HIVE-COTE [7], FCN [29], ResNet [29], LSTM [32], and GRU [33] models. The formula structures of STONE are given as the six commonly used structures described in Section IV-C. The parameters in STONE are set as $\sigma = 1$ and $\zeta = 1$. We run the STONE for 100 epochs, and in each epoch we train the model using all the training data and evaluate the model using the test data. Since the dataset archive was published, many machine learning (ML) models have been applied to perform the classification task [3]. These ML models are less interpretable and human-readable than the STONE classifier. The weights in these ML models represent how much each feature in the model contributes to the prediction, however, the model cannot be written as a human-readable formula. The STONE classifier can not only reflect the contribution of the features but also express the model as a wSTL formula that is human-readable and interpretable. The learned wSTL formula can tell us what kind of temporal properties determine the problem of interest.

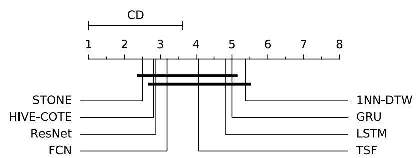
Results: The classification accuracy for STONE and other competitive models on the test set is reported in Table I. The accuracy of STONE is chosen as the best accuracy of the six formula structures proposed in Section IV-C, and the number (i) next to the accuracy means the i-th formula structure in Section IV-C has the highest accuracy. The classification accuracy for the other models is the same as their original papers. The best result for each dataset is highlighted in **bold** format. From Table I, we could observe STONE achieves a better average classification accuracy on the 16 datasets. Also, we compute the average rank of each classifier and use a critical difference diagram [34] shown in Fig. V-A to visualize this comparison, where a bold horizontal line denotes a collection of classifiers that are statistically similar. On average, STONE outperforms the other TSC models across the 16 datasets. The above results demonstrate that STONE is highly competitive to the state-of-the-art models.

B. Interpretability of STONE

The main advantage of STONE is its interpretability and human-readability. For example, the wSTL formula learned

Name	#Train	#Test	Length	1NN-DTW	TSF	HIVE-COTE	FCN	ResNet	LSTM	GRU	**STONE**
Coffee	28	28	286	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	71.42	71.42	**100.00** / (4)
DistalPhalanxOutlineCorrect	600	276	80	72.46	77.17	77.17	76.00	77.10	78.00	72.82	**80.16** / (2)
Earthquakes	322	139	512	72.66	74.82	74.82	72.70	71.20	**79.85**	78.41	**79.85** / (5)
ECG200	100	100	96	88.00	85.50	85.00	88.90	87.40	82	81.00	**90.00** / (1)
ECGFiveDays	23	861	136	79.67	93.69	**100.00**	98.70	97.50	94.54	95.81	95.65 / (4)
GunPoint	50	150	150	91.33	95.07	**100.00**	**100.00**	99.10	77.33	82.67	94.00 / (4)
Ham	109	105	431	60.00	74.29	66.67	71.80	75.70	66.67	63.81	**82.85** / (1)
HandOutlines	1000	370	2709	87.84	91.89	**93.24**	80.60	91.10	70.81	69.72	90.00 / (2)
Herring	64	64	512	53.12	60.94	68.75	60.80	61.90	**70.31**	67.18	**70.31** / (1)
ItalyPowerDemand	67	1029	24	95.53	97.00	96.30	96.10	96.30	96.30	96.89	**97.08** / (5)
SonyAIBORobotSurface1	20	601	70	69.55	75.64	76.54	95.80	**96.00**	84.52	83.02	92.51 / (3)
SonyAIBORobotSurface2	27	953	65	85.94	81.86	92.76	**97.90**	97.80	75.65	81.63	84.57 / (3)
Strawberry	613	370	235	94.59	96.49	97.03	97.20	**98.10**	89.72	89.45	97.79 / (2)
TwoLeadECG	23	1139	82	86.83	90.39	99.65	**100.00**	**100.00**	66.63	67.60	**100.00** / (2)
Wafer	1000	6164	152	99.59	99.50	**99.94**	99.90	99.70	98.23	98.92	96.70 / (1)
Wine	57	54	234	61.11	62.96	77.78	58.70	74.40	64.81	62.96	**81.48** / (1)
Average accuracy				81.13	84.82	87.85	87.19	88.95	79.17	78.95	**89.63**

TABLE I
CLASSIFICATION ACCURACY (%) OF STONE AND OTHER TSC MODELS ON 16 BENCHMARK DATASETS.



captionCritical difference diagram showing average ranks of STONE and other TSC models on 16 benchmark datasets.

by classifying the "Coffee" dataset is $\tilde{\phi} = \Diamond^{\boldsymbol{w}}_{[0,285]} 1.8975 s \leq -0.6531$, where $\boldsymbol{w}$ is the 286-dimensional weight variable. The three most important normalized weights are

$$w_{157} = 0.3227, w_{158} = 0.2691, w_{159} = 0.2070, \tag{16}$$

which means the data at time points $k = 157, 158, 159$ are more important in classifying the two classes of data. Correspondingly, the data belonging to time interval $[150, 164]$ is shown in Fig 5. We could observe that the difference between the two classes of data is more significant during $k \in [157, 159]$ than at other time points. This temporal information is more informative than the STL formula learned from the STL learning algorithm, $\phi = \Diamond_{[154,165]} s \leq -0.8017$ because $\tilde{\phi}$ identifies the data at time points $157, 158, 159$ are more important than the other time points in the interval $[154, 165]$ in classifying the data. Another example is the wSTL formula learned by classifying the "ItalyPowerDemand" dataset, which is expressed as

$$\tilde{\phi} = \Box^{\boldsymbol{w}^1}_{[0,23]} \Diamond^{\boldsymbol{w}^2}_{[0,23]} 1.1493 s \leq 0.3569. \tag{17}$$

The three most important normalized weights in $\boldsymbol{w}^1$ are

$$w^1_{17} = 0.5170, w^1_{6} = 0.2168, w^1_{16} = 0.2082, \tag{18}$$

and the five most important normalized weights in $\boldsymbol{w}^2$ are

$$\begin{aligned} &w^2_1 = 0.1068, w^2_2 = 0.1594, w^2_{12} = 0.1862, \\ &w^2_{13} = 0.1880, w^2_{14} = 0.1618, \end{aligned} \tag{19}$$

which implies the data at time points $k = 7, 8, 17, 18, 19, 20$ are more important in deciding the season of power usage. In particular, we could observe an obvious difference between the two classes happening during 17:00 and 20:00 in Fig. 6, which corresponds to the fact that in summer, the power demands start to decrease when evening comes as the weather gets cool, while in winter, the power demands start to increase when evening comes as the weather gets cold. The above experimental results demonstrate that STONE could yield a formula that is close to natural language and can reveal the temporal and logical characteristics useful for the classification.

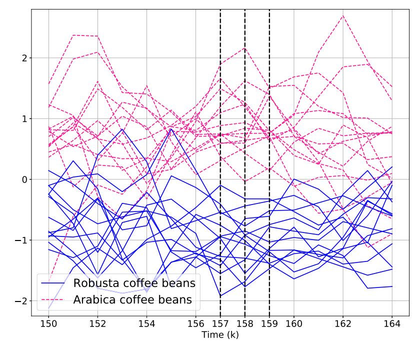


Fig. 5. Time-series data between 150 and 164 of "Coffee" dataset with annotated time points $k = 157, 158, 159$.

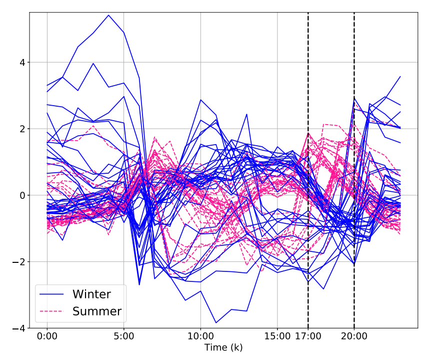


Fig. 6. Time-series data of "ItalyPowerDemand" dataset with annotated time interval $17:00 - 20:00$.

C. Computational Efficiency

Traditional STL learning algorithms in [12], [24] can also express the model as a human-readable STL formula. However, as mentioned earlier, the learned STL formula cannot reflect the importance of data at different time points in determining the characteristics of interest. Instead, wSTL formulas can express the importance of data at different time points by introducing weights. In the meantime, wSTL learning algorithm can also improve the computational efficiency, i.e. reduce training time. We compare the training time for the STL learning algorithm and the wSTL learning algorithm on each of the 16 datasets. The formula structure used in the STL learning algorithm is the same as the best formula structure in STONE. For each dataset, the training time of STONE is less than the training time of the STL learning algorithm. The average training time for wSTL learning algorithm is 6.65 seconds, which is 45 times faster than the average training time for STL learning algorithm (300.92 seconds). Therefore, STONE is more computationally efficient than the STL learning algorithm.

D. STONE with Weight Sparsification

As the structures of wSTL formulas become sophisticated, the number of parameters in the STONE will grow. As a result, the memory required for back-propagation will be intensive. Sparsifying the weights of the STONE is one way to reduce the memory footprint of the network. In the meantime, from previous experimental results, we could observe that data at certain time points play a more important role in determining a property. Weight sparsification could refine the formula by keeping the subformulas that are more important. A particular weight sparsification approach is by thresholding the number ($\bar{s}$) of weights to keep, which is called top-$\bar{s}$ sparsification, i.e.

$$\tilde{w}_i = \begin{cases} \bar{w}_i & \text{if } \bar{w}_i \text{ is one of the top-}\bar{s}\text{ weights,} \\ 0 & \text{otherwise,} \end{cases} \tag{20}$$

where $\tilde{w}_i$ is the i-th weight after sparsification.

The weight sparsification experiments are performed on eight datasets in Fig. 7. We aim to explore the effect of sparsification on classification accuracy. The sparsification level $\bar{s}$ is chosen as $\bar{s} \in [1, 5]$, and the classification accuracy for different sparsification levels is shown in Fig. 7. We could observe generally with the decreasing of $\bar{s}$, the classification accuracy also decreases as the important subformulas are neglected. In practice, we need to consider the trade-off between the memory saved and the classification accuracy to determine the sparsification level such that the requirements for memory and classification accuracy are both satisfied.

E. Interpretability Comparison

Many interpretable TSC models such as Shapelet Transform (ST) [28], Time Series Forest (TSF) [6], and Symbolic Aggregate Approximation and Vector Space Model (SAX-VSM) [35] have been developed to extract features from subsequences of time-series data and identify the discriminatory subsequences to represent a class. However, all these models cannot provide a human-readable formula. In contrast, STONE can both identify the discriminatory intervals and express the discriminatory information as a logical formula that is human-readable. We use the "GunPoint" dataset to compare the interpretability of STONE and SAX-VSM. For the "Gun" class, an actor moves his hand above a hip-mounted holster, and moves his hand down to grasp the gun, and moves the gun up to point it at a target and returns the gun to the holster, and returns his hand to the side. For the "Point" class, the actor directly moves his finger up and points with his finger to the target, and then returns his hand to the side. For both classes, the x position (x-pos) of the centroid of the actor's hand is recorded. The discriminatory patterns identified by STONE and SAX-VSM are shown in Fig. 8, in which patterns from STONE are consistent with the patterns from SAX-VSM. The extracted patterns correspond to the difference between moving hand down to grasp the gun from the holster and directly moving hand up to the shoulder level. More importantly, if we truncate the small weights, the discriminatory patterns learned by STONE can be described by a human-readable wSTL formula as $\tilde{\phi} = (x(28) \leq 0.2456) \vee (x(29) \leq 0.2093) \vee (x(30) \leq 0.2103) \vee (x(31) \leq 0.2207) \vee (x(32) \leq 0.2311) \vee (x(33) \leq 0.2429) \vee (x(34) \leq 0.2495) \vee (x(35) \leq 0.2612) \vee (x(36) \leq 0.2831) \vee (x(37) \leq 0.3695)$, which reads as "x-pos at 28 s is smaller than 0.2456 or x-pos at 29 s is smaller than 0.2093 or x-pos at 30 s is smaller than 0.2103 or x-pos at 31 s is smaller than 0.2207 or x-pos at 32 s is smaller than 0.2311 or x-pos at 33 s is smaller than 0.2429 or x-pos at 34 s is smaller than 0.2495 or x-pos at 35 s is smaller than 0.2612 or x-pos at 36 s is smaller than 0.2831 or x-pos at 37 s is smaller than 0.3695". The patterns at 28 s and 29 s describe moving the hand down to grasp the gun and the patterns from 29 s to 37 s correspond to moving the hand up to the shoulder level. In summary, previous interpretable TSC models cannot provide a human-readable formula to describe the discriminatory patterns, while STONE can simultaneously identify the discriminatory patterns and provide a human-readable formula that describes the discriminatory patterns.

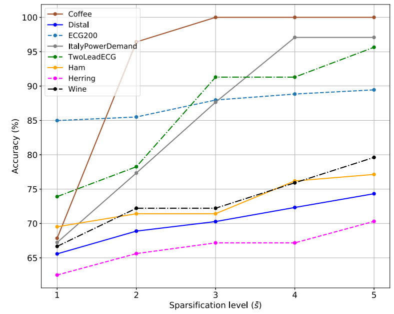


Fig. 7. Classification accuracy of STONE with different sparsification levels on eight benchmark datasets.

VI. Conclusion

We propose new semantics for wSTL, which extends STL by incorporating weights into the specifications and whose quantitative satisfaction enjoys three key properties: non-influence of zero weights, ordering of influence, and monotonicity. A novel framework called STONE combining the

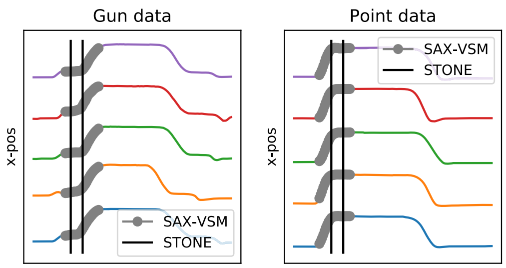


Fig. 8. Discriminatory patterns extracted by STONE and SAX-VSM on the "GunPoint" dataset.

characteristics of neural networks and wSTL is presented, where a neuron has not only a corresponding activation function but also a logical meaning. Due to the differentiable property of the quantitative satisfaction of wSTL, the parameters of wSTL can be learned from STONE in an end-to-end fashion, and the task of TSC can be accomplished. STONE is shown to be competitive against state-of-the-art TSC models with a series of experiments on benchmark time-series datasets. Moreover, the outcome of STONE can be expressed as a logical formula that is interpretable and human-readable. Comparisons with a traditional STL learning approach demonstrate that STONE is on average 45 times faster than the traditional STL learning approach on the benchmark datasets.

References

[1] R. Riegel, A. Gray, F. Luus, N. Khan, N. Makondo, I. Y. Akhalwaya, H. Qian, R. Fagin, F. Barahona, U. Sharma *et al.*, "Logical neural networks," *arXiv preprint arXiv:2006.13155*, 2020.

[2] B. Qian, Y. Xiao, Z. Zheng, M. Zhou, W. Zhuang, S. Li, and Q. Ma, "Dynamic multi-scale convolutional neural network for time series classification," *IEEE Access*, vol. 8, pp. 109 732–109 746, 2020.

[3] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[4] Y. He, J. Pei, X. Chu, Y. Wang, Z. Jin, and G. Peng, "Characteristic subspace learning for time series classification," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 1019–1024.

[5] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and information systems*, vol. 7, no. 3, pp. 358–386, 2005.

[6] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Information Sciences*, vol. 239, pp. 142–153, 2013.

[7] J. Lines, S. Taylor, and A. Bagnall, "Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles," *ACM Transactions on Knowledge Discovery from Data*, vol. 12, no. 5, 2018.

[8] Q. Ma, W. Zhuang, and G. Cottrell, "Triple-shapelet networks for time series classification," in *2019 IEEE International Conference on Data Mining (ICDM)*, 2019, pp. 1246–1251.

[9] C. Yoo and C. Belta, "Rich time series classification using temporal logic," in *Robotics: Science and Systems*, 2017.

[10] G. Bombara, C.-I. Vasile, F. Penedo, H. Yasuoka, and C. Belta, "A decision tree approach to data classification using signal temporal logic," in *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, 2016, pp. 1–10.

[11] Z. Xu and U. Topcu, "Transfer of temporal logic formulas in reinforcement learning," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 4010–4018.

[12] Z. Kong, A. Jones, A. Medina Ayala, E. Aydin Gol, and C. Belta, "Temporal logic inference for classification and prediction from data," in *Proceedings of the 17th international conference on Hybrid systems: computation and control*, 2014, pp. 273–282.

[13] N. Mehdipour, C.-I. Vasile, and C. Belta, "Arithmetic-geometric mean robustness for control from signal temporal logic specifications," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1690–1695.

[14] R. Yan and A. Julius, "A decentralized B&B algorithm for motion planning of robot swarms with temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7389–7396, 2021.

[15] M. Charitidou and D. V. Dimarogonas, "Signal temporal logic task decomposition via convex optimization," *IEEE Control Systems Letters*, 2021.

[16] D. Gundana and H. Kress-Gazit, "Event-based signal temporal logic synthesis for single and multi-robot tasks," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3687–3694, 2021.

[17] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML, "The ucr time series classification archive," October 2018.

[18] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2010, pp. 92–106.

[19] N. Mehdipour, C.-I. Vasile, and C. Belta, "Specifying user preferences using weighted signal temporal logic," *IEEE Control Systems Letters*, vol. 5, no. 6, pp. 2006–2011, 2021.

[20] F. Yang, Z. Yang, and W. W. Cohen, "Differentiable learning of logical rules for knowledge base reasoning," in *Advances in Neural Information Processing Systems*, 2017, pp. 2319–2328.

[21] L. Serafini and A. d. Garcez, "Logic tensor networks: Deep learning and logical reasoning from data and knowledge," *arXiv preprint arXiv:1606.04422*, 2016.

[22] T. Rocktäschel and S. Riedel, "End-to-end differentiable proving," in *Advances in Neural Information Processing Systems*, 2017, pp. 3788–3800.

[23] Z. Xu, M. Ornik, A. A. Julius, and U. Topcu, "Information-guided temporal logic inference with prior knowledge," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1891–1897.

[24] R. Yan, Z. Xu, and A. Julius, "Swarm signal temporal logic inference for swarm behavior analysis," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3021–3028, 2019.

[25] N. Cabello, E. Naghizade, J. Qi, and L. Kulik, "Fast and accurate time series classification through supervised interval search," in *2020 IEEE International Conference on Data Mining (ICDM)*, 2020, pp. 948–953.

[26] P. Schäfer, "The boss is concerned with time series classification in the presence of noise," *Data Mining and Knowledge Discovery*, vol. 29, no. 6, pp. 1505–1530, 2015.

[27] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data mining and knowledge discovery*, vol. 31, no. 3, pp. 606–660, 2017.

[28] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data mining and knowledge discovery*, vol. 28, no. 4, pp. 851–881, 2014.

[29] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 1578–1585.

[30] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," *arXiv preprint arXiv:1603.06995*, 2016.

[31] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[32] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[33] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[34] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[35] P. Senin and S. Malinchik, "Sax-vsm: Interpretable time series classification using sax and vector space model," in *2013 IEEE 13th international conference on data mining*. IEEE, 2013, pp. 1175–1180.