Neuro-Symbolic Generation of Explanations for Robot Policies with Weighted Signal Temporal Logic

Mikihisa Yuasa[1], Ramavarapu S. Sreenivas[1], Huy T. Tran[1]

***Abstract*—Learning-based policies have demonstrated success in many robotic applications, but often lack explainability. We propose a neuro-symbolic explanation framework that generates a weighted signal temporal logic (wSTL) specification which describes a robot policy in a human-interpretable form. Existing methods typically produce explanations that are verbose and inconsistent, which hinders explainability, and are loose, which limits meaningful insights. We address these issues by introducing a simplification process consisting of predicate filtering, regularization, and iterative pruning. We also introduce three explainability metrics—conciseness, consistency, and strictness—to assess explanation quality beyond conventional classification accuracy. Our method—TLNET—is validated in three simulated robotic environments, where it outperforms baselines in generating concise, consistent, and strict wSTL explanations without sacrificing accuracy. This work bridges policy learning and explainability through formal methods, contributing to more transparent decision-making in robotics.**

***Index Terms*—Formal Methods in Robotics and Automation, Reinforcement Learning, Deep Learning Methods, Human-Centered Robotics**

I. INTRODUCTION

RECENT innovations in learning-based methods, particularly those using neural networks, have improved high-level decision-making and low-level control in robotics. However, neural network policies are inherently black-box in nature, which poses significant concerns for safety-critical robotic applications, such as autonomous vehicles, where explainability is paramount. Various approaches have been explored to enhance the explainability of learned policies [1], including post-hoc techniques [2] and methods to learn policies that are inherently explainable [3].

In this paper, we focus on post-hoc analysis to enable explanation of any given robot policy. While existing approaches, like saliency maps [4], feature attribution [5], and non-temporal concept/rule extraction [6], provide valuable information, they often lack language-based explanations, which limits human interpretability. Autoregressive language models have shown promising results for providing language-based explanations [7], but they do not support formal verification of the output explanation, which limits their applicability in safety-critical domains [8].

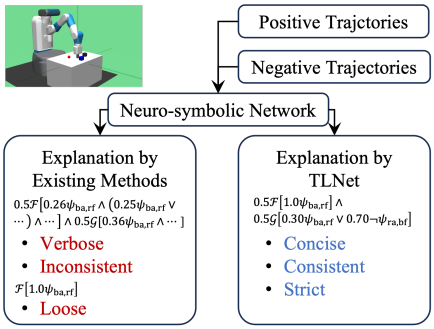


Fig. 1. Existing neuro-symbolic methods often produce explanations that are verbose, inconsistent, and loose. Our method, TLNET, addresses this by generating concise, consistent, and strict wSTL specifications.

Temporal logic offers an alternative approach to specifying robotic tasks and system constraints in an interpretable yet mathematically precise manner that supports verification [9]–[11]. Recent efforts have also used temporal logic to infer explanations of given trajectory data. For example, [12], [13] use a decision tree (DT) classifier to infer the linear temporal logic (LTL) or signal temporal logic (STL) specification that best explains the difference between two classes of trajectories. [14] uses a greedy search over candidate LTL explanations driven by policy similarity metrics. However, these methods are limited in their ability to infer specifications where certain predicates are more critical than others. *Weighted STL* (wSTL) [15] extends STL by introducing importance weights over predicates, which can highlight the relative priority of various task components. The expressiveness of wSTL thus makes it particularly suitable for describing robotic behaviors in settings with competing or hierarchical objectives.

The presence of real-valued weights in wSTL has enabled the use of neural networks for inferring wSTL specifications that explain given trajectory data. These methods have shown high classification accuracy for various applications, including anomaly detection [16], fault diagnosis [17], and weather forecasting [18]. However, existing work only focuses on classification accuracy, which does not fully capture the actual explainability of learned specifications. For example, as shown in Figure 1, these methods often produce specifications that are unnecessarily verbose (i.e., they include many unnecessary

Manuscript received: May, 30, 2025; Revised October, 14, 2025; Accepted January, 28, 2026. This paper was recommended for publication by Editor Editor A. Bera upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by the Office of Naval Research (ONR) under Grant N00014-20-12249 and JASSO Study Abroad Fellowship Program (Graduate Degree Program).

[1]The Grainger College of Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801 USA. {myuasa2,rsree,huytran1}@illinois.edu

Digital Object Identifier (DOI): see top of this page.

predicates), inconsistent across seeds, and/or loose, such that they are satisfied but do not provide useful insights (e.g., "catch-all" specifications that include many conjunctions or overly simplified specifications). Some works address verbosity by incorporating heuristic simplifications, like only keeping the top-k predicates [16], [19] or using a greedy process to remove weights [20]. However, no existing methods effectively balance classification accuracy with verbosity, inconsistency, and looseness.

In this work, we propose TLNET, a neural network architecture for inferring wSTL explanations of robot policies that are concise (i.e., include as few predicates as needed), consistent (i.e., produce similar explanations across training seeds), and strict (i.e., contain as few disjunctions as needed). TLNET achieves this through a multi-step simplification process consisting of predicate filtering, a regularized loss function, and iterative weight pruning. We provide theoretical justification for our network's weight distributivity and introduce explainability metrics—conciseness, consistency, and strictness—to complement standard classification accuracy. We demonstrate our method in three simulated robotics environments, showing it outperforms baselines in our proposed metrics while matching or improving classification accuracy.

II. PRELIMINARIES

Let a trajectory (or a signal) of n-dimensional states be a discrete-time series $\tau = s_0, s_1, \ldots, s_H$, where $s_t \in S$ is a state at timestep $t \in \mathbb{Z}_{\geq 0}$, $S \subseteq \mathbb{R}^n$ is the state space, and H is the time horizon of τ. We also define an interval I as $I := [a, b] = \{k \in \mathbb{Z}_{\geq 0} | a \leq k \leq b; a, b \in \mathbb{Z}_{\geq 0}\}$, $|I|$ as the cardinality of I, and $t + I$ as the interval $[t + a, t + b]$.

A. Weighted Signal Temporal Logic

We consider wSTL syntax [15] defined recursively as $\phi := \top \mid \psi \mid \neg\phi \mid \bigwedge_{i=1}^{N} {}^{w}\phi_i \mid \bigvee_{i=1}^{N} {}^{w}\phi_i \mid \mathcal{G}_I^w \phi \mid \mathcal{F}_I^w \phi$, where $\top$ is the logical *True* value, $\psi := f(s) \geq c$ is an atomic predicate given a function $f : S \to \mathbb{R}$ with constant $c \in \mathbb{R}$, and $\neg$, $\wedge$, and $\vee$ are Boolean *negation*, *conjunction*, and *disjunction* operators, respectively. $\mathcal{G}_I$ and $\mathcal{F}_I$ are *globally* and *eventually* operators, respectively. For conjunctive and disjunctive operators, we define the weights as $w := [w_i]_{i=1:N} \in \mathbb{R}^N_{>0}$, where N is the number of subformulae in the operator. For temporal operators, we define the weights as $w := [w_k]_{k \in I} \in \mathbb{R}^{|I|}_{>0}$ for interval I. For notational simplicity, we use the following notation interchangeably: $\bigwedge_{i=1}^{N} {}^{w}\phi_i = \bigwedge_{i=1}^{N} w_i \phi_i$ and $\bigvee_{i=1}^{N} {}^{w}\phi_i = \bigvee_{i=1}^{N} w_i \phi_i$.

B. Normalized wSTL

Our neural network architecture proposed in Section III-A requires the weights within each operator to be normalized. We discuss the reasons for this requirement in Section III-E. Given this requirement, we define normalized wSTL syntax recursively as $\phi := \top \mid \psi \mid \neg\phi \mid \bigwedge_{i=1}^{N} {}^{w}\phi_i \mid \bigvee_{i=1}^{N} {}^{w}\phi_i \mid \mathcal{G}_I^w \phi \mid \mathcal{F}_I^w \phi$, where all terminology carries over from wSTL, except we further require $\sum_{i=1}^{N} w_i = \sum_{k \in I} w_k = 1$, where $w_i, w_k \in [0, 1]$. We allow the weights to be zero because the zero weights do not influence the overall robustness [16].

To avoid large input values to our neural network, we also define normalized quantitative satisfaction semantics, or *robustness*, by extending [15] as follows,

$$r(\psi, \tau, t) := \begin{cases} \frac{f(s_t) - c}{\sup_{s \in S} f(s) - c} & \text{if } f(s_t) - c \geq 0, \\ \frac{f(s_t) - c}{\inf_{s \in S} f(s) - c} & \text{otherwise} \end{cases}, \tag{1}$$

$$r(\neg\phi, \tau, t) := -r(\phi, \tau, t), \tag{2}$$

$$r\left(\bigwedge_{i=1}^{N} {}^{w}\phi_i, \tau, t\right) := \otimes^{\wedge}\left(w, [r(\phi_i, \tau, t)]_{i=1:N}\right), \tag{3}$$

$$r\left(\bigvee_{i=1}^{N} {}^{w}\phi_i, \tau, t\right) := \oplus^{\vee}\left(w, [r(\phi_i, \tau, t)]_{i=1:N}\right), \tag{4}$$

$$r\left(\mathcal{G}_I^w \phi, \tau, t\right) := \otimes_I^{\mathcal{G}}\left(w, [r(\phi, \tau, t')]_{t' \in t+I}\right), \tag{5}$$

$$r\left(\mathcal{F}_I^w \phi, \tau, t\right) := \oplus_I^{\mathcal{F}}\left(w, [r(\phi, \tau, t')]_{t' \in t+I}\right), \tag{6}$$

where the aggregation functions $\otimes^{\wedge} : \mathbb{R}^N_{\geq 0} \times \mathbb{R}^N \to \mathbb{R}, \oplus^{\vee} : \mathbb{R}^N_{\geq 0} \times \mathbb{R}^N \to \mathbb{R}, \otimes_I^{\mathcal{G}} : \mathbb{R}^{|I|}_{\geq 0} \times \mathbb{R}^{|I|} \to \mathbb{R}$, and $\oplus_I^{\mathcal{F}} : \mathbb{R}^{|I|}_{\geq 0} \times \mathbb{R}^{|I|} \to \mathbb{R}$ correspond to the operators $\wedge, \vee, \mathcal{G}$, and $\mathcal{F}$, respectively. In practice, the infimum and supremum bounds in Equation (1) can be inferred from physical properties of the environment (we assume known map sizes in our results) or sampled trajectories. For notational simplicity, we interchangeably use $r(\phi, \tau, t)$ and $r^{\phi}_{\tau t}$.

Building from [16], we define the aggregation functions with a scaling constant $\sigma > 0$ as follows,

$$\otimes^{\wedge}(w, [r^{\phi_i}_{\tau t}]_{i=1:N}) := \frac{\sum_{i=1}^{N} w_i r^{\phi_i}_{\tau t} \exp(\frac{-r^{\phi_i}_{\tau t}}{\sigma})}{\sum_{i=1}^{N} w_i \exp(\frac{-r^{\phi_i}_{\tau t}}{\sigma})}, \tag{7}$$

$$\oplus^{\vee}(w, [r^{\phi_i}_{\tau t}]_{i=1:N}) := \frac{\sum_{i=1}^{N} w_i r^{\phi_i}_{\tau t} \exp(\frac{r^{\phi_i}_{\tau t}}{\sigma})}{\sum_{i=1}^{N} w_i \exp(\frac{r^{\phi_i}_{\tau t}}{\sigma})}, \tag{8}$$

$$\otimes_I^{\mathcal{G}}(w, [r^{\phi}_{\tau t'}]_{t' \in t+I}) := \frac{\sum_{t' \in t+I} w_{t'} r^{\phi}_{\tau t'} \exp(\frac{-r^{\phi}_{\tau t'}}{\sigma})}{\sum_{t' \in t+I} w_{t'} \exp(\frac{-r^{\phi}_{\tau t'}}{\sigma})}, \tag{9}$$

$$\oplus_I^{\mathcal{F}}(w, [r^{\phi}_{\tau t'}]_{t' \in t+I}) := \frac{\sum_{t' \in t+I} w_{t'} r^{\phi}_{\tau t'} \exp(\frac{r^{\phi}_{\tau t'}}{\sigma})}{\sum_{t' \in t+I} w_{t'} \exp(\frac{r^{\phi}_{\tau t'}}{\sigma})}. \tag{10}$$

We empirically found that decreasing σ better approximates the max/min aggregations, but can increase training instability due to larger gradients, while increasing σ too much can lead to nonsensical explanations.

C. Problem Statement

We consider the following binary classification task. Let $\mathcal{T} = \{(\tau_i, l_i)\}_{i=1:N_{\text{data}}}$ be a labeled dataset, where $\tau_i \in \mathbb{R}^{n \times H}$ is the i^{th} trajectory with label $l_i \in \{1, -1\}$ and $N_{\text{data}} \in \mathbb{Z}_{>0}$ is the number of data points in the dataset.

Problem 1. Given a dataset $\mathcal{T}$, find a wSTL specification ϕ that balances classification accuracy with conciseness, consistency, and strictness.

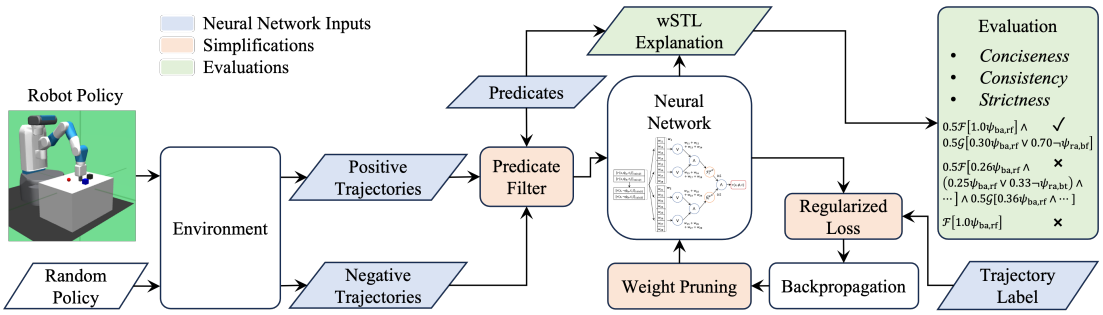


Fig. 2. We propose a neuro-symbolic approach for generating an explanation of a given robot policy using wSTL. Our network architecture, TLNET, includes a simplification process that ensures resulting explanations balance classification accuracy with conciseness, consistency, and strictness.

III. METHODOLOGY

We first introduce our neural network architecture, TLNET, which allows us to represent wSTL structures in a fully differentiable form. We then introduce our simplification process, which enhances the explainability of inferred explanations, and our trajectory sampling process. Finally, we propose novel evaluation metrics for explainability—*conciseness*, *consistency*, and *strictness*—and provide theoretical justification for our network architecture. Figure 2 and Algorithm 1 summarize our overall approach.

A. Neural Network Architecture

Let $\Psi = \{\psi_i\}_{i=1:N_{\mathrm{AP}}}$ be a predefined set of atomic predicates and $\Psi_\neg = \{\neg\psi_i\}_{i=1:N_{\mathrm{AP}}}$ be a set of their negations. We focus on representing normalized wSTL structures composed of temporal operators containing these predicates in conjunctive normal form (CNF), as follows,

$$\phi = 0.5\mathcal{F}_I^{\bar{w}}\left[\bigwedge_{i=1}^{N_{\mathrm{AP}}} w_i^{\mathcal{F}} \bigvee_{j=1}^{2N_{\mathrm{AP}}} w_{ij}^{\mathcal{F}}\psi_j\right] \wedge 0.5\mathcal{G}_I^{\bar{w}}\left[\bigwedge_{i=1}^{N_{\mathrm{AP}}} w_i^{\mathcal{G}} \bigvee_{j=1}^{2N_{\mathrm{AP}}} w_{ij}^{\mathcal{G}}\psi_j\right], \tag{11}$$

where $w_i^{\mathcal{F}}, w_{ij}^{\mathcal{F}}, w_i^{\mathcal{G}}, w_{ij}^{\mathcal{G}} \in [0,1]$ are importance weights and $\psi_j \in \Psi \cup \Psi_\neg$ are predefined atomic predicates. We define the weights $\bar{w}$ for temporal operators over interval $I = [0, H]$ to be $\bar{w} = [\bar{w}_k = 1/|I|]_{k\in I}$. We define the conjunction weights of the temporal clauses to be 0.5 to equally weigh the importance of the task ($\mathcal{F}$) and constraint ($\mathcal{G}$) parts of the explanation. We choose this class of structures because they allow us to represent a general class of robot policies aiming to complete a task while satisfying a set of constraints [14], [21].

We represent specifications of the form given in Equation (11) by designing a neural network $f_{\mathbf{w}}$ parameterized by $\mathbf{w}$, where $\mathbf{w}$ defines the importance weights of the currently represented specification $\phi_{\mathbf{w}}$. The input to the neural network is a sequence of the robustness values for each predicate $\psi \in \Psi \cup \Psi_\neg$ for each timestep in a trajectory τ. The output is the robustness of the current specification, $r(\phi_{\mathbf{w}}, \tau)$, for that trajectory. The neural network is trained to minimize a regularized classification loss (discussed in Section III-B), given a labeled dataset $\mathcal{T}$.

Algorithm 1 Explanation Generation

Input: Dataset $\mathcal{T}$, predicates $\Psi = \{\psi_i\}_{i=1:N_{\mathrm{AP}}}$, similarity threshold S_{th}, number of pruning iterations N_{pr}, number of weights to prune $N_{\mathbf{w}}$, loss function $\tilde{\mathcal{L}}$.
Output: Explanation ϕ

```
Evaluate robustness of trajectories in T using Ψ.
Filter out predicates whose similarities are above S_th.
Initialize the neural network f_w with parameters w.
Optimize w to minimize L̃.
for i = 1 : N_pr do
    Prune zeroed and smallest N_w parameters in w.
    Re-normalize and optimize w to minimize L̃.
    if only one non-zero weight in w^F/w^G then break.
return φ generated from w and Ψ using Equation (12).
```

We reduce the number of parameters to be optimized by distributing the weights in Equation (11), resulting in a specification of the following form,

$$\phi_{\mathbf{w}} = 0.5\mathcal{F}_I^{\bar{w}}\left[\bigwedge_{i=1}^{N_{\mathrm{AP}}} \bigvee_{j=1}^{2N_{\mathrm{AP}}} \tilde{w}_{ij}^{\mathcal{F}}\psi_j\right] \wedge 0.5\mathcal{G}_I^{\bar{w}}\left[\bigwedge_{i=1}^{N_{\mathrm{AP}}} \bigvee_{j=1}^{2N_{\mathrm{AP}}} \tilde{w}_{ij}^{\mathcal{G}}\psi_j\right], \tag{12}$$

where $\mathbf{w} = \mathbf{w}^{\mathcal{F}} \parallel \mathbf{w}^{\mathcal{G}}$, $\mathbf{w}^{\mathcal{F}} = [\tilde{w}_{ij}^{\mathcal{F}}]_{i=1:N_{\mathrm{AP}}, j=1:2N_{\mathrm{AP}}}$, $\mathbf{w}^{\mathcal{G}} = [\tilde{w}_{ij}^{\mathcal{G}}]_{i=1:N_{\mathrm{AP}}, j=1:2N_{\mathrm{AP}}}$, $\tilde{w}_{ij}^{\mathcal{F}} = w_i^{\mathcal{F}} w_{ij}^{\mathcal{F}}$, and $\tilde{w}_{ij}^{\mathcal{G}} = w_i^{\mathcal{G}} w_{ij}^{\mathcal{G}}$. This weight distribution reduces the number of optimized parameters from $2N_{\mathrm{AP}} + 4N_{\mathrm{AP}}^2$ to $4N_{\mathrm{AP}}^2$. We justify this weight distribution operation in Section III-E. Figure 3 shows a TLNET architecture for a dataset containing two predicates.

B. Simplification Process

A key component of TLNET is its simplification process, which aims to produce more human-interpretable wSTL explanations while preserving classification performance. This process consists of three steps: a predicate filter, loss regularization, and weight pruning.

1) Predicate Filter: As a pre-processing step, we implement a predicate filter that eliminates predicates that exhibit similar behavior across both positive and negative trajectory sets, as they contribute little to discriminating between those

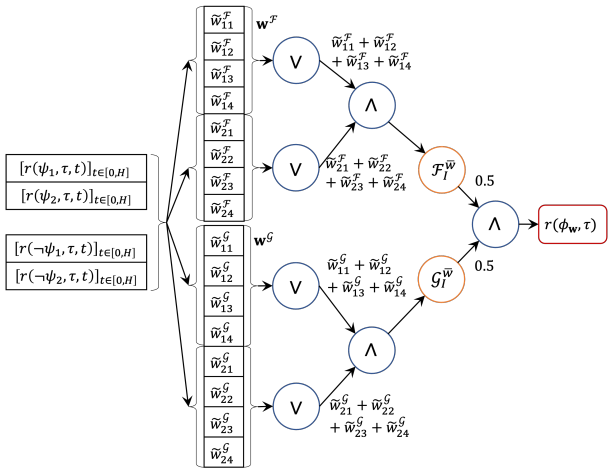


Fig. 3. Example TLNET architecture with two predicates, where $\phi_{\mathbf{w}}$ is the explanation generated from $\mathbf{w}$.

sets and may introduce unnecessary complexity. Given an arbitrary dataset of trajectories $\mathcal{D}$ and an atomic predicate ψ, we define the *robustness distribution* for ψ as,

$$d(\mathcal{D},\psi) = \frac{1}{|\mathcal{D}|}\left[|\mathcal{D}_1|,|\mathcal{D}_2|,|\mathcal{D}_3|\right], \tag{13}$$

$$\mathcal{D}_1 := \{\,\tau \mid r(\psi,\tau,t) \geq 0 \quad \forall t\in[0,H],\tau\in\mathcal{D}\,\}, \tag{14}$$

$$\mathcal{D}_2 := \mathcal{D}\setminus(\mathcal{D}_1\cup\mathcal{D}_3), \tag{15}$$

$$\mathcal{D}_3 := \{\,\tau \mid r(\psi,\tau,t) < 0 \quad \forall t\in[0,H],\tau\in\mathcal{D}\,\}. \tag{16}$$

That is, $\mathcal{D}_1$ is the set of trajectories where ψ is always satisfied, $\mathcal{D}_2$ is the set where ψ is satisfied at least once, and $\mathcal{D}_3$ is the set where ψ is never satisfied.

Now, let $\mathcal{T}_p = \{(\tau,l)|l=1,(\tau,l)\in\mathcal{T}\}$ be the positive set of trajectories in $\mathcal{T}$ and $\mathcal{T}_n = \mathcal{T}\setminus\mathcal{T}_p$ be the negative set. For each predicate $\psi\in\Psi$, we compute $d(\mathcal{T}_p,\psi)$ and $d(\mathcal{T}_n,\psi)$, and then calculate the cosine similarity S between those distributions. We then remove predicates from the set Ψ (and their negations from $\Psi_\neg$) which have a cosine similarity $S\geq S_{\text{th}}$, where S_{th} is a user-defined threshold.

2) Loss Regularization: The regularization step uses two regularizers to improve the conciseness of inferred explanations. The first regularizer, a *temporal clause regularizer*, R_{T}, is defined as,

$$R_{\text{T}}(\mathbf{w}) := \sum_{j=1}^{N_{\text{AP}}} \max\left(\sum_{i=1}^{N_{\text{AP}}}\tilde{w}^{\mathcal{F}}_{ij}, \sum_{i=1}^{N_{\text{AP}}}\tilde{w}^{\mathcal{F}}_{i(j+N_{\text{AP}})}\right) \max\left(\sum_{i=1}^{N_{\text{AP}}}\tilde{w}^{\mathcal{G}}_{ij}, \sum_{i=1}^{N_{\text{AP}}}\tilde{w}^{\mathcal{G}}_{i(j+N_{\text{AP}})}\right). \tag{17}$$

Intuitively, this regularizer encourages the task and constraint clauses to contain different atomic predicates, and is independent of clause sparsity.

The second regularizer, a *disjunctive clause regularizer*, R_{D}, is defined as,

$$R_{\text{D}}(\mathbf{w}^{\mathcal{O}}) := \sum_{i=1}^{N_{\text{AP}}-1}\sum_{j=i+1}^{N_{\text{AP}}}\sum_{k=1}^{2N_{\text{AP}}}\tilde{w}^{\mathcal{O}}_{ik}\tilde{w}^{\mathcal{O}}_{jk}, \tag{18}$$

where $\mathbf{w}^{\mathcal{O}}\in\{\mathbf{w}^{\mathcal{F}},\mathbf{w}^{\mathcal{G}}\}$ defines the weights within one of our temporal operators and $\tilde{w}^{\mathcal{O}}_{ij},\tilde{w}^{\mathcal{O}}_{jk}\in\mathbf{w}^{\mathcal{O}}$. Intuitively, this regularizer encourages disjunctive clauses within a temporal clause to contain different predicates, and is again independent of clause sparsity.

Given $(\tau,l)\in\mathcal{T}$, our overall loss, $\tilde{\mathcal{L}}$, is then,

$$\tilde{\mathcal{L}}(\tau,l,\mathbf{w}) = \mathcal{L}(\tau,l,\mathbf{w}) + \lambda_{R_{\text{T}}}R_{\text{T}}(\mathbf{w}) + \lambda_{R_{\text{D}}}[R_{\text{D}}(\mathbf{w}^{\mathcal{F}}) + R_{\text{D}}(\mathbf{w}^{\mathcal{G}})], \tag{19}$$

where $\mathcal{L}$ is a classification loss and $\lambda_{R_{\text{T}}},\lambda_{R_{\text{D}}}\in\mathbb{R}_{\geq 0}$ are regularizer hyperparameters. We use the classification loss from [16],

$$\mathcal{L}(\tau,l,\mathbf{w}) = \exp(-\zeta l r(\phi_{\mathbf{w}},\tau)), \tag{20}$$

where $\zeta\in\mathbb{R}_{>0}$ is a hyperparameter.

3) Weight Pruning: The weight pruning step further simplifies the currently inferred explanation by removing unnecessary weights. First, all weights with a value of zero are removed from the network. Then, among the remaining weights, the $N_{\text{w}}\in\mathbb{Z}_{>0}$ smallest weights—where N_{w} is a user-specified hyperparameter—are removed to eliminate the least contributing parts of the specification. The remaining weights are then normalized. The pruning process is performed $N_{\text{pr}}\in\mathbb{Z}_{>0}$ times and terminates if there is only one non-zero weight in $\mathbf{w}^{\mathcal{F}}$ or $\mathbf{w}^{\mathcal{G}}$ to ensure the desired task-constraint structure. Empirically, increasing N_{w} leads to training instability, since pruning changes the computation graph structure of the neural network. Decreasing N_{pr} can lead to unnecessarily complex explanations.

C. Trajectory Sampling

We obtain positive trajectories in $\mathcal{T}$ by executing the target policy in the environment. These trajectories can also be sampled from existing datasets. We obtain negative trajectories in $\mathcal{T}$ by executing a random-walk policy in the environment.

D. Explainability Evaluation Metrics

Beyond classification performance, TLNET also emphasizes the human interpretability of inferred explanations. We quantify this emphasis by introducing three explainability metrics—*conciseness*, *consistency*, and *strictness*.

Let ϕ be a wSTL explanation we want to evaluate and N be the number of temporal clauses that we expect to appear in the explanation. For example, in this work, we expect the inferred explanations to have one $\mathcal{F}$ clause and one $\mathcal{G}$ clause (thus $N=2$) because we assume there should be a task component and a constraint component. Now, let ϕ_n be the n-th expected temporal clause in ϕ, where $\phi_n=\emptyset$ if that clause does not exist in ϕ. Let the (unweighted) STL counterpart of ϕ_n be $\hat{\phi}_n$, where $n\in\{1,\dots N\}$. For example, in our work, if $\phi = \mathcal{G}(0.1\psi_1\wedge 0.9\psi_2)$, then $\phi_1=\emptyset$, $\hat{\phi}_1=\emptyset$, $\phi_2=\mathcal{G}(0.1\psi_1\wedge 0.9\psi_2)$, and $\hat{\phi}_2=\mathcal{G}(\psi_1\wedge\psi_2)$.

1) Conciseness: We define the *conciseness* of ϕ as,

$$\text{Conciseness} := \frac{1}{N}\sum_{n=1}^{N}\begin{cases}\frac{1}{m_n} & \text{if } m_n \neq 0,\\ 0 & \text{otherwise,}\end{cases}$$

where m_n is the number of predicates present in n-th expected temporal clause ϕ_n. Conciseness ranges from 0 to 1, with higher scores implying a shorter explanation with at least one predicate inside each expected temporal clause.

2) Consistency: Let $\Phi = [\phi^k]_{k=1:K}$ be a sequence of K explanations for the same policy (e.g., from different training seeds). Then, let $\Phi_n = [\phi_n^k]_{k=1:K}$ be the sequence of the n-th expected temporal clauses in Φ, where ϕ_n^k is the n-th expected temporal clause for ϕ^k. Let the STL counterpart of Φ_n be $\hat{\Phi}_n$. Then, let the set of unique elements of $\hat{\Phi}_n$ be $\Sigma_n = \{\hat{\phi}_n \in \hat{\Phi}_n\}$. We define the *consistency* of Φ as,

$$\text{Consistency} := \frac{1}{N}\sum_{n=1}^{N}\begin{cases}\max_{\sigma_n\in\Sigma_n}\sum_{\hat{\phi}_n\in\hat{\Phi}_n}\frac{\mathbb{I}(\sigma_n=\hat{\phi}_n)}{K|\Sigma_n|} & \text{if } \Sigma_n \neq \emptyset,\\ 0 & \text{otherwise.}\end{cases}$$

Consistency ranges from 0 to 1, with higher scores indicating greater agreement in clause structures across explanations, reflecting structural stability in learned explanations.

3) Strictness: Finally, let P be the number of potentially present predicates in ϕ (e.g., $P = 2N_{\text{AP}}$ in TLNET). Denote the numbers of conjunctions and disjunctions in ϕ_n as C_n and D_n, respectively. We define the *strictness* of ϕ as,

$$\text{Strictness} := \frac{1}{N}\sum_{n=1}^{N}\begin{cases}\frac{1}{P-C_n+D_n} & \text{if } \phi_n \neq \emptyset,\\ 0 & \text{otherwise.}\end{cases}$$

Strictness ranges from 0 to 1, with higher scores implying stricter logical expressions by rewarding conjunctions and penalizing disjunctions. For example, given a temporal clause $\phi_0 = \mathcal{F}[(0.5\psi_0 \vee 0.25\psi_1) \wedge 0.25\psi_2]$ and $P = 6$, we get $C_0 = 1$ and $D_0 = 1$, with strictness of 0.167 for ϕ_0.

E. Weight Distribution in wSTL

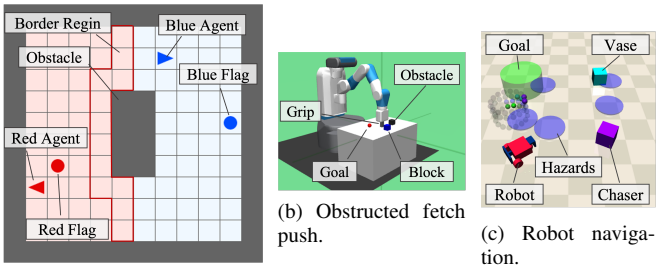
A key assumption in TLNET is that the weights can be distributed across Boolean and temporal clauses. We now prove that such a distribution is theoretically sound. The normalized weights of a wSTL formula in CNF can be syntactically distributed across Boolean operators as follows.[1]

Theorem 1 (Syntactic Weight Distributivity of Boolean Operators)**.** A normalized wSTL formula ϕ in CNF can be represented as follows,

$$\phi = \bigwedge_{i=1}^{n} w_i \bigvee_{j=1}^{m_i} w_{ij}(\neg)\psi_{ij} \equiv \bigwedge_{i=1}^{n}\bigvee_{j=1}^{m_i}\tilde{w}_{ij}(\neg)\psi_{ij}, \tag{21}$$

where $n, m_i \in \mathbb{Z}_{>0}$, $w_i, w_{ij} \in \mathbb{R}_{>0}$, $\tilde{w}_{ij} := w_i w_{ij}$, $\sum_{i=1}^{n} w_i = \sum_{j=1}^{m_i} w_{ij} = 1$, and ψ_{ij} are predicates. Furthermore, the weights $\tilde{w}_{ij}$ satisfy $\sum_{i=1}^{n}\sum_{j=1}^{m_i}\tilde{w}_{ij} = 1$.

[1]Syntactic weight distributivity also holds for temporal operators by distributing operator weights to the normalized interval weights followed by distributions to the inner Boolean structures.



(a) CtF environment.

(b) Obstructed fetch push.

(c) Robot navigation.

Fig. 4. Initial states of our test environments with relevant objects shown.

Proof.

$$\begin{aligned}\{\text{LHS}\} &= w_1(w_{11}(\neg)\psi_{11} \vee \ldots \vee w_{1m_1}(\neg)\psi_{1m_1})\\ &\quad \ldots \wedge w_n(w_{n1}(\neg)\psi_{n1} \vee \ldots \vee w_{nm_n}(\neg)\psi_{nm_n})\\ &= (w_1 w_{11}(\neg)\psi_{11} \vee \ldots \vee w_1 w_{1m_1}(\neg)\psi_{1m_1})\\ &\quad \ldots \wedge (w_n w_{n1}(\neg)\psi_{n1} \vee \ldots \vee w_n w_{nm_n}(\neg)\psi_{nm_n})\\ &= \bigwedge_{i=1}^{n}\bigvee_{j=1}^{m_i} w_i w_{ij}(\neg)\psi_j = \{\text{RHS}\}.\end{aligned}$$

□

IV. RESULTS

We perform a series of empirical experiments to compare our method to relevant baselines for robotic tasks. We consider three baselines for inference of wSTL explanations, each adapted to model our normalized wSTL structures (given by Equation (12)):

- *NN-TLI* [20]: this method implements a greedy pruning strategy that independently removes network weights that do not affect classification accuracy after optimization. We removed their weight-rounding process (to convert to STL) to consider the structure in Equation (12) and used the aggregation functions given in Equations (7) to (10), since our empirical results showed more training stability with those functions.
- *STONE (top-3, top-5)* [16], [19]: this method retains the 3 or 5 largest network weights after training.
- *DT (STL)* [13]: this method uses decision trees to mine STL (not wSTL) explanations.

The first two baselines, NN-TLI and STONE (top-3, top-5), were implemented using the same neural network architecture as TLNET to ensure a consistent basis for comparison. For each policy we explain, we collected 500 positive trajectories using the trained policy and 500 negative trajectories using a random-walk policy. Each dataset was split 80% and 20% for training and evaluation, respectively. Hyperparameter values are summarized in Table I. Our code, including videos of target policies, is available online[2].

A. Test Environments

We consider the three environments shown in Figure 4, which include discrete and continuous state-action spaces, navigation and manipulation tasks, and varied predicates.

[2]https://github.com/LIRA-illinois/tlnet

TABLE I
HYPERPARAMETER VALUES AND THEIR RANGES.

Environments	σ	ζ	S_{th}	$\lambda_{R_{\text{T}}}$	$\lambda_{R_{\text{D}}}$	N_{pr}	N_{w}	H
CtF	0.5	1.0	0.99	0.01	0.1	34	1	100
Fetch Push	0.5	1.0	0.99	0.01	0.1	20	1	200
Robot Nav.	0.5	1.0	0.99	0.01	0.1	28	1	400
Range ($\in$)	$\mathbb{R}_{>0}$	$\mathbb{R}_{>0}$	[-1, 1]	$\mathbb{R}_{\geq 0}$	$\mathbb{R}_{\geq 0}$	$\mathbb{Z}_{>0}$	$\mathbb{Z}_{>0}$	$\mathbb{Z}_{>0}$

1) Capture-the-flag (CtF): CtF is a discrete grid-world environment with adversarial dynamics where two agents compete to capture each other's flags, based on [14]. If both agents occupy adjacent cells in a given territory, the defending agent eliminates the other with 75% probability. We considered blue agent policies optimized for five scenarios. The first four considered different red agent heuristic policies (*capture*, *fight*, *patrol*, *mix*) with the same reward function (capture: +1, kill: +0.5, being captured: -1, timestep penalty: -0.01). The fifth scenario (*capture 0*) used a *capture* red agent with the reward function modified to remove the kill reward (kill: 0). The red agent policies are defined as follows: *capture* pursues the flag; *fight* pursues the blue agent; *patrol* defends a boundary region; and *mix* uses *fight* when the blue agent is in the red territory and *capture* otherwise. Red agents perform a random action 25% of the time. We defined eight atomic predicates based on the defeated ("df") status $\{1, -1\}$ of the red agent and Euclidean distances among the blue agent ("ba"), blue flag ("bf"), blue territory ("bt"), red agent ("ra"), red flag ("rf"), and obstacle ("ob"): $\psi_{\text{ba,bt}}$, $\psi_{\text{ba,ob}}$, $\psi_{\text{ba,ra}}$, $\psi_{\text{ba,rf}}$, $\psi_{\text{ra,bf}}$, $\psi_{\text{ra,bt}}$, $\psi_{\text{ra,df}}$, $\psi_{\text{ra,ob}}$.

2) Fetch Push: We also consider a continuous control task based on the Fetch mobile manipulator [22]. The robot must push a blue block ("b") to a red target ("t") using its grip ("g"), without disturbing a black obstacle ("o"). The state and action spaces are continuous ($s \in \mathbb{R}^{55}, a \in \mathbb{R}^4$). Eight predicates were defined based on the distances between the grip, block, target, and obstacle, including drop-related ("d") conditions: ψ_{gb}, ψ_{gt}, ψ_{go}, ψ_{bt}, ψ_{bo}, ψ_{ot}, ψ_{bd}, ψ_{od}.

3) Robot Navigation: Finally, we extended the CarGoal-v0 environment [23] to create a more complex adversarial navigation task. The ego robot ("e") must reach a green goal ("g") while avoiding a violet chaser ("c") moving with a fixed speed ($v_c = 7.5 \times 10^{-4}$). Static hazards ("h") and a movable vase ("v") are present but not penalized. The state and action dimensions are $s \in \mathbb{R}^{88}$ and $a \in \mathbb{R}^2$, respectively. Ten distance-based predicates were defined, involving Euclidean distances between the robot, the chaser, the goal, and the hazards: ψ_{ec}, ψ_{eg}, ψ_{eh}, ψ_{ev}, ψ_{gc}, ψ_{gh}, ψ_{gv}, ψ_{hc}, ψ_{hv}, ψ_{vc}.

We included predicates irrelevant to task success in each environment (e.g., $\psi_{\text{ra,ob}}$, ψ_{ot}, ψ_{hv}) to evaluate whether the considered methods could eliminate them from their explanations. All predicates are distance-based (e.g., $d_{\text{ba,ra}} < 0.5$) except for $\psi_{\text{ra,df}}$ (binary). Average TLNET optimization times were 170.8 s (CtF), 118.3 s (Fetch Push), and 116.1 s (Robot Navigation).[3]

[3]With an Intel Xeon Silver 4314 CPU and NVIDIA RTX A4000 GPU.

B. Policy Optimization

TLNET is a general framework to generate explanations of robot policies, though, we focused on policies optimized by deep reinforcement learning algorithms in this work. We used Stable-Baselines3 [24] for training all policies. For the CtF and navigation environments, we used PPO [25]; for the Fetch Push task, we used SAC [26] with hindsight experience replay (HER) [27]. Hyperparameters were taken from prior work [14], [23], [28]. For each policy, we performed 10 independent optimization runs with random initialization and collected classification accuracy and explainability metrics.

C. Experimental Results

1) Benefits & Limitations of TLNET*:* Table II presents the most frequent explanation obtained by each method for each scenario. Across all scenarios, TLNET consistently inferred both task and constraint clauses, while also producing concise and relatively comprehensible explanations. In comparison, NN-TLI typically yielded verbose explanations that are difficult to understand, while STONE often captured only one temporal clause, resulting in incomplete explanations.

The explanations produced by TLNET are also qualitatively reasonable. For all CtF scenarios, TLNET correctly inferred the task clause as $\psi_{\text{ba,rf}}$ (*"blue agent captures the red flag"*)—no other method correctly inferred this task for every scenario. The inferred CtF constraints are also syntactically reasonable. For example, in the Capture scenario, the blue agent (global) constraint includes *"red agent not on the blue flag"* because the *capture* red policy only targets the blue flag, which requires the blue agent to learn to prevent the red agent from reaching the blue flag. Other scenarios, such as Fight and Patrol, did not include this predicate in the inferred constraint because those red policies do not target the blue flag. In comparison, the CtF constraints inferred by NN-TLI are generally too verbose to understand, while STONE did not infer any constraints. The DT (STL) baseline often produced a task- or constraint-only STL explanation, typically with semantically-inaccurate clauses. For Fetch Push, TLNET also correctly inferred the task clause as ψ_{bt} (*"block reaches the target"*), which no other method correctly inferred. While TLNET inferred part of the expected constraint ($\psi_{\text{gb}}, \psi_{\text{bt}}$), it missed an expected constraint $\neg\psi_{\text{go}}$ ("grip does not touch the obstacle"), likely due to insufficient occurrences of the grip touching the obstacle in the sampled data. NN-TLI also inferred part of the expected constraint, but also missed the $\neg\psi_{\text{go}}$ constraint. Meanwhile, both variations of STONE inferred an incorrect constraint of ψ_{bt} (*"block reaches the target"*), since this is actually the task and not possible to always satisfy.

For Robot Navigation, all methods correctly inferred the task clause but TLNET was the only one to infer the correct constraint clause as $\neg\psi_{\text{ec}}$ (*"the robot does not hit the chaser"*).

As shown in Tables III and IV, TLNET also outperformed all baselines in conciseness, consistency, and strictness, with marginal differences in classification accuracy, except for DT (STL) in consistency. Across all scenarios, TLNET achieved

[4]Statistical significance (Friedman test, Dunn-Bonferroni analysis) against the second best values indicated with * ($p < 0.05$), ** ($p < 0.01$).

TABLE II
COMPARISON OF THE MOST FREQUENTLY INFERRED EXPLANATIONS WITH FREQUENCIES (%).

Scenario	TLNET (Ours)		NN-TLI		STONE (top-3)		STONE (top-5)		DT	
CtF Capture	$0.5\mathcal{F}_L^{\bar{w}}[1.0\psi_{\mathrm{ba,rf}}]\wedge 0.5\mathcal{G}_I^{\bar{w}}[0.30\psi_{\mathrm{ba,rf}}\vee 0.70\neg\psi_{\mathrm{ra,bf}}]$	60%	$0.5\mathcal{F}_I^{\bar{w}}[0.26\psi_{\mathrm{ba,rf}}\wedge(0.25\psi_{\mathrm{ba,rf}}\vee 0.33\neg\psi_{\mathrm{ra,bt}})\wedge(0.09\neg\psi_{\mathrm{ba,bt}}\vee 0.07\neg\psi_{\mathrm{ra,bt}})]\wedge 0.5\mathcal{G}_I^{\bar{w}}[0.36\psi_{\mathrm{ba,rf}}\vee 0.64\neg\psi_{\mathrm{ra,bf}}]$	20%	$\mathcal{F}_I^{\bar{w}}[0.73\neg\psi_{\mathrm{ra,bt}}\wedge 0.27\neg\psi_{\mathrm{ba,bt}}]$	50%	$\mathcal{F}_I^{\bar{w}}[0.83\neg\psi_{\mathrm{ra,bt}}\wedge 0.17\psi_{\mathrm{ba,rf}}]$	60%	$\mathcal{F}_I[\psi_{\mathrm{ba,rf}}\vee\psi_{\mathrm{ra,bf}}\vee\neg\psi_{\mathrm{ba,bt}}]$	100%
CtF Fight	$0.5\mathcal{F}_L^{\bar{w}}[1.0\psi_{\mathrm{ba,rf}}]\wedge 0.5\mathcal{G}_I^{\bar{w}}[0.33\psi_{\mathrm{ba,rf}}\vee 0.67\neg\psi_{\mathrm{ba,ra}}]$	70%	$0.5\mathcal{F}_L^{\bar{w}}[0.77\psi_{\mathrm{ba,rf}}\wedge(0.15\psi_{\mathrm{ba,rf}}\vee 0.08\psi_{\mathrm{ra,bf}})]\wedge 0.5\mathcal{G}_I^{\bar{w}}[(0.15\neg\psi_{\mathrm{ba,bt}}\vee 0.11\neg\psi_{\mathrm{ra,df}})\wedge 0.06\psi_{\mathrm{ba,rf}}\vee 0.04\neg\psi_{\mathrm{ra,df}})\wedge(0.06\psi_{\mathrm{ba,rf}}\vee 0.06\neg\psi_{\mathrm{ba,bt}}\vee 0.04\neg\psi_{\mathrm{ra,df}})\wedge(0.02\psi_{\mathrm{ba,ra}}\vee 0.19\psi_{\mathrm{ba,rf}}\vee 0.16\neg\psi_{\mathrm{ba,bt}}\vee 0.11\neg\psi_{\mathrm{ra,df}})]$	70%	$\mathcal{F}_I^{\bar{w}}[1.0\psi_{\mathrm{ba,rf}}]$	100%	$\mathcal{F}_I^{\bar{w}}[1.0\psi_{\mathrm{ba,rf}}]$	100%	$\mathcal{F}_I[\psi_{\mathrm{ba,rf}}]\wedge\mathcal{G}_I[\psi_{\mathrm{ba,bt}}]$	100%
CtF Patrol	$0.5\mathcal{F}_L^{\bar{w}}[1.0\psi_{\mathrm{ba,rf}}]\wedge 0.5\mathcal{G}_I^{\bar{w}}[0.52\psi_{\mathrm{ba,rf}}\vee 0.48\neg\psi_{\mathrm{ra,bt}}]$	70%	$0.5\mathcal{F}_L^{\bar{w}}[0.35\psi_{\mathrm{ba,rf}}\wedge(0.09\psi_{\mathrm{ba,rf}}\vee 0.55\psi_{\mathrm{ra,df}})]\wedge 0.5\mathcal{G}_I^{\bar{w}}[(0.29\psi_{\mathrm{ba,rf}}\vee 0.04\psi_{\mathrm{ra,bf}}\vee 0.07\neg\psi_{\mathrm{ra,df}})\wedge(0.30\psi_{\mathrm{ba,rf}}\vee 0.19\psi_{\mathrm{ra,bf}}\vee 0.02\neg\psi_{\mathrm{ba,bt}})\wedge 0.04\psi_{\mathrm{ra,bf}}\wedge 0.05\neg\psi_{\mathrm{ra,bf}}]$	30%	$\mathcal{F}_I^{\bar{w}}[1.0\psi_{\mathrm{ra,df}}]$	100%	$\mathcal{F}_I^{\bar{w}}[1.0\psi_{\mathrm{ra,df}}]$	90%	$\mathcal{G}_I[\neg\psi_{\mathrm{ba,rf}}\wedge\psi_{\mathrm{ba,bt}}]$	100%
CtF Mix	$0.5\mathcal{F}_L^{\bar{w}}[1.0\psi_{\mathrm{ba,rf}}]\wedge 0.5\mathcal{G}_I^{\bar{w}}[0.44\psi_{\mathrm{ba,rf}}\vee 0.56\neg\psi_{\mathrm{ra,bf}}]$	60%	$0.5\mathcal{F}_I^{\bar{w}}[1.0\psi_{\mathrm{ba,rf}}]\wedge 0.5\mathcal{G}_I^{\bar{w}}[0.21\neg\psi_{\mathrm{ra,bf}}\wedge 0.47\psi_{\mathrm{ba,rf}}\wedge(0.04\neg\psi_{\mathrm{ra,bf}}\vee 0.05\neg\psi_{\mathrm{ra,df}})\wedge(0.18\psi_{\mathrm{ba,rf}}\vee 0.03\psi_{\mathrm{ra,df}}\vee 0.04\neg\psi_{\mathrm{ra,bf}}\vee 0.02\neg\psi_{\mathrm{ra,bt}})]$	10%	$\mathcal{F}_I^{\bar{w}}[1.0\psi_{\mathrm{ba,bt}}]$	60%	$\mathcal{F}_I^{\bar{w}}[0.48\neg\psi_{\mathrm{ba,bt}}\wedge 0.52\psi_{\mathrm{ba,rf}}]$	40%	$\mathcal{F}_I[\neg\psi_{\mathrm{ba,rf}}\vee\neg\psi_{\mathrm{ba,ra}}]\wedge\mathcal{G}_I[\psi_{\mathrm{ba,rf}}]$	100%
CtF Capture 0	$0.5\mathcal{F}_L^{\bar{w}}[1.0\psi_{\mathrm{ba,rf}}]\wedge 0.5\mathcal{G}_I^{\bar{w}}[0.31\psi_{\mathrm{ba,rf}}\vee 0.69\neg\psi_{\mathrm{ra,bf}}]$	80%	$0.5\mathcal{F}_I^{\bar{w}}[0.08\psi_{\mathrm{ba,rf}}\wedge(0.40\psi_{\mathrm{ba,rf}}\vee 0.31\neg\psi_{\mathrm{ra,bt}})\wedge(0.07\neg\psi_{\mathrm{ba,bt}}\vee 0.14\neg\psi_{\mathrm{ra,bt}})]\wedge 0.5\mathcal{G}_I^{\bar{w}}[(0.33\psi_{\mathrm{ba,rf}}\vee 0.58\neg\psi_{\mathrm{ra,bf}})\wedge(0.05\psi_{\mathrm{ba,rf}}\vee 0.04\neg\psi_{\mathrm{ra,bt}})]$	20%	$\mathcal{F}_I^{\bar{w}}[0.67\neg\psi_{\mathrm{ra,bt}}\wedge 0.33\neg\psi_{\mathrm{ba,bt}}]$	60%	$\mathcal{F}_I^{\bar{w}}[0.83\neg\psi_{\mathrm{ra,bt}}\wedge 0.17\psi_{\mathrm{ba,rf}}]$	40%	$\mathcal{F}_I[\psi_{\mathrm{ba,rf}}\vee\neg\psi_{\mathrm{ba,ra}}\vee\neg\psi_{\mathrm{ba,bt}}]$	100%
Fetch Push	$0.5\mathcal{F}_I^{\bar{w}}[1.0\psi_{\mathrm{bt}}]\wedge 0.5\mathcal{G}_I^{\bar{w}}[0.41\psi_{\mathrm{gb}}\vee 0.59\psi_{\mathrm{bt}}]$	100%	$0.5\mathcal{F}_L^{\bar{w}}[0.74\psi_{\mathrm{bt}}\wedge(0.10\psi_{\mathrm{bt}}\vee 0.16\psi_{\mathrm{od}})]\wedge 0.5\mathcal{G}_I^{\bar{w}}[1.0\neg\psi_{\mathrm{bd}}]$	40%	$\mathcal{G}_I^{\bar{w}}[1.0\psi_{\mathrm{bt}}]$	100%	$\mathcal{G}_I^{\bar{w}}[1.0\psi_{\mathrm{bt}}]$	100%	$\mathcal{G}_I[\neg\psi_{\mathrm{og}}]$	100%
Robot Nav.	$0.5\mathcal{F}_L^{\bar{w}}[1.0\psi_{\mathrm{eg}}]\wedge 0.5\mathcal{G}_I^{\bar{w}}[1.0\neg\psi_{\mathrm{ec}}]$	70%	$\mathcal{F}_I^{\bar{w}}[1.0\psi_{\mathrm{eg}}]$	70%	$\mathcal{F}_I^{\bar{w}}[1.0\psi_{\mathrm{eg}}]$	100%	$\mathcal{F}_I^{\bar{w}}[1.0\psi_{\mathrm{eg}}]$	100%	$\mathcal{F}_I[\psi_{\mathrm{eg}}\vee\psi_{\mathrm{ec}}\vee\psi_{\mathrm{eh}}]$	100%

TABLE III
COMPARISON OF CONCISENESS, CONSISTENCY, STRICTNESS WITH STANDARD DEVIATION. THE BEST RESULTS ARE BOLDED.[4]

	Conciseness					**Consistency**					**Strictness**				
			STONE					STONE					STONE		
Scenario	Ours	NN-TLI	top-3	top-5	DT	Ours	NN-TLI	top-3	top-5	DT	Ours	NN-TLI	top-3	top-5	DT
CtF Capt.	$\mathbf{0.41}^{*}_{\pm 0.06}$	$0.21_{\pm 0.04}$	$0.20_{\pm 0.05}$	$0.22_{\pm 0.12}$	$0.12_{\pm 0.00}$	0.33	0.07	0.08	0.13	**0.50**	$\mathbf{0.12}^{*}_{\pm 0.01}$	$0.10_{\pm 0.01}$	$0.07_{\pm 0.01}$	$0.08_{\pm 0.02}$	$0.04_{\pm 0.00}$
CtF Fight	$\mathbf{0.39}^{*}_{\pm 0.04}$	$0.18_{\pm 0.05}$	$0.25_{\pm 0.00}$	$0.25_{\pm 0.00}$	$0.12_{\pm 0.00}$	0.68	0.09	0.50	0.50	**1.00**	$\mathbf{0.12}^{*}_{\pm 0.00}$	$0.09_{\pm 0.01}$	$0.06_{\pm 0.00}$	$0.06_{\pm 0.00}$	$\mathbf{0.12}^{*}_{\pm 0.00}$
CtF Patrol	$\mathbf{0.38}^{*}_{\pm 0.04}$	$0.18_{\pm 0.10}$	$0.25_{\pm 0.00}$	$0.28_{\pm 0.07}$	$0.50_{\pm 0.00}$	**0.63**	0.05	0.50	0.53	0.50	$\mathbf{0.12}^{*}_{\pm 0.00}$	$0.09_{\pm 0.02}$	$0.06_{\pm 0.00}$	$0.07_{\pm 0.02}$	$0.07_{\pm 0.00}$
CtF Mix	$\mathbf{0.39}^{*}_{\pm 0.06}$	$0.12_{\pm 0.06}$	$0.21_{\pm 0.05}$	$0.16_{\pm 0.09}$	$0.17_{\pm 0.00}$	0.27	0.06	0.10	0.09	**1.00**	$\mathbf{0.11}^{*}_{\pm 0.01}$	$0.07_{\pm 0.01}$	$0.06_{\pm 0.01}$	$0.07_{\pm 0.02}$	$\mathbf{0.11}^{*}_{\pm 0.01}$
CtF Capt.0	$\mathbf{0.42}^{*}_{\pm 0.04}$	$0.21_{\pm 0.05}$	$0.20_{\pm 0.04}$	$0.30_{\pm 0.13}$	$0.42_{\pm 0.00}$	0.45	0.11	0.15	0.24	**0.50**	$\mathbf{0.11}_{\pm 0.00}$	$\mathbf{0.11}_{\pm 0.02}$	$0.07_{\pm 0.00}$	$0.10_{\pm 0.03}$	$0.04_{\pm 0.00}$
Fetch Push	$\mathbf{0.42}^{*}_{\pm 0.00}$	$0.31_{\pm 0.08}$	$0.25_{\pm 0.00}$	$0.25_{\pm 0.00}$	$0.25_{\pm 0.00}$	**1.00**	0.28	0.50	0.50	0.50	$\mathbf{0.11}^{**}_{\pm 0.00}$	$\mathbf{0.11}^{**}_{\pm 0.02}$	$0.06_{\pm 0.00}$	$0.06_{\pm 0.00}$	$0.06_{\pm 0.00}$
Robot Nav.	$\mathbf{0.47}^{**}_{\pm 0.04}$	$0.22_{\pm 0.06}$	$0.25_{\pm 0.00}$	$0.25_{\pm 0.00}$	$0.12_{\pm 0.00}$	**0.68**	0.15	0.50	0.50	0.50	$\mathbf{0.10}^{**}_{\pm 0.00}$	$0.05_{\pm 0.01}$	$0.05_{\pm 0.00}$	$0.05_{\pm 0.00}$	$0.04_{\pm 0.00}$

TABLE IV
COMPARISON OF ACCURACY. THE BEST RESULTS ARE BOLDED.

	Accuracy				
			STONE		
Scenario	Ours	NN-TLI	top-3	top-5	DT
CtF Capt.	$\mathbf{0.95}_{\pm 0.02}$	$\mathbf{0.95}_{\pm 0.01}$	$\mathbf{0.95}_{\pm 0.01}$	$0.94_{\pm 0.02}$	$\mathbf{0.95}_{\pm 0.00}$
CtF Fight	$\mathbf{0.94}_{\pm 0.02}$	$0.92_{\pm 0.02}$	$\mathbf{0.94}_{\pm 0.02}$	$0.93_{\pm 0.01}$	$\mathbf{0.94}_{\pm 0.00}$
CtF Patrol	$\mathbf{0.95}_{\pm 0.02}$	$0.93_{\pm 0.02}$	$0.94_{\pm 0.01}$	$0.94_{\pm 0.02}$	$\mathbf{0.95}_{\pm 0.00}$
CtF Mix	$0.87_{\pm 0.01}$	$\mathbf{0.89}_{\pm 0.04}$	$0.88_{\pm 0.03}$	$0.88_{\pm 0.05}$	$\mathbf{0.89}_{\pm 0.00}$
CtF Capt.0	$\mathbf{0.95}_{\pm 0.02}$	$\mathbf{0.95}_{\pm 0.01}$	$\mathbf{0.95}_{\pm 0.01}$	$\mathbf{0.95}_{\pm 0.02}$	$\mathbf{0.95}_{\pm 0.00}$
Fetch Push	$\mathbf{0.99}^{**}_{\pm 0.01}$	$0.97_{\pm 0.02}$	$0.97_{\pm 0.01}$	$0.97_{\pm 0.01}$	$\mathbf{0.99}^{**}_{\pm 0.00}$
Robot Nav.	$0.97_{\pm 0.01}$	$0.96_{\pm 0.01}$	$0.96_{\pm 0.01}$	$0.96_{\pm 0.01}$	$\mathbf{0.98}_{\pm 0.01}$

up to 1.9× higher conciseness, 2.6× higher consistency, and 2.0× higher strictness. These results support the qualitative findings above, validating the explainability of our method's outputs and the use of these explainability metrics.

2) Benefit of Including Weights: We now demonstrate the benefit of including weights within inferred explanations (e.g., using wSTL rather than STL [13], [20], [29] or LTL [12]) by examining how inferred weights differ across CtF scenarios. Consider the Capture and Mix explanations in Table II. Both *capture* and *mix* red policies target the blue flag, but the *capture* policy is more aggressive in doing so. The blue agent, in response, learns slightly different policies: a more defensive policy in Capture, prioritizing the protection of its flag, and a more aggressive policy in Mix, focusing more on capturing the red flag. This policy difference is reflected in the weights in the constraint clause, where the $\neg\psi_{\mathrm{ra,bf}}$ predicate receives a higher weight in the Capture explanation (0.70) than in the Mix explanation (0.56), due to the Capture policy's prioritization of flag protection. Conversely, the Mix explanation shows a higher weight for the $\psi_{\mathrm{ba,rf}}$ predicate (0.44) than the Capture explanation (0.30), due to its prioritization of flag capture. Note that $\psi_{\mathrm{ba,rf}}$ appears in both the task and constraint clauses in these explanations. This duplication results from our design choice to fix equal weights (0.5) on the task and constraint clauses, which forces TLNET to express prioritization between the task and constraint by including relevant task predicates in the constraint clause.

Now, consider the Capture and Capture 0 scenarios, which differ in the reward to kill the red agent (0.5 vs. 0). While one might naively expect this difference to produce different policy explanations, it actually should not change the optimal blue agent policies because, to win the game, the blue agent should first defend its flag (by killing the red agent), since the red agent uses the *capture* policy. We see that TLNET successfully captured this nuance by inferring constraints with nearly identical weights, reflecting its robustness to minor

TABLE V
ABLATION STUDY OF TLNET IN THE CTF CAPTURE SCENARIO.

Features	Concise.	Consist.	Strict.	Accuracy
Full	**0.41**$_{\pm 0.06}$	**0.33**	**0.12**$_{\pm 0.01}$	**0.95**$^{*}_{\pm 0.02}$
No Filter: $_{S_{\text{th}}>1.0}$	0.29$_{\pm 0.07}$	0.17	0.11$_{\pm 0.01}$	**0.95**$^{*}_{\pm 0.01}$
No Reg.: $\lambda_{R_{\text{T}}}=\lambda_{R_{\text{D}}}=0$	0.40$_{\pm 0.09}$	0.23	**0.12**$_{\pm 0.01}$	**0.95**$^{*}_{\pm 0.02}$
No Pruning: $_{N_{\text{w}}=0}$	0.15$_{\pm 0.02}$	0.20	0.06$_{\pm 0.00}$	0.78$_{\pm 0.05}$
Argmin Negative Traj.	0.40$_{\pm 0.05}$	0.32	**0.12**$_{\pm 0.01}$	**0.95**$^{*}_{\pm 0.01}$

reward differences that still lead to similar policies. NN-TLI and STONE did not infer similar constraints across these scenarios. More broadly, this result also demonstrates the importance of inferring explanations from actual observed behaviors of RL agents rather than simply looking at their specified reward function.

3) Importance of TLNET *Components:* Finally, the ablation study results shown in Table V show that filtering, regularization, and pruning are all important to the performance of TLNET. Removing predicate filtering notably reduced conciseness and consistency, removing regularization decreased conciseness and significantly reduced consistency with no accuracy improvement, and removing pruning largely reduced all metrics. We also tested an argmin approach to sample negative trajectories, where rollouts used the lowest-probability action in the target policy, and saw no significant difference in results compared to random sampling.

V. CONCLUSION

We introduce TLNET, a neuro-symbolic framework for generating wSTL explanations of robot policies through predicate filtering, loss regularization, and weight pruning. We also propose novel explainability metrics—conciseness, consistency, and strictness—to assess explanation quality beyond classification accuracy. Experiments in three simulated robotic environments show our method outperforms baselines on these metrics while retaining high classification accuracy. Future directions include real-world deployment with online inference (which would require improving sample efficiency and handling predicate uncertainty from sensor noise and partial observability), learning task-constraint weights with a stabilizer, improving negative data generation, exploring post-hoc enforcement of the task-constraint format, and user studies.

REFERENCES

[1] S. Milani, N. Topin, M. Veloso, and F. Fang, "Explainable Reinforcement Learning: A Survey and Comparative Review," *ACM Comput. Surv.*, vol. 56, no. 7, pp. 168:1–168:36, 2024.

[2] A. Iucci, A. Hata, A. Terra, R. Inam, and I. Leite, "Explainable Reinforcement Learning for Human-Robot Collaboration," in *2021 20th Int. Conf. Advanced Robot.*, Feb. 2021, pp. 927–934.

[3] E. M. Kenny, M. Tucker, and J. Shah, "Towards Interpretable Deep Reinforcement Learning with Human-Friendly Prototypes," in *The 11th Int. Conf. Learn. Representations*, Sept. 2022.

[4] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. J. Rezende, "Towards interpretable reinforcement learning using attention augmented agents," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, Dec. 2019, no. 1107, pp. 12 360–12 369.

[5] N. Puri, S. Verma, P. Gupta, D. Kayastha, S. Deshmukh, B. Krishnamurthy, and S. Singh, "Explain Your Move: Understanding Agent Actions Using Specific and Relevant Feature Attribution," in *Int. Conf. Learn. Representations*, Sept. 2019.

[6] R. Ragodos, T. Wang, Q. Lin, and X. Zhou, "ProtoX: Explaining a Reinforcement Learning Agent via Prototyping," *Adv. in Neural Inf. Process. Syst.*, vol. 35, pp. 27 239–27 252, Dec. 2022.

[7] L. Trigg, B. Morgan, A. Stringer, L. Schley, and D. F. Hougen, "Natural Language Explanation for Autonomous Navigation," in *AIAA DATC/IEEE 43rd Digit. Avionics Syst. Conf.*, Sept. 2024, pp. 1–9.

[8] I. Buzhinsky, "Formalization of natural language requirements into temporal logics: a survey," in *2019 IEEE 17th Int. Conf. on Industrial Informatics (INDIN)*, vol. 1, July 2019, pp. 400–406.

[9] M. Cosler, C. Hahn, D. Mendoza, F. Schmitt, and C. Trippel, "nl2spec: Interactively Translating Unstructured Natural Language to Temporal Logics with Large Language Models," in *Comput. Aided Verification*, 2023, pp. 383–396.

[10] E. Bartocci, C. Mateis, E. Nesterini, and D. Nickovic, "Survey on mining signal temporal logic specifications," *Information and Computation*, vol. 289, p. 104957, Nov. 2022.

[11] B. P. Bhuyan, A. Ramdane-Cherif, R. Tomar, and T. P. Singh, "Neuro-symbolic artificial intelligence: a survey," *Neural Computing and Applications*, vol. 36, no. 21, pp. 12 809–12 844, July 2024.

[12] J.-R. Gaglione, D. Neider, R. Roy, U. Topcu, and Z. Xu, "Learning Linear Temporal Properties from Noisy Data: A MaxSAT-Based Approach," in *Automat. Technol. Verification Anal.*, 2021, pp. 74–90.

[13] G. Bombara and C. Belta, "Offline and Online Learning of Signal Temporal Logic Formulae Using Decision Trees," *ACM Trans. Cyber-Phys. Syst.*, vol. 5, no. 3, pp. 22:1–22:23, 2021.

[14] M. Yuasa, H. T. Tran, and R. S. Sreenivas, "On Generating Explanations for Reinforcement Learning Policies: An Empirical Study," *IEEE Control Syst. Lett.*, vol. 8, pp. 3027–3032, 2024.

[15] N. Mehdipour, C.-I. Vasile, and C. Belta, "Specifying User Preferences Using Weighted Signal Temporal Logic," *IEEE Control Syst. Lett.*, vol. 5, no. 6, pp. 2006–2011, Feb. 2021.

[16] R. Yan, A. Julius, M. Chang, A. Fokoue, T. Ma, and R. Uceda-Sosa, "STONE: Signal Temporal Logic Neural Network for Time Series Classification," in *21st Int. Conf. Data Min.*, Dec. 2021, pp. 778–787.

[17] R. Tian, M. Cui, and G. Chen, "A Neural-Symbolic Network for Interpretable Fault Diagnosis of Rolling Element Bearings Based on Temporal Logic," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–14, 2024.

[18] N. Baharisangari, K. Hirota, R. Yan, A. Julius, and Z. Xu, "Weighted Graph-Based Signal Temporal Logic Inference Using Neural Networks," *IEEE Control Syst. Lett.*, vol. 6, pp. 2096–2101, 2022.

[19] R. Yan, T. Ma, A. Fokoue, M. Chang, and A. Julius, "Neuro-symbolic Models for Interpretable Time Series Classification using Temporal Logic Description," in *2022 IEEE Int. Conf. Data Mining*, Jan. 2022, pp. 618–627.

[20] D. Li, M. Cai, C.-I. Vasile, and R. Tron, "Learning Signal Temporal Logic through Neural Network for Interpretable Classification," in *2023 American Control Conf. (ACC)*, May 2023, pp. 1907–1914.

[21] X. Li, Z. Serlin, G. Yang, and C. Belta, "A formal methods approach to interpretable reinforcement learning for robotic planning," *Science Robotics*, vol. 4, no. 37, p. eaay6276, Dec. 2019.

[22] R. de Lazcano, K. Andreas, J. J. Tai, S. R. Lee, and J. Terry, "Gymnasium robotics," 2024. [Online]. Available: http://github.com/Farama-Foundation/Gymnasium-Robotics

[23] J. Ji, B. Zhang, J. Zhou, X. Pan, W. Huang, R. Sun, Y. Geng, Y. Zhong, J. Dai, and Y. Yang, "Safety Gymnasium: A Unified Safe Reinforcement Learning Benchmark," *Advances in Neural Inf. Process. Syst.*, vol. 36, pp. 18 964–18 993, Dec. 2023.

[24] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *J. Mach. Learn. Res.*, vol. 22, pp. 1–8, 2021.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 2017, arXiv:1707.06347.

[26] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *Proc. 35th Int. Conf. Machine Learn.*, July 2018, pp. 1861–1870.

[27] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight Experience Replay," in *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[28] A. Raffin, "RL Baselines3 Zoo," https://github.com/DLR-RM/rl-baselines3-zoo, 2020, GitHub repository.

[29] E. Aasi, C. I. Vasile, M. Bahreinian, and C. Belta, "Classification of Time-Series Data Using Boosted Decision Trees," in *2022 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2022, pp. 1263–1268.