Applied Intelligence
https://doi.org/10.1007/s10489-021-03107-6

Timed failure propagation graph construction with supremal language guided Tree-LSTM and its application to interpretable fault diagnosis

Gang Chen[1]

Accepted: 13 December 2021

Abstract
Timed failure propagation graphs (TFPGs) perform fault diagnosis in a transparent way. However, accurate TFPGs depend on experts' knowledge or accurate model of the system, which is hard to obtain for complex systems. This paper presents a data-driven TFPG construction approach for fault diagnosis, which finds spectral-timed failure propagation graphs (sTFPG) directly from data. The sTFPG construction problem is transformed into a spectral-temporal logic inference problem and solved with a tree-structured long short-term memory (LSTM) network. Therefore, no expert or accurate model is needed to construct the TFPG. Moreover, the training process is guided and sped up by the supremacy property of the spectral-temporal logic, which focuses on the structure information of the signals and incorporates the physical meanings in the learning process. Experimental results on real rolling element bearing data sets illustrate that the performance of the proposed fault diagnosis method is comparable with state-of-the-art machine learning methods in fault diagnosis accuracy, and outperforms the logic-based method in computational efficiency. Additionally, fault diagnosis with TFPG can be understood by humans and reveal the fault mechanism.

Keywords Bearing fault diagnosis · Spectral temporal logic · Supremal language · Timed failure propagation graph · Tree-structured long short-term memory networks

1 Introduction

Timed Failure Prorogation Graphs (TFPGs) have attracted intensive studies among scholars and have been widely used for fault diagnosis of safety-critical systems [4, 7]. TFPG can be seen as a symbolic model of failure dynamics among a dynamic system, describing the occurrence of failure events and their propagations over time in the systems. They are powerful tools for fault diagnosis tasks for two reasons: Firstly, they are capable to model temporal logic relationships between basic faults and intermediate events. The temporal logic relationships among basic events rely on the operational modes of the systems, which also define the constraints of the event delays; Secondly, TFPGs can be easily explained with natural language and understood by human users, which enables the cooperation between fault diagnosis systems and human maintainers, thus reduces the cost of maintenance and enhances the system performance [22].

In the FAME project [5], the TFPGs were often manually derived from a given dynamic system based on experts' knowledge about the abstract representation of the system's behaviour under specific faulty conditions. In [4], a comprehensive approach was introduced to validate manually built TFPGs against the behaviours of the corresponding system. However, since modern industrial systems become more and more complex, it is impossible for experts to know the abstract representation and define the TFPGs precisely. Therefore, in line with considerable interest in TFPGs as tools for timed failure propagation modelling, many scholars have used them as the basics for fault diagnosis implementations and tried to generate the TFPGs automatically with the system behaviours [1, 7].

To reduce the work of experts, article [27] proposed an automated synthesis approach to construct TFPGs based on timed automata. Following a component topology, it

✉ Gang Chen
datagangchen@126.com

[1] Shien-Ming Wu School of Intelligent Engineering, South China University of Technology, Guangzhou 511442, China

Published online: 19 February 2022

discovers discrepancies between the output signals and the failures in input signals, traversing the zone graph of the automata, while it could not provide any formal characterization. As an improvement, article [4] presented an approach to support generic finite and infinite-state transition systems, defined discrepancies and failure modes as generic properties of the system state, and produce TFPGs with well-defined formal characteristics. The above two approaches assume the discrepancy nodes are given, which assumes the abstracted states of the system are given, thus not suitable for complex systems whose abstractions are hard to get. With the structural information, article [14] synthesized the TFPG with component models, but did not consider system dynamics. This approach is only possible with well-defined component models. To make good use of historical maintenance data, article [28] presented an approach for TFPG maturation, in which the TFPG can be improved with the historical maintenance data. However, the quality of data restricts the quality of the resulting model, and no guarantee can be obtained.

In this paper, we present a data-driven approach for constructing a kind of TFPG, called sTFPG, which takes as input a set of labelled time-series data of the system and does not require the knowledge of discrepancy nodes or structural model of the system. We first define a formal language, called spectral temporal logic (STL), which is a formal language to specify the temporal behaviours of the frequency events among the signals from the systems to be diagnosed and can be understood by human users. Moreover, we show that every sTFPG is equivalent to an STL formula, thus the sTFPG construction problem can be transformed into a *spectral temporal logic inference* problem. This work is extensive research about our previous work in [9], in which we construct the TFPG by searching through a set of predefined discrepant nodes. Since the complexity and expressiveness of the approach are restricted by the size of predefined discrepant nodes, this paper tries to use a machine learning method to address the complexity issue in our previous work. Moreover, the TFPG in this paper is defined over the spectral of the signals, thus it is expected to be more robust to noise and be suitable for vibration signals.

The sTFPG construct process in this paper is inspired by natural language generation. Both of them try to choose a set of words (atomic formulas in temporal logic) sequentially to construct a sentence (formula in temporal logic). The key issue that should be solved is to encode the meaning of the sentence in terms of hierarchical and nested structures of words [12, 13, 25, 25, 31]. Similar to the field of natural language generation, a key challenge for the sTFPG construct is how to incorporate semantic and syntax information in the learning process. In the field of interpretable fault diagnosis, semantic and syntax information are of great importance, since they include physical meanings of the diagnosis results and mechanism of the faults.

Existing data-driven approaches have many advantages over the knowledge-based methods (knowledge about the abstraction or model of the system) in TFPG construction, but they cannot be directly used to construct sTFPGs for three reasons. Firstly, without the knowledge of the system to be interpreted, the constructed sTFPGs may not show the physical properties of the system and will find irrelevant knowledge. For example, when we apply the data-driven approach in [9] to diagnose the faults of rolling element bearings, the sTFPGs may tell us that the energies of the resonant frequencies of inner race fault and outer race fault are different, i.e., inner race fault has larger impulse energies than outer race fault. However, based on the mechanism of rolling-element bearing fault, which tells us that the strikes of rollers on the fault surface will excite the fault signals and produce the resonant frequencies of structures between the bearing and transducers [23]. Namely, the difference between inner race fault signals and outer race is the resonant frequencies, but not the energies of the frequencies. Secondly, traditional machine learning approaches cannot determine the supremacy of the discovered sTFPGs and loss the structure information among signals. When we apply traditional machine learning methods to find an interpretation for the signals, such as the method in [10], it is possible to find two different formal descriptions that can interpret the properties of the signals, but it is hard to tell which one is better. Even though the method in [10] evaluates different logic-based methods with a numerical metric, called robustness degree, the numerical metric does not reveal the physical meanings of the signals. Thirdly, robustness degree is sensitive to the amplitude values of signals, and it ignores the structure information of the signals. However, in practice, structure information contains the physical process knowledge of a system and the mechanism of the faults. As mentioned in the above rolling element bearing fault diagnosis example, robustness degree can distinguish two kinds of faults based on different resonant energies of the signals, but it cannot distinguish different faults based on the temporal relationship among resonant energies of the signals.

In this paper, we address the aforementioned issues of machine learning methods by investigating the internal relationship between different spectral temporal logic formulas. Given two spectral temporal logic formulas, we not only care about how good the formulas can interpret the signals via robustness degree metric but also care about the structure difference among formulas via supremacy analysis. To our knowledge, it is the first attempt to consider the supremacy of the learned formal languages. Compared with existing

studies, the contributions of this paper include threefold as follows:

- We propose a novel two-domain formal logic that is suitable to describe the time-frequency properties of vibration signals, thus it can be used to describe fault features among vibration signals and diagnose faults for rotary machines.
- We develop a fully data-driven approach to construct sTFPGs without the need for expert knowledge, in which the learning process is guided by supremacy analysis of formal languages, avoiding losing structure information among signals. The proposed approach is the first attempt to unify the discrete logic events reasoning and continue variables learning problem in one framework.
- We demonstrate the performance of the proposed approach in fault diagnosis by experiments with real data sets collected from rolling element bearings. The experimental results indicate the proposed method can reveal the fault mechanism of rolling-element bearing and diagnose the faults in noisy environments with high accuracy.

This paper is organized as follows: Section 2 introduces the STL, the supremacy of STL language and formulates the sTFPG construction problem; Section 3 presents a supremal language guided tree-structured long short-term memory network (T-LSTM) based framework to solve the problem; Section 4 applies the proposed method to data sets collected from a rotary machine for fault diagnosis tasks; Finally, section 5 concludes the paper.

2 Problem statement

This section will define STL formally and give an example to show the properties of STL. Then we define the problem solved in this paper formally.

2.1 Spectral temporal logic

Spectral temporal logic is an extended version of signal temporal logic [20], and defined over discrete-frequency, discrete-time, continuous-valued time-frequency representation of a signal. Given a frequency domain $F = \{kf_0 | k = 0, 1, 2, \cdots\}$, a time domain $T = \{k\tau | k = 0, 1, 2, \cdots\}$, a discrete-frequency, discrete-time, continuous-valued time-frequency representation is a function $x \in \mathcal{F}(T, F, \mathbb{R}^n)$, where τ and f_0 are the sampling interval for time and frequency, respectively. $\mathcal{F}(A, B, C)$ denotes the set of all functions from $A \times B$ to C. Note that the time-frequency representation is multidimensional in the third direction, since the signal can be multidimensional. When the signal is multidimensional, its time-frequency representation is computed for each dimension independently. We use $x(t, f)$ to denote the value of time-frequency representation for signal x at time t and frequency f. Moreover, the time-frequency representation of signal x comes from the second temporal moment over time, called moment spectrogram. At every time t, we use a window to select a piece of signal, and calculate the second temporal moment to get the spectrum at time t based on the method in [10]. Then we slide the window to get the time-frequency representation of the signal. Formally, STL is defined as follows.

Definition 1 Spectral temporal logic is a time-frequency logic defined over signals' moment spectrogram. The syntax of an STL formula φ is defined recursively as:

$$\begin{aligned} \psi &:= \mu | \psi_1 \wedge \psi_2 | \Box_{[f_1,f_2]} \mu | \Diamond_{[f_1,f_2]} \mu, \\ \varphi &:= \psi | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | \varphi_1 \mathcal{U}_{[a,b]} \varphi_2 \end{aligned} \tag{1}$$

where $[f_1, f_2]$ and $[a, b]$ denote the frequency and time interval, respectively. μ is a predicate over the spectrogram, and $\mu := g(x(t, f)) \sim c$ with $g \in \mathcal{F}(\mathbb{R}^n, \mathbb{R})$ being a function that maps the spectral of the signal at time t and frequency f to a real value, $\sim \in \{\geq, <\}$, and $c \in \{kc_0 | k \in \mathbb{N}\}$ being a constant, where c_0 is the sampling interval. We use $\vee$ to denote logic ("or") and use $\wedge$ to denote logic ("and") operators, respectively. Moreover, the spectral operator $\Box$ denotes "always" operator, indicating the statement following the operator always true, and $\Diamond$ denotes "eventually" operator, indicating the statement to be true at leas once between f_1 and f_2 Hz, respectively. Finally, the temporal operator $\mathcal{U}$ denotes "Until" operator. For example, $\Box_{[f_1,f_2]} \mu$ means μ is always true between f_1 and f_2 Hz, and $\Diamond_{[f_1,f_2]} \mu$ means μ is eventually true between f_1 and f_2 Hz.

Note that STL is a two-domain logic, i.e., time domain and frequency domain, but it defines the time domain and frequency domain properties independently. All the spectral patterns are defined with ψ and all the temporal patterns are defined with φ. We also define a quantitative semantics for STL, called *robustness degree* $\rho : \mathcal{F}(T, F, \mathbb{R}^n) \times \Psi \to \mathbb{R}$, which maps an STL formula $\varphi \in \Psi$ and a spectrogram $x \in \mathcal{F}(T, F, \mathbb{R}^n)$ to a real value. The robustness degree measures the robustness of the formula with respect to perturbation, which is measured by how far the spectrogram of a signal x is away from satisfying STL formula φ at (t, f), denoted as $\rho(x, \varphi, t, f)$ and defined as:

$$
\begin{aligned}
\rho(x,(p(x)<\pi),t,f) &= \pi - g(x(t,f)) \\
\rho(x,(p(x)\geq\pi),t,f) &= g(x(t,f)) - \pi \\
\rho(x,\Box_{[f_1,f_2)}\mu,t,f) &= \min_{f'\in[f+f_1,f+f_2)} \rho(x,\mu,t,f') \\
\rho(x,\Diamond_{[f_1,f_2)}\mu,t,f) &= \max_{f'\in[f+f_1,f+f_2)} \rho(x,\varphi,t,f') \\
\rho(x,\varphi_1\wedge\varphi_2,t,f) &= \min\left(\rho(x,\varphi_1,t,f),\rho(x,\varphi_2,t,f)\right) \\
\rho(x,\varphi_1\vee\varphi_2,t,f) &= \max\left(\rho(x,\varphi_1,t,f),\rho(x,\varphi_2,t,f)\right) \\
\rho(x,\varphi_1\mathcal{U}_{[a,b]}\varphi_2,t,f) &= \max_{t'\in[t+a,t+b]}(\min(\rho(x,\varphi_2,t',f), \\
&\qquad \min_{t''\in[t,t']}\rho(x,\varphi_1,t'',f))
\end{aligned}
\tag{2}
$$

$\rho(x,\varphi,t,f)\geq 0$ means the behaviours of signal x can be described with φ at time t and frequency f correctly, denoted as $x[t,f]\vDash\varphi$, and $\rho(x,\varphi,t,f)<0$ means the behaviours of signal x violates φ at time t and frequency f, denoted as $x[t,f]\nvDash\varphi$. Therefore, to check whether a signal x satisfies a formula φ at (t,f), we only need to calculate the robustness degree $\rho(x,\varphi,t,f)$. To simplify the notations, we use $\rho(x,\varphi)$ to denote $\rho(x,\varphi,0,0)$ for short.

Next, we will introduce the relation between STL and sTFPG. The sTFPG used in this paper is a kind of TFPG defined specifically for capturing spectral temporal features. With small modifications from [9], the sTFPG used in this paper is a directed graph, in which the edges are from failure mode(s) to discrepancy events and can be defined as follows.

Definition 2 (sTFPG). An *sTFPG* is a tuple $G=\langle F,D,E,ET,DC,DP\rangle$, where (i) F is a set of failure mode nodes; (ii) D is a set of discrepancy nodes; (iii) $E\subseteq V\times V$ is a set of edges with $V=F\cup D$; (iv) $ET: E\rightarrow I$ maps an edge $e\in E$ to a time interval $[t_{\min}(e),t_{\max}(e)]\in I$ with $t_{\min}(e)$ and $t_{\max}(e)$ being the minimum and maximum propagation times on the edge e; (v) $DC: D\rightarrow\{AND,OR\}$ maps a discrepancy node $d\in D$ to its discrepancy type; and (vi) DP maps a discrepancy node $d\in D$ to an STL formula $\psi := \Box_{[f_1,f_2]}(l(x)<\pi_1\wedge l(x)\geq\pi_1)|\Diamond_{[f_1,f_2]}(l(x)<\pi_1\wedge l(x)\geq\pi_2)$ over a time-frequency representation x, where $\pi_1>\pi_2$, and $l(x)<\pi_2$ and $l(x)\geq\pi_2$ are the predicates defined in (1).

sTFPGs inherit all the desirable properties of TFPGs, but the semantics of an sTFPG are defined over time-frequency representation of signals, which makes sTFPGs be suitable for rotary machines fault diagnosis. We have the following proposition.

Proposition 1 *If a spectrogram x actives a node in an sTFGP G, denoted as $d\in D$, there is an STL formula φ_d associated to the node, such that $\rho(x,\varphi_d)\geq 0$.*

Lemma 1 in [9] shows that when a signal actives a node in sTFPG G, it must satisfy a signal temporal logic formula. Since STL proposed in this paper has similar semantics to signal temporal logic used in [9] when the nodes are atomic formulas about the spectral properties, it is easy to prove Proposition 1. Based on the Lemma 1 in [9], we know that whenever there exists a temporal operator $\mathcal{U}$ in an STL formula φ, it can be mapped to an edge in G. Further, if the directed graph starting from a failure mode is ended at one discrepancy node, we say the STL formula defined by the last node of the directed graph is the STL formula for the failure mode.

2.2 Illustration example

Example 1 Figure 1 shows one such sTFPG. The intervals over the edges denote the time intervals when a failure event will propagate from a node to the child node. An OR node will be satisfied when one of its parents' event has reached to the node within the time interval defined by the edge, while an AND node will be satisfied when all of its parents' events have reached to the node within the time interval defined by the edges. In this paper, the edges to an AND node have the same time interval for simplicity. The main difference between the sTFPG used in this paper and a traditional TFPG is that the discrepancy nodes of our sTFPG are defined by atomic STL formulas. For example, formula $\varphi_1=\Diamond_{[550,1250]}(x(t,f)\geq -51.0\wedge x(t,f)<-49.5)$ is attached to D1, which requires the spectrogram should be eventually greater than -51.0 dB and smaller than -49.5 dB between 550 and 1250 Hz at time t, formula $\varphi_2=\Box_{[350,900]}(x(t,f)\geq -50.5\wedge x(t,f)<-47.5)$ is for node D2, which requires the spectrogram should be always smaller than -47.5 dB and larger then -50.5 dB between 350 and 900 Hz at time t, formula $\varphi_3=\Box_{[1200,2000]}(x(t,f)<-47.0\wedge x(t,f)\geq -50.5)$ is for D3, which requires the spectrogram should be always greater than -50.5 dB and smaller than -47.0 dB between 1200 and 2000 Hz at time t, and formula $\varphi_4=\Diamond_{[1200,1800]}(x(t,f)\geq -51.5\wedge x(t,f)<-47.5)$ is for D4, which requires the spectrogram should be eventually greater than -51.5 dB and smaller than -47.5 dB between 1200 and 1800 Hz at time t, respectively. The graph shows that when failure FM happens, event D1 will be triggered within the next 1 second, i.e., φ_1 is satisfied within the next 1 second, then event D2 will be triggered within the next 0.1 seconds and event D3 will be triggered within the next 0.1 seconds, i.e., formula φ_3 will be satisfied. Event D4 will be triggered within 0.2 seconds after event D2 and D3 are triggered, i.e., formula φ_4 will be satisfied. The sTFPG can be mapped to a formula $\varphi=((\varphi_1\mathcal{U}_{[0.0,0.1]}\varphi_3)\wedge(\varphi_1\mathcal{U}_{[0,0.1]}\varphi_2))\mathcal{U}_{[0.0,0.2]}\varphi_4$, which is the satisfaction condition for node D4.

Figure 2 shows a signal's moment spectrogram that satisfies the sTFPG in Figure 1. The moment spectrogram shows

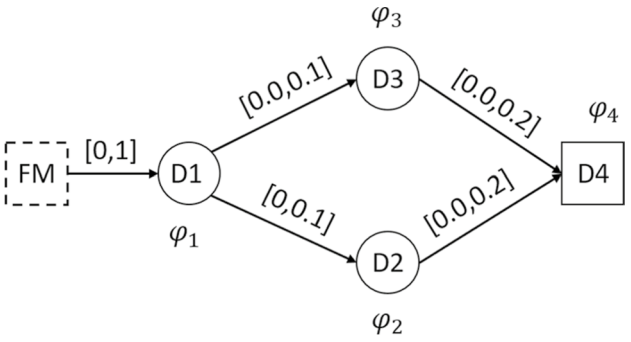

Fig. 1 An illustration of an sTFPG G. Circles are *OR* nodes. Dotted and solid boxes are failure mode node and *AND* node, respectively

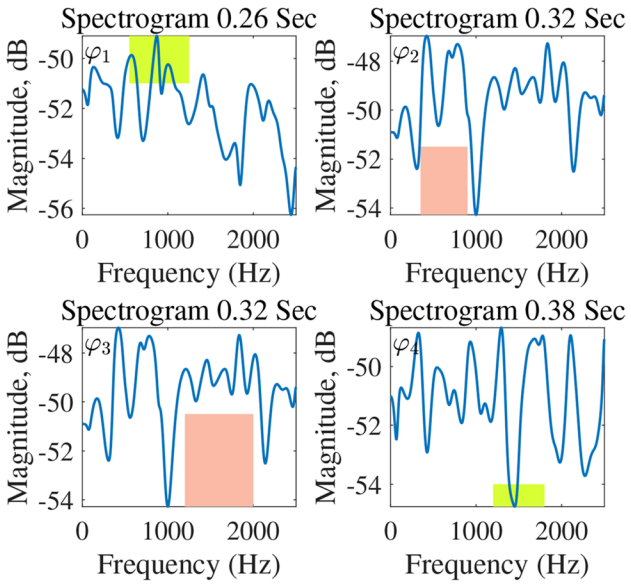

Fig. 2 Spectral of the signal at time 0.24 sec, 0.4 sec, 0.48 sec, and 0.58, respectively. The green regions are the spectrums should reach, and the pink regions are the spectrums should not avoid

φ_1 is satisfied at time 0.26 seconds (the moment spectrogram is larger than -51.0 dB and smaller than -49.5 dB at least once between 550 and 1250 Hz); φ_2 is satisfied at 0.32 seconds (the moment spectrogram is always larger than -50.5 dB and smaller than -47.5 dB between 350 and 900 Hz at 0.32 seconds). The time shift is 0.06, which is in the interval [0, 0.1]; φ_3 is satisfied at 0.32 seconds (the moment spectrogram is always larger -50.5 dB and smaller than -47.0 dB between 1200 and 2000 Hz), which happens in the interval [0.0, 0.1]; Similarity, we can see φ_4 is satisfied at time 0.38 seconds (the moment spectrogram is larger than -51.5 dB and smaller than -47.5 dB at least once between 1200 and 1800 Hz), which is within the next 0.0 to 0.2 seconds after φ_2 and φ_3 are satisfied. Therefore, the moment spectrogram satisfies formula φ.

2.3 Supremal language

In this subsection, we discuss the supremacy of STL formulas. Based on Proposition 1, any STL formula can be mapped to an sTFPG. Since the STL formulas are defined over the continuous-valued, discrete frequency and discrete time space, in order to get a symbolic representation of the time-frequency representation, we partition the state space of the signals' spectrums into intervals with equal length. As shown in Figure 3, the state space is partitioned and we assign a low case alphabet to each partition, which can be seen as an abstraction of the spectrum space [24]. Based on the partition, we can obtain an abstraction for each signal's spectrum. Given a spectrum of a signal at time t, denoted as $x(t, f)$, we can define a word $\omega = \alpha(x(t,f))$, which is the sequence of low case alphabet, where α is a map that maps a spectral value to an alphabet. Moreover, since the node of sTFPG is defined with formula $\psi := \Box_{[f_1,f_2]}(l(x) < \pi_1 \wedge l(x) \geq \pi_1)|\Diamond_{[f_1,f_2]}(l(x) < \pi_1 \wedge l(x) \geq \pi_1)$, in which the predicates $(l(x) < \pi_1 \wedge l(x) \geq \pi_1)$ define a rectangle among the spectral-amplitude plane. Similarity, we assign an upper case alphabet to each rectangle. As shown in Figure 3 (left), we assign alphabet A_3 to the rectangle defined by $\psi = (x(3,3) \geq 3 \wedge x(3,3) < 6)$. Obviously, the rectangle may contain many small partitions, and when ψ (A_3) is satisfied, $a \vee b \vee c$ is true at $x(3, 3)$. Namely, the value of the spectrum at $x(3, 3)$ should be a or b or c. Based on the semantic of STL, we can check that Figure 3 (middle) satisfies formula $\Box_{[3,7]}\psi$ $(\Box_{[3,7]}A_3)$ and Figure 3 (right) satisfies formula $\Diamond_{[3,7]}\psi$ $(\Diamond_{[3,7]}A_3)$.

In the time domain, we focus on the structure of the signals and ignore the precise temporal relationship, thus we ignore the temporal interval after "Until" operator. Given an STL formula φ, we use $\alpha(\varphi)$ to denote the abstracted formula, called $\alpha-$STL formula. Moreover, we denote the node formulas in sTFPG as a set of atomic proposition AP. Obviously, the syntax and semantics of $\alpha-$STL is the same with linear temporal logic (LTL) [8], which is defined based on a set of atomic propositions AP as follows.

$$\phi := \top | p \in AP | \phi_1 \vee \phi_2 | \phi_1 \wedge \phi_2 | \phi_1 \mathcal{U} \phi_2, \tag{3}$$

where ϕ, ϕ_1, and ϕ_2 are LTL formulas and $\top$ denote logic "true". The logic is defined over $\Sigma = 2^{AP}$. Since the LTL formulas defined by syntax in (3) can be transformed into Büchi automata [15], the abstracted $\alpha-$STL formulas can also be transformed into Büchi automaton. A Büchi automaton is a 5-tuple $\mathcal{A} = (Q, \Sigma, \delta, I, F)$, where Q denotes a finite set of states, Σ is a finite alphabet, and $\Sigma' = 2^{\Sigma}$; $\delta : Q \times \Sigma' \to Q$ to be a transition function, denoting the

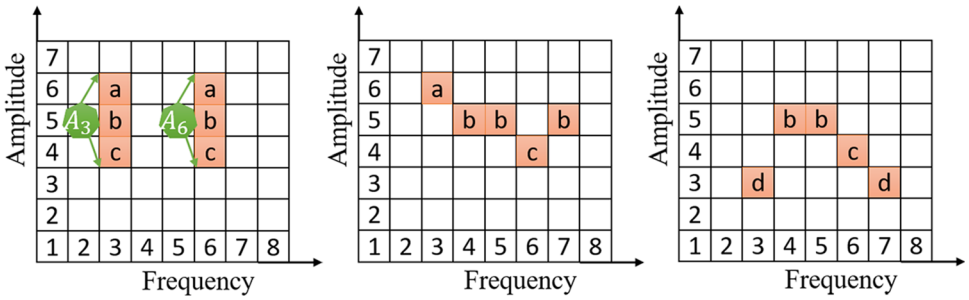

Fig. 3 Abstraction representation of a spectrum. (left) abstraction maps of the spectrum; (middle) spectrum that satisfies $\wedge_{i=3}^{7} A_i$ (w.r.t. $\square_{[3,7]}(a \wedge b \wedge c)$); (right) spectrum that satisfies $\vee_{i=3}^{7} A_i$ (w.r.t. $\Diamond_{[3,7]}(a \vee b \vee c)$)

state changing relations; $I \subseteq Q$ is a set of initial states, and $F = \{F_1, F_2, \cdots, F_m\}$, where $F_j \subseteq Q \times \Sigma' \times Q$ are sets of accepting transitions.

Example 1 (Continued.) In this example, we reconsider the sTFPG in Figure 1 and show how the TFPG can be represented by an STL formula. The formula associated to the sTFPG is $\varphi = ((\varphi_1 \mathcal{U}_{[0.1,0.3]} \varphi_3) \wedge (\varphi_1 \mathcal{U}_{[0,0.2]} \varphi_2)) \mathcal{U}_{[0.1,0.4]} \varphi_4$, where $\varphi_1, \varphi_2, \varphi_3$ and φ_4 are defined in Example 1. If we see $\varphi_1, \varphi_2, \varphi_3$ and φ_4 as atomic propositions, we can obtain an LTL formula $\phi = ((\varphi_1 \mathcal{U} \varphi_3) \wedge (\varphi_1 \mathcal{U} \varphi_2)) \mathcal{U} \varphi_4$. Here the atomic proposition $AP = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$.

In the above example, the atomic propositions are the atomic formulas, which are independent from the signals. To allow the Büchi automata to have more detail description capacity, in the rest of this paper, we expand the atomic formulas with symbols from the abstraction map α. For example, formula $\square_{[f_1,f_2]}(l(x) < \pi_1 \wedge l(x) \geq \pi_1)$ can be expanded into $\wedge_{i=f_1}^{f_2} A_i$, where A_i is the abstracted symbol for rectangle $(l(x) < \pi_1 \wedge l(x) \geq \pi_1)$ at frequency i. Similarly, formula $\Diamond_{[f_1,f_2]}(l(x) < \pi_1 \wedge l(x) \geq \pi_1)$ can be expanded into $\vee_{i=f_1}^{f_2} A_i$. Note that the expanded formulas are still LTL formulas and can be transformed into Büchi automata, but the transformed automata have larger atomic proposition sets.

Next, we consider the supremacy of sTFPGs, in which we investigate which sTFPG is the best one to interpret the signals. The underlying assumption for this investigation is that a better sTFPG will be robust to the variation of the signals' amplitude values, and captures the structure features of the signals, thus is more expressive. This assumption is reasonable, since the mechanism of fault is usually unchanged, while the signals from the system to be diagnosed are usually contaminated by noises or affected by the variation of operations. The noises and varying operations will change the amplitude values. Thus a more expressive sTFPGs will be closer to the fault mechanism. Given a set of sTFPGs, we say the associated STL formula for the most expressive sTFPG as the supremal formula, which is defined as follows.

Definition 3 (Supremal Formula) Given a set of STL formulas Φ, we say a formula is the supremal formula among Φ, denoted as $\mathcal{S}(\Phi)$ if the following condition holds:

$$\forall x, \forall \varphi \in \Phi, \text{ if } \alpha(x) \vDash \alpha(\varphi), \text{ then } \alpha(x) \vDash \alpha(\mathcal{S}(\Phi)). \tag{4}$$

where $\alpha(x)$ is the abstraction for signal x. Note that the supremal formula is defined over formula set Φ, but not defined over signals, which indicates that the supremacy of a formula is independent from the signals. This property is very important, since data-driven methods usually assume we have enough data, but in practice, we cannot guarantee the coverage of the data for the learning tasks. If the supremal formula is independent from the data, it will be more robust to the size and quality of data sets. A related concept about supremacy is partial order, in which for any two formulas φ_1, φ_2, we say φ_1 is a partial order of φ_2, denoted as $\varphi_1 \preceq \varphi_2$, if $\forall x$, $\alpha(x) \vDash \alpha(\varphi_1) \Longrightarrow \alpha(x) \vDash \alpha(\varphi_2)$.

Since each STL formula can be mapped to an LTL formula, the supremacy of formulas can be investigated in terms of formal language. We use Σ to denote a finite alphabet, i.e., a set of event, and Σ^+ is the set of all finite sequences of events (an event is an element of Σ) taken from Σ and we have $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$, where ϵ is the empty sequence. A sequence of events is also called a *string*, and $\sigma \in s$ means that event σ occurs at least once in s [29]. For any string $s \in \Sigma^*$, $\epsilon s = s\epsilon = s$. Given a Büchi automaton $\mathcal{A}$, the transition function can be extended to strings in Σ^* by letting $\delta(q, \epsilon) = q$ for all $q \in Q$, and $\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$ otherwise. For a state $q \in Q$ and a event $\sigma \in \Sigma$, the transition $\delta(q, \sigma)$ is eligible when the transition is defined, denoted as $\delta(q, \sigma)!$. The set of eligible strings among an automaton $\mathcal{A}$ is called the language of $\mathcal{A}$, denoted as $L(\mathcal{A}) = \{s \in \Sigma^* | \exists q_0 \in I, \delta(q_0, s)!\}$. The accepting language of the automaton $\mathcal{A}$ is denoted as $L_a(\mathcal{A}) = \{s \in \Sigma^* | \exists q_0 \in I, \delta(q_0, s)!$ and $\exists \sigma \in s, \sigma \in F\}$. With a little abuse of notation, here we use $L^\alpha(\varphi)$ to denote the language of an STL formula under abstraction mapping α and $L_a^\alpha(\varphi)$ the accepting language, respectively. Based on

the definition of supremal formula, we have the following lemma.

Lemma 1 *Given a set of STL formula* Φ, φ_S *is the supremal formula among* Φ *if* $\forall\varphi \in \Phi$, *we have* $L_a^\alpha(\varphi) \subseteq L_a^\alpha(\varphi_S)$.

The proof of Lemma 1 is obviously. The condition indicates that for every accepting string of $L_a^\alpha(\varphi)$, there exists the same string in $L_a^\alpha(\varphi_S)$, which means that for any signal x, when the associated string satisfies formula φ, i.e., $\alpha(x) \in L_a^\alpha(\varphi))$, it must satisfies φ_S, i.e., $\alpha(x) \in L_a^\alpha(\varphi_S)$. However, the condition for supremal formula is quit conservative, since for any two formulas φ_1, φ_2, it is likely that the two formulas only share some of the accepting language. Namely $L_a^\alpha(\varphi_1) \setminus L_a^\alpha(\varphi_2) \neq \emptyset$ and $L_a^\alpha(\varphi_2) \setminus L_a^\alpha(\varphi_1) \neq \emptyset$. Moreover, when two formulas' associated automata have different alphabet sets, it is unlikely that the two automata share any accepting language, which makes it hard to measure the supremacy level of the formulas. To deal with this situation, we introduce a map, called *natural projection* [32]. For two alphabets Σ_1 and Σ_2, which come from the associated Büchi automata for two STL formulas φ_1 and φ_2, the natural projection $P_{\Sigma_1\to\Sigma_2} : \Sigma_1^* \to \Sigma_2^*$ removes from traces $s \in \Sigma_1^*$ all events not in Σ_2^*, defined as

$$P_{\Sigma_1\to\Sigma_2}(s\sigma) = \begin{cases} P_{\Sigma_1\to\Sigma_2}(s)\sigma, & \text{if } \sigma \in \Sigma_2 \\ P_{\Sigma_1\to\Sigma_2}(s), & \text{otherwise .} \end{cases} \tag{5}$$

To simplify the notation, we use $P_{\varphi_1\to\varphi_2}$ to denote $P_{\Sigma_1\to\Sigma_2}$ for simplicity. Figure 4 shows a conceptual illustration of the natural projection of two formulas φ_1, φ_2, in which $P_{\varphi_1\to\varphi_2}(L_a^\alpha(\varphi_1))$ indicates the strings in $L_a^\alpha(\varphi_1)$ that can find their associated projected strings in $L_a^\alpha(\varphi_2)$. If all the strings in $L_a^\alpha(\varphi_1)$ can find their projected strings in $L_a^\alpha(\varphi_2)$, we say φ_2 has larger supremacy than φ_1. Then the metric for supremacy level of two formulas can be defined as

$$r(\varphi_1, \varphi_2) = \begin{cases} \frac{|P_{\varphi_1\to\varphi_2}(L_a^\alpha(\varphi_1)) \cap L_a^\alpha(\varphi_2)|}{|P_{\varphi_2\to\varphi_1}(L_a^\alpha(\varphi_2)) \cap L_a^\alpha(\varphi_1)|}, & \text{if } P_{\varphi_2\to\varphi_1}(L_a^\alpha(\varphi_2)) \neq \emptyset, \\ 0, \text{ otherwise ,} \end{cases} \tag{6}$$

where $|\cdot|$ denotes the cardinality of a set. Based on the definition of supremacy level, if $\varphi_1 \preceq \varphi_2$, then $r(\varphi_1, \varphi_2) \leq 1$ and for all $\varphi \in \Phi$, $r(\varphi, \mathcal{S}(\Phi)) \leq 1$.

2.4 Problem formulation

We wish to construct an sTFPG by finding an STL formula, which can be used to classify time-series data from a rotational machine. Moreover, the associated language defined by the STL formula is a supremal language among the formula space, which gives an expressive explanation about the failure propagation properties among the machine. Here we consider the case in which the sTFPG construction procedure can learn from historical data that has been labelled according to whether or not it represents a faulty behavior. More formally, we will solve Problem 1.

Problem 1 Let $X = X^+ \cup X^-$ be a labeled set of moment spectrogram obtained by calculating the second temporal moment over time to a set of time-series data from a rotational machine. Let $\mathcal{D}$ be a set of atomic formulas having the form of $\psi := \Box_{[f_1,f_2]}(l(x) < \pi_1 \wedge l(x) \geq \pi_1)$ and $\Diamond_{[f_1,f_2]}(l(x) < \pi_1 \wedge l(x) \geq \pi_1)$, and α is the abstraction map for STL formulas. Let $\mathcal{T}$ be a set of temporal operators having the form of $\mathcal{U}_{[a,b]}$. The goal is to choose N atomic formulas from $\mathcal{D}$ and connect these atomic formulas with Boolean operator $\wedge, \vee$ and temporal operator $\mathcal{U}_{[a,b]}$, such that

$$R(X, \varphi_N) = \min(\min_{x\in X^+}(\rho(x, \varphi_N)), \min_{x\in X^-}(\rho(x, \neg\varphi_N))) + \eta\frac{1}{|\Phi|}\sum_{\varphi\in\Phi} r(\varphi_N, \varphi), \tag{7}$$

is expected to be maximized, where $\rho(x, \varphi_N)$ and $\rho(x, \neg\varphi_N)$ are the robustness degrees of time-frequency representation x with respect to φ_N and its negation, respectively. The first term of $R(X, \varphi_N)$ is the robustness degree of φ_N with respect to X, and the second term is the average supremacy level among a set of formulas Φ, where Φ is a set of formulas among the search space and constructed dynamically during the construction procedure. η is a constant value, which balances the effect of the two terms.

This problem is modified from the supervised learning problem previously addressed in [20]. In [20], the formula is defined with signal temporal logic, and the searching process is guided by a predefined order, whose search

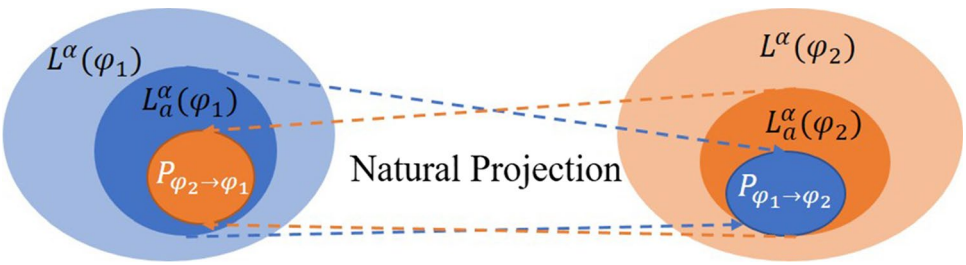

Fig. 4 A conceptual illustration of the natural projection of two formulas φ_1, φ_2

space grows exponentially with respect to the number of dimensions of the signal and the length of the formula. Moreover, the formulas learned in [20] cannot be mapped to an sTFPG. In this paper, we will find STL formulas with T-LSTM for fault diagnosis tasks and the searching procedure will consider the supremacy of the found STL formulas. Considering the definition of supremal language, we have the following lemma.

Lemma 2 (Existence) *Given $\mathcal{D}$ and $\mathcal{T}$ as described in Problem 1, there exists a set of formulas Φ, such that any STL formula φ constructed with $\mathcal{D}$ and $\mathcal{T}$, $\exists \varphi' \in \Phi, r(\varphi, \varphi') \leq 1$.*

The proof of Lemma 2 is simple, since we can increase the size of Φ to satisfy the condition. The issue is that it is hard to figure out the size of Φ, which is related to the size of $\mathcal{D}$ and the way how $\mathcal{D}$ is generated. In this paper, we set the size of Φ through experiments.

3 Spectral-timed failure propagation graph construction with T-LSTM

The goal of this paper is to construct an STL formula, such that the cost function in (7) is maximized. Figure 5 illustrates the overall framework proposed in this paper, which includes three components: Policy network, Parser network, and Evaluator. The policy network adopts a stochastic policy and samples an atomic formula from $\mathcal{D}$ at each step, which is a formula generation network. It keeps sampling until the number of atomic formulas reaches a limitation N. The parser takes the sequence of sampled atomic formulas as inputs and outputs an STL formula by constructing the parsing tree for the formula with a sequence of actions, denoted as P_φ. The Evaluator evaluates the performance of the generated formula. Since the reward can be computed once the final structure of the formula is available, the process can be naturally addressed by the policy gradient method [30].

3.1 Tree-LSTM for syntactic parsing

The parser component tries to select a sequence of operations from a predefined set $\{S, D, C, \mathcal{U}_{[a_1,b_1]}, \cdots, \mathcal{U}_{[a_i,b_i]}, \cdots\}$, where S denotes a shift operator, D means $\vee$ operator, C means $\wedge$ operator and $\mathcal{U}_{[a_1,b_1]}$ means $\mathcal{U}_{[a_1,b_1]}$ operator, to construct an STL formula with the sequence of atomic formulas chosen by the Policy network. As shown in Figure 6, a unique sequence of $\{S, D, C, \mathcal{U}_{[a_1,b_1]}, \cdots, \mathcal{U}_{[a_i,b_i]}, \cdots\}$ operations corresponds to a unique binary parse tree of the selected atomic formulas, which further leads to a unique STL formula. We note that for a sequence of atomic formulas of length N, there are exactly N shift (S) operations and $N-1$ CONJUNCTION/DISJUNCTION/UNTIL ($C/D/\mathcal{U}_{[a,b]}$) operations that are needed to construct the parse tree.

Our Parser follows the Stack-augmented Parser-Interpreter Neural Network (SPINN) [6], which is a shift-reduce parser that uses LSTM as its composition function. Given an input sequence of atomic formulas $\varepsilon = \{\omega_1, \omega_2, \cdots, \omega_N\}$, the parser tracks an index p starting from the leftmost formula ($p = 1$) and maintains a stack. To parse the sequence of formulas, parser chooses a sequence of operations $\hat{\varepsilon} = \{a_1, a_2, \cdots, a_{2N-1}\}$, where $a_t \in \{S, D, C, \mathcal{U}_{[a_1,b_1]}, \cdots, \mathcal{U}_{[a_i,b_i]}, \cdots\}$. When an S operation is chosen, the formula ω_i is pushed to the stack and the pointer will move to the next atomic formula (p_{++}) for next step of choosing; while when a $D/C/\mathcal{U}_{[a,b]}$ operation is chosen, the stack pops two elements out, and combines them into a single formula with $\vee/\wedge/\mathcal{U}$ operator, and pushes it back to the stack. Here we use T-LSTM as the parsing function, which maps a operation to a value, denoting the probability that the operation will be chosen. To select the operations, we parameterize each operation $a_t \in \{S, D, C, \mathcal{U}_{[a_1,b_1]}, \cdots, \mathcal{U}_{[a_i,b_i]}, \cdots\}$ with a policy network $P_\varphi(\cdot|\mathbf{s}_p, W)$, where $\mathbf{s}_p$ represents the current state at stack p and W is the parameter matrix of the network. During the parsing procedure, a two-layer feed-forward network is used to approximate the state vector estimation function, whose inputs are the hidden states of the

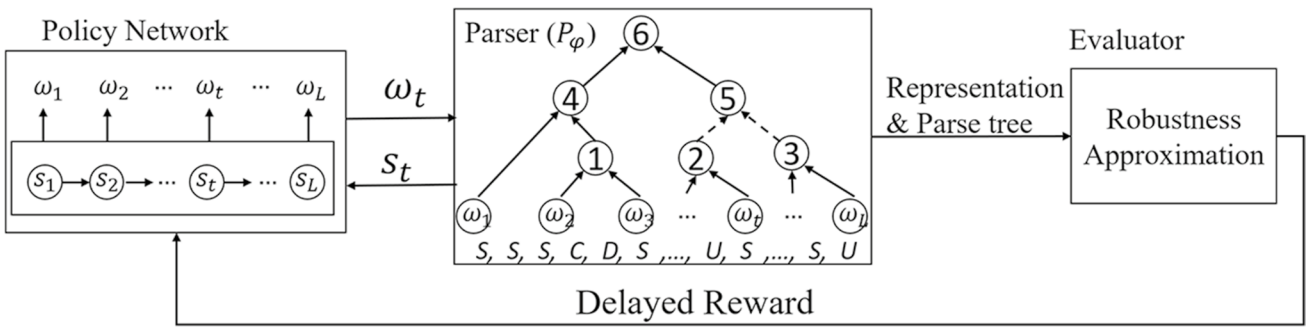

Fig. 5 Structure of the graph construction process. The policy network samples an atomic formula at each state. The Parser finds the syntax parsing tree and outputs the final formula φ for approximation task when the parsing tree is completed. The approximation performance, supremacy level and robustness degree obtained provide the reward to the networks

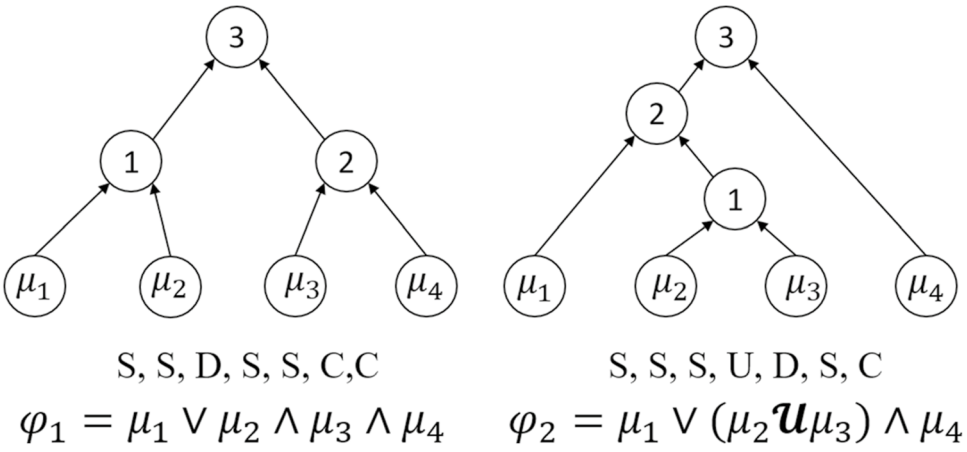

Fig. 6 Two examples of the trees and their corresponding operation sequences. There are 4 input atomic formulas (4 leaf nodes) for each example, so 7 actions are needed to construct a valid parsing tree. A SHIFT (S) operation introduces a leaf node, while a DISJUNCTION(D)/CONJUNCTION(C)/UNTIL(U) operation combines two previously nodes and introduces a non-leaf node. Obviously, different operation sequences lead to different tree structures and STL formula

top two elements of the stack h_i and h_j and the probability of operation a_p to be chosen is defined as:

$$\begin{aligned} \mathbf{s}_p &= \text{ReLU}(W^p[h_i, h_j, a_p] + b^p) \\ P_\varphi(a_p|\mathbf{s}_p;\Psi) &= \text{softmax}(W^2\mathbf{s}_p + b^2) \end{aligned} \tag{8}$$

where $\Psi = \{W^1, W^2, b^1, b^2\}$ are the parameters for the two-layer feed-forward network. Figure 7 (right) shows the encoding map for operators in $\mathcal{T}$. The first bit indicates the spectral operator used, i.e., 0 for "always" and 1 for "eventually", the second bit indicates the dimensional index of the signal, the third bit indicates the comparison operator, i.e., 0 for $\geq$ and 1 for $<$, the fourth bit indicates the scale value and the last two bits indicate the spectral bounds for the spectral operator. Figure 7 (left) shows the encoding map for atomic formula, where each atomic formula is represented with a six-dimensional vector. The first bit indicates the spectral operator used, i.e., 0 for "always" and 1 for "eventually", the second bit indicates the dimensional index of the signal, the third bit indicates the comparison operator, i.e., 0 for $\geq$ and 1 for $<$, the fourth bit indicates the scale value and the last two bits indicate the spectral bounds for the spectral operator. For example, formula $\square_{[1,2]}(x_1 \geq 0.1)$ can be encoded as vector [0, 1, 0, 0.1, 1, 2].

Evaluator: To evaluate the performance of the parser network, the last state vector $\mathbf{s}_N$ is fed to the evaluator network, which is a two-layer feed-forward network and the reward function is the combination of robustness degree, supremacy level and robustness degree approximation error, defined as:

$$J(\Omega) = -(W_s\mathbf{s}_N + b_s - \rho(X, \varphi_N))^2 + \lambda R(X, \varphi_N) \tag{9}$$

where Ω denotes all the parameters, $W_s\mathbf{s}_N + b_s$ is the approximation for the robustness degree, and λ is a factor that balances the importance of cost function and approximation error. The goal of the Parser network and Evaluator is to maximize the reward $J(\Omega)$. Note that the calculation of $J(\Omega)$ needs a set of formulas Φ. In this step, we initialize Φ with M STL formulas with length N, then during the temporal logic inference procedure, Φ is updated with the new generated formula φ_N by calculating the supremacy level $r(\varphi, \varphi_N)$ for all $\varphi \in \Phi$. If $r(\varphi, \varphi_N) > 1$ for all $\varphi \in \Phi$, then Φ is not updated; else Φ is updated by replacing the formula that has smallest supremacy level.

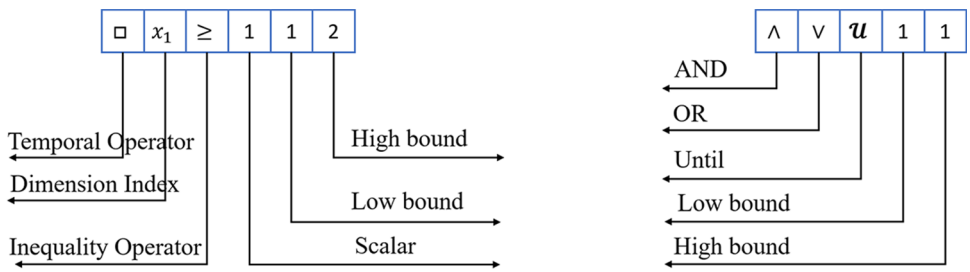

Fig. 7 Encoding maps for atomic formulas in $\mathcal{D}$ and operators in $\mathcal{T}$

3.2 Policy network for atomic formula generation

The Policy network adopts a stochastic policy $\pi(\omega_t|\mathbf{s}_t;\Theta)$ and uses a delayed reward to improve the policy. During the learning process, it samples an atomic formula with the probability at each state whose representation s_p is constructed by the hidden states in the Parser network. Let ω_t is the atomic formula at time t, the policy is defined as,

$$\begin{aligned}\hat{s}_t &= \mathrm{ReLU}(W^l[h_i, h_j, \omega_t] + b^p)\\ \pi(\omega_t|\mathbf{s}_t;\Theta) &= \mathrm{softmax}(W_p\hat{s}_t + b_p)\end{aligned} \tag{10}$$

where $\pi(\omega_t|\mathbf{s}_t;\Theta)$ denotes the probability of choosing ω_t, and $\Theta = \{W_i, b_p, W^i, b^p\}$ denotes the parameters of the Policy network. In the beginning, the hidden states h_i and h_j of the stack are initialized with zero vectors, and the state of the current formula representation $\mathbf{s}_p$ is calculated based on Eqn.(8), then the word at the begin is sampled based on Eqn.(10). Since there are N *shift* operations will be chosen by the Parser network if the length of the formula is N, the Policy network will sample an atomic formula from $\mathcal{D}$ whenever all the atomic formulas have been pushed to the stack, i.e., only the $\mathbf{s}_p$ at the step when p_{++} point to null will be used for the policy network. During the training process, we use the policy gradient method to optimize the parameters of the policy network, aiming to maximize the expected reward and the gradient is,

$$\bigtriangledown_\Theta = \sum_{t=1}^{N} R(X, \varphi_N)\bigtriangledown_\Theta \log\pi(\omega_t|\mathbf{s}_t;\Theta) \tag{11}$$

where $R(X, \varphi_N)$ is the reward defined in Eqn.(7) obtained when the formula is constructed completely. Compared with traditional machine learning-based methods that use classification performance as the reword function, e.g., the reward function in [3, 6, 11, 33], the reward function used in this paper includes the structure information of the formulas, which shapes the reward function by supremacy. Even though there exist many language machine learning methods that have incorporated the syntax information in the learning process, such as the methods in [21, 34, 35], these methods encode the syntax information into a vector representation, which losses the physical meaning of the syntax, thus cannot be used to interpret the fault diagnosis results.

3.3 Training process

In this section, we train the above networks using a policy gradient algorithm. To simplify the training process, here we train the three components jointly, which is shown in Algorithm 1. The training process includes three steps. Firstly, we pre-train the Parser network and Evaluator. Secondly, we pre-train the Policy network while keeping the parameters of the other two models fixed. At last, we jointly train all three components. Compared with other learning algorithms, our algorithm is guided by the language, which shapes the reward and speeds up the learning process.

Algorithm 1 Tree-LSTM Network Training Process.

Input: A set of labelled moment spectrogram $X = X^+ \cup X^-$, a set of atomic formulas $\mathcal{D}$ and a set of operators $\mathcal{T}$ and two positive integers N and M.
Output: The optimal STL formula φ_N.

```
Initialize Policy Network, Parser and Evaluator with random weights Ψ, Θ, a set of STL formulas Φ and Ω.
Pre-train the Parser networks and Evaluator while keeping the parameters of Policy network fixed by maximizing Eqn.(9);
Fix the parameters of the Parser and Evaluator networks, and pre-train the Policy network by Eqn.(11);
repeat
    for 6-steps do
        Sample a sequence of words with Policy network ω_1, ω_2, ··· , ω_N.
        for p in 1 : N do
            Compute s_p and a_p by Eqn.(8).
        Compute J(Ω) by Eqn.(9).
        Update Φ.
        Update Evaluator parameters Ω by Eqn.(9).
        Update Parser network parameters Ψ by Eqn.(8).
    for 6-steps do
        Generate a sequence of words with Policy network ω_1, ω_2, ··· , ω_N.
        for p in 1 : N do
            Compute s_p and a_p by Eqn.(8).
        Compute J(Ω) by Eqn.(9).
        Update Φ.
        Update Policy network parameters Θ by Eqn.(10).
until Convergence
```

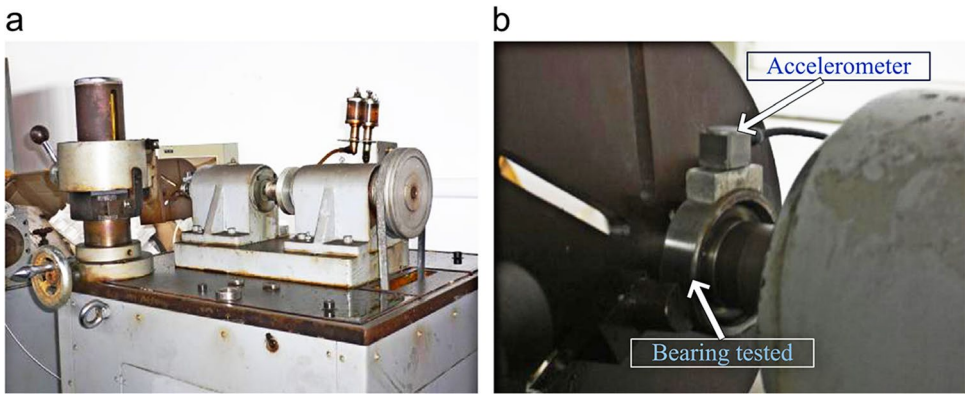

Fig. 8 **a** The test rig and (**b**) the location of the accelerometer for signal collection

4 Case studies

In this section, we will evaluate the performance of the proposed method for sTFPG construction and its application to fault diagnosis and interpretation. The signals are processed with Matlab and Python environments.

4.1 Fault diagnosis for fixed speed rolling element bearing

In this experiment, the rotational machine is a rolling-element bearing test-rig as shown in Figure 8, which has been used in [19]. To collect the faulty signals of rolling element bearings, the electrical-discharge machining method was used to introduce single pitting faults. The faults are added to the surface of the race or the rolling body of a series of rolling element bearings (one type fault for each). The signals are collected with the shaft speed being fixed and the sampling rate is 12 kHz.

During the experiment, we introduce three kinds of faults to the bearings, i.e., rolling element fault, inner race fault, and outer race fault. After all the data has been collected, 200 signal samples with length 10000 for each condition (sampled within 0.83 seconds) were used for demonstration (600 pieces in all). To obtain spectrograms of the signals, we calculate the second temporal moment over time with the Matlab embedded function for each piece of the signals. Therefore, we have 200 spectrograms samples for each bearing condition. Then we construct the labeled training and testing set. The positive training set for inner fault includes 100 samples from inner fault signals, and the negative set includes 100 samples from the other two conditions (50 samples for each). Then the training sets for the other two conditions are constructed accordingly. The positive testing set for inner fault includes the rest 100 samples from inner fault signals, and the negative testing set includes 100 samples from the other two faults (50 samples for each). We construct the training and testing sets for each fault independently and accordingly.

The training process is initialized by generating 1000 atomic formulas for $\mathcal{D}$ and 100 temporal operators with the form of $\mathcal{U}_{[a,b]}$ for $\mathcal{T}$. The spectrum partition size for formulas in $\mathcal{D}$ is 50 Hz, and the amplitude partition size for formulas in $\mathcal{D}$ is 0.5 dB. Since the temporal partition size is not relevant to computational complexity, here the minimum interval for the temporal domain is 0.05 seconds. The Policy and Parser produce an 18-dimensional vector for the state vector and we set the length of the formula with 4 words. We run

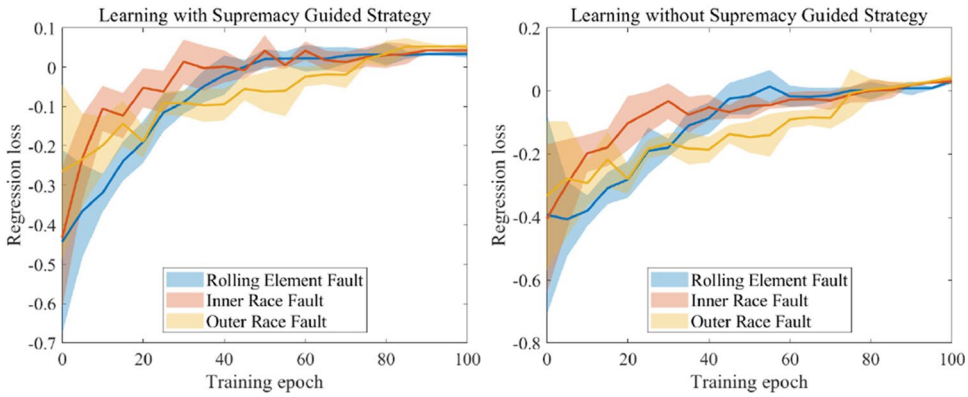

Fig. 9 Average robustness degree and its variance obtained for each training epoch among testing data sets for learning algorithm with supremacy guided strategy (left) and without supremacy guided strategy (right)

Table 1 sTFPG associated STL formulas for inner race, outer race and rolling element faults with real experiment data

Fault type	Interpretation
Inner Race	$\varphi_I = \Diamond_{[0,0.5]}((\varphi_1 \mathcal{U}_{[0,0.05]}\varphi_2)\mathcal{U}_{[0,0.1]}\varphi_3)\mathcal{U}_{[0,0.15]}\varphi_4$; where $\varphi_1 = \Box_{[1200,2200]}(x(t,f) < -65.0 \wedge x(t,f) \geq -67.5)$, $\varphi_2 = \Box_{[1200,2200]}(x(t,f) \geq -62.5 \wedge x(t,f) < -60.5)$, $\varphi_3 = \Box_{[1200,2200]}x(t,f) < -67.5 \wedge x(t,f) \geq -69.0))$, $\varphi_4 = \Box_{[2900,4400]}(x(t,f) \geq -63.5 \wedge x(t,f) < -62.5)$.
Outer Race	$\varphi_O = \Diamond_{[0,0.5]}((\varphi_1 \mathcal{U}_{[0,0.1]}\varphi_2) \wedge \varphi_3)\mathcal{U}_{[0,0.2]}\varphi_4))$; where $\varphi_1 = \Box_{[1000,1200]}(x(t,f) < -68.5 \wedge x(t,f) \geq -70.0)$, $\varphi_2 = \Box_{[1000,1200]}(x(t,f) \geq -63.0 \wedge x(t,f) < -61.5)$, $\varphi_3 = \Box_{[5000,5750]}(x(t,f) < -68.0 \wedge x(t,f) \geq -70.0))$, $\varphi_4 = \Diamond_{[5000,5750]}(x(t,f) \geq -63.5 \wedge x(t,f) < -60.5)$.
Rolling Element	$\varphi_R = \Diamond_{[0,0.5]}(((\varphi_1 \wedge \varphi_2)\mathcal{U}_{[0,0.15]}\varphi_3)\mathcal{U}_{[0,0.05]}\varphi_4)$; where $\varphi_1 = \Diamond_{[950,1550]}(x(t,f) < -68.5 \wedge x(t,f) \geq -69.5)$, $\varphi_2 = \Diamond_{[4000,5150]}(x(t,f) \geq -65.5 \wedge x(t,f) < -62.5))$, $\varphi_3 = \Box_{[2400,2600]}(x(t,f) \geq -65.0 \wedge x(t,f) < -62.5)$, $\varphi_4 = \Box_{[950,2800]}(x(t,f) < -69.0 \wedge x(t,f) \geq -70.0)$.

Table 2 The semantics for STL formulas for inner race, outer race and rolling element faults with real experiment data

Fault type	Interpretation	Semantics
Inner Race	$\varphi_I = \Diamond_{[0,0.5]}((\varphi_1 \mathcal{U}_{[0,0.05]}\varphi_2)\mathcal{U}_{[0,0.1]}\varphi_3)\mathcal{U}_{[0,0.15]}\varphi_4$;	((**LOW** then **HIGH**) then **LOW**) then **HIGH**
Outer Race	$\varphi_O = \Diamond_{[0,0.5]}((\varphi_1 \mathcal{U}_{[0,0.1]}\varphi_2) \wedge \varphi_3)\mathcal{U}_{[0,0.2]}\varphi_4))$;	((**LOW** then **HIGH**) and **LOW**) then **HIGH**
Rolling Element	$\varphi_R = \Diamond_{[0,0.5]}(((\varphi_1 \wedge \varphi_2)\mathcal{U}_{[0,0.15]}\varphi_3)\mathcal{U}_{[0,0.05]}\varphi_4)$;	((**LOW** and **HIGH**) then **HIGH**) then **LOW**

the training on a 64bit Linux computer with a 16-core CPU at 3.8 GHz, GeForce GTX 1070 GPU, and 64GB RAM.

Figure 9(left) shows the average robustness degrees and their variances of 10 trails obtained by Algorithm 1 for each epoch among the testing sets. These results indicate that our algorithm can reach positive robustness and classify the conditions of the bearings for three faults, indicating the obtained STL formula can diagnose the faults correctly among the testing set. In Figure 9(right), we also show the results of a modified Algorithm 1, in which we infer the formal language without the guide from the supremacy level. Namely, the modified reward function is defined as

$$R'(X, \varphi_N) = \min(\min_{x \in X^+}(\rho(x, \varphi_N)), \min_{x \in X^-}(\rho(x, \neg\varphi_N))). \quad (12)$$

where the supremacy level is not used. Figure 9 shows the algorithm guided by supremacy used about 40 epochs to reach a positive robustness degree, while the algorithm without supremacy guide needed about 80 epochs to reach a positive robustness degree. The comparison results indicate supremacy-guided algorithm can speed up the learning procedure. Moreover, the results also show the supremacy-guided algorithm can obtain formulas with larger robustness degrees, which indicates the obtained formulas will be more robust to noise.

Table 1 shows the learning results for the three faults with STL formulas. Note that we add $\Diamond_{[0,0.5]}$ before every generated formula, allowing the formulas to be satisfied at any time within 0.5 seconds, which emits the time alignment process for the signals. The semantics of the formulas are shown in Table 2, in which we use **LOW** and **HIGH** to indicate the low and high energy concentration, respectively. The semantics have omitted the temporal information for simplicity, which can be found in the captions in Figure 10, 11 and12. The sTFPGs with respect to the formulas

Table 3 Fault diagnosis results with sTFPGs for real bearing signals

Fault type	Robustness Degree		Error Rate	
-	Training	Testing	Training	Testing
Inner Race Fault	0.032	0.022	0.000	0.000
Outer Race Fault	0.023	-0.033	0.000	0.030
Rolling Element Fault	0.043	0.011	0.000	0.000

are shown in Figure 10-12, which shows that the fault patterns for the faults are different in the energy concentration. The sTFPGs have some nodes denoting the spectral are larger than some values within some frequency bands, showing that the occurring of fault conditions will lead to a larger value for the spectral at the time within these frequency bands (**HIGH**). Then the larger energy concentration bands will be followed by low energy concentration within some time intervals (**HIGH** then **LOW**). Moreover, different faults will have different concentration patterns in the frequency domain. Since the fault mechanism of rolling-element bearing is that the impulse energy comes from the strikes of rollers on the fault surface and excites the striking response of the bearing system, which means our sTFPGs, which denote a sequence of impulse energies, are in line with the fault mechanism. With the knowledge given by the sTFPG, maintainers know that avoiding energy concentration will reduce the risk of system failure, e.g., adding vibration isolation devices.

Table 3 shows the average faults diagnosis results for the experimental data with the learned STL formulas among 10 tails, in which the robustness and error rate is used as the metrics. Note that the formula that obtains positive robustness indicates it diagnose the fault correctly. On the contrary, the formula that obtains negative robustness may lead to misdiagnosis. For example, the outer race has negative

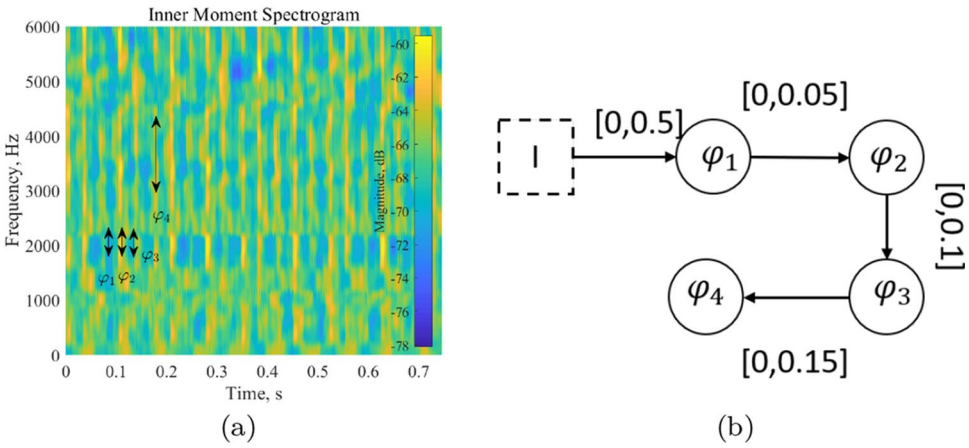

Fig. 10 **a** A moment spectrogram from an inner race fault bearing; (**b**) The sTFPG obtained by $\varphi_I = \Diamond_{[0,0.5]}((\varphi_1 \mathcal{U}_{[0,0.05]} \varphi_2) \mathcal{U}_{[0,0.1]} \varphi_3) \mathcal{U}_{[0,0.15]} \varphi_4$, which shows when inner fault happens (I in dash box), event φ_1 will happen within the next 0.5 seconds, then φ_2 will happen within the next 0.02 seconds, then φ_3 will happen within the next 0.1 seconds, then event φ_4 will happen within the next 0.15 seconds. The frequency events φ_1, φ_2, φ_3 and φ_4 are defined in Table 1 for φ_I

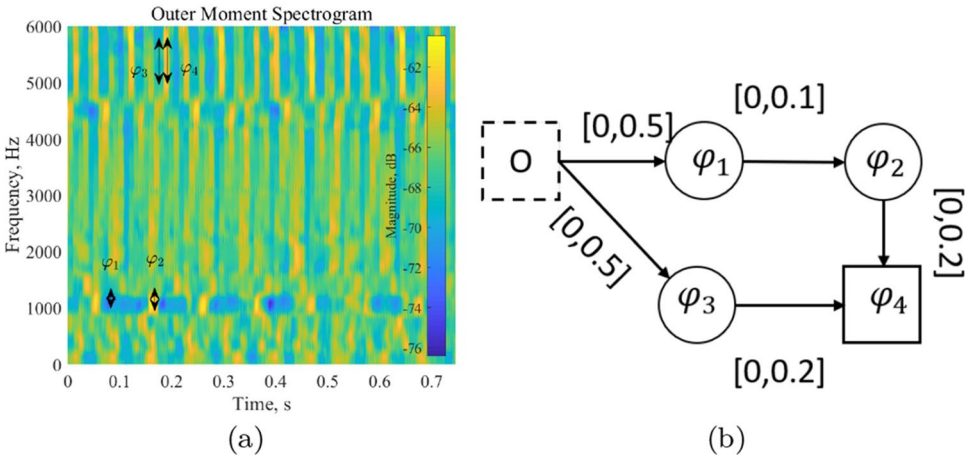

Fig. 11 **a** A moment spectrogram from an outer race fault bearing; (**b**) The sTFPG obtained by $\varphi_O = \Diamond_{[0,0.5]}((\varphi_1 \mathcal{U}_{[0,0.11]} \varphi_2) \wedge \varphi_3) \mathcal{U}_{[0,0.2]} \varphi_4))$, which shows when outer fault happens (O in dash box), event φ_1 and φ_3 will happen within the next 0.5 seconds. After φ_1 happens, then φ_2 will happen within the next 0.1 seconds. After φ_2 and φ_3 happen, then φ_4 will happen within the next 0.2 seconds. The frequency events φ_1, φ_2, φ_3 and φ_4 are defined in Table 1 for φ_O

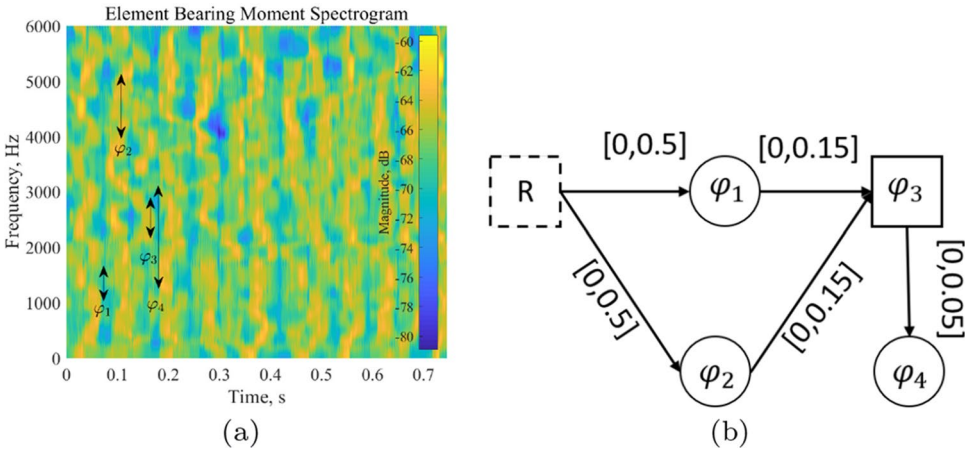

Fig. 12 **a** A moment spectrogram from a rolling element fault bearing; (**b**) The sTFPG obtained by $\varphi_R = \Diamond_{[0,0.5]}(((\varphi_1 \wedge \varphi_2) \mathcal{U}_{[0,0.15]} \varphi_3) \mathcal{U}_{[0,0.05]} \varphi_4)$, which shows when rolling element bearing fault happens (R in dash box), event φ_1 and φ_2 will happen within the next 0.5 seconds. After φ_1 and φ_2 happen, then φ_3 will happen within the next 0.15 seconds, then φ_4 will happen within the next 0.05 seconds. The frequency events φ_1, φ_2, φ_3 and φ_4 are defined in Table 1 for φ_R

robustness among testing data sets leading to misdiagnosis in Table 3, which may be caused by the noise or the differences of fault patterns or distribution between training and testing data. The results not only show our method can find the sTFPGs that are in line with the failure mechanisms of rolling element bearing, but also achieve fault diagnosis errors that are less than 5% among the test data sets. Moreover, the increase of robustness may not lead to a decrease in error rate, since we can only guarantee that positive robustness leads to a zero error rate. When we use a robustness definition that is smooth and differentiable, it is expected to have a relationship that a larger robustness degree will lead to a smaller error rate. In our future work, we will try to investigate the effect of robustness degree definition on

Table 4 Fault diagnosis error rate with the methods in [2, 9, 18] and sTFPGs for rolling element bearing fault diagnosis

Fault Type	Error Rate			
	Fisher+SVM [18]	GA+SVM [2]	rTFPG [9]	Proposed method
Inner Race	0.010	0.015	0.255	**0.000**
Outer Race	0.035	**0.025**	0.235	0.035
Rolling Element	0.020	0.010	0.115	**0.000**

the performance of the proposed method in fault diagnosis. Additionally, in order to have a better performance with respect to error rate, we can modify the evaluator's reward function $J(\Omega)$ and use the evaluator to approximate the error rate instead of the robustness degree.

In order to illustrate the performance of sTFPG in fault diagnosis, we conduct some comparison experiments in terms of fault diagnosis accuracy. In the first experiment, we compare our method with two feature-based fault diagnosis algorithms and one TFPG-based method. The comparison results are shown in Table 4, in which we compare the proposed sTFPG method with Fisher [18], Genetic Algorithm (GA) based SVM [2] and the rTFPG in [9]. The performances shown in the table are obtained among test data sets. The results show that our method outperforms the other methods in general, and only the diagnosis for outer race fault has worse performance than the GA+SVM method (0.035 vs 0.025). Moreover, the rTFPG has worse performance, and the reason is that the rTFPG is defined directly over the original signals, thus rTFPG is sensitive to noise.

In the second experiment, we further investigate the properties of our method by conducting a comparison experiment with the state-of-the-art formal logic-based methods, in which we compare the performance between our method and methods in [20] and [10] over the inner fault case. In this experiment, the lengths of the formulas are set to 4 for all approaches, and we check the performance of these methods at 20, 40, 60, and 80 minutes during the training process. The results are shown in Table 5, which shows that our method and the method in [20] can achieve zero miss-classification rate within 40 minutes, while the method in [10] cannot fully diagnose the faults correctly. [20] can obtain a good performance since it searches along with a predefined order for the optimal formulas. When the length of the formula is smaller, it can obtain a good performance. The method in [10] tries to use a Gaussian process to approximate the robustness degree function, which is a hard task due to the non-convex, non-differentiable and non-smooth properties of the robustness degree function. Therefore, [10] cannot reach a good performance within a limited time. Moreover, the formula learned in [20] and [10] cannot be equivalent to sTFPGs, which is a shortage for these two methods. Moreover, without the guide of supremal language, the formulas found by these two methods do not focus on capturing the structure information of the signals.

4.2 Discussions

The above case studies indicate that the supremacy of formal logic can guide the search procedure. Moreover, if the signals' structure information is important for the decision process, and the signals are contaminated by noise, then the supremacy-guided approach has advantages over other methods. However, the calculation of supremacy level is a computation costly process. Therefore, if the training data set is small, the benefit of the supremacy-based strategy may not cancel out the computation cost for supremacy level calculation.

Another concern of the proposed method is that it does not have advantages over other methods when the bearing vibration signals are collected with variational shaft rotation speed. Since the structure of the signals for different fault types are mixed with each other in this case, while the proposed method is good at finding the underlying structure of the data. Therefore, when we only consider fault diagnosis error rate, the proposed method does not have advantages over state-of-the-art methods for fault diagnosis under complex operating environments. However, when we use logic formulas for fault detection tasks, in which structure information is important to distinguish the fault signals from normal signals, the proposed method will have supremacy over other methods.

In order to improve the performance of the proposed method, extensions of the research can be focused on

Table 5 Comparison results between the proposed method and the other logic based methods for inner race fault

Method	Error Rate/ Robustness			
Time (min)	20	40	60	80
Proposed Method	0.280/-0.262	0.165/-0.087	0.050/-0.058	**0.000/0.022**
Frequency Temporal Logic [10]	0.215/-0.212	0.135/-0.113	0.050/-0.016	0.015/-0.013
Temporal Logic [20]	0.270/-0.243	0.150/-0.215	0.135/-0.019	0.115/0.023

noise-resistant sTFPG. Namely, de-noising techniques should be included when we define the formal language, e.g., the predicate function $p(x)$ in (2) is a low pass filter operator, such that the learned sTFPG can deal with noise. Since the TFPG provides the failure propagation information, an extension of its application can be focused on remaining useful life (RUL) estimation for rolling element bearings. When the sTFPG is used for RUL estimation, the robustness of the learned formula can be an indicator for the RUL, since it shows how much the collected signal satisfies the formula.

De-noising techniques have been intensively studied in the field of bearing fault diagnosis. sTFPG shows the failure prorogation pattern among the time-frequency representation, it provides an underlying structure of the signals. This structure information can be used to guide the de-noising procedure for bearing fault diagnosis. For example, when a de-noising auto-encoder shown in [26] is used to deal with the noise among rolling element bearing signals, the spectral temporal logic encoded by sTFGP can be used to guide the learning process, in which the robustness degree can be a part of the cost function during the training process. This combination of sTFGP and de-noising auto-encoder has two advantages. Firstly, the sTFGP restricts the searching space for parameters optimization among auto-encoder, thus it speed-ups the learning procedure. Secondly, the obtained signals processed by the auto-encoder will not lose the structure information, avoiding learning a result that has no physical meaning. Moreover, cross-domain knowledge can be encoded with sTFPG, e.g., knowledge from human experts and knowledge from deep learning results. The encoded knowledge can be used to guide the fault diagnosis process when we use deep learning techniques to diagnose the faults. Therefore, another extension of this work is cross-domain intelligent fault classification [16, 17].

In this paper, the proposed method is used to detect multiple bearing faults, thus it can be used to detect multiple faults. Moreover, the underlying mechanism of the proposed method is finding a spectral temporal logic formula to classify the faults. The found formula can be mapped to a timed failure propagation graph. Since spectral temporal logic can capture the time-frequency properties among signals, and the gear faults can be obtained from time-frequency analysis based on existing techniques, it is possible to use spectral temporal logic to detect the faults of gear. Moreover, gear fault signals and bearing fault signals have similar patterns, e.g., the faults will cause a sequence of impulses among the time-frequency representation of the signals. Thus the proposed method can be applied to detect gear faults.

5 Conclusions

The paper presents a novel approach to construct sTFPGs automatically from labelled data sets. The constructed sTFPGs can be used for fault diagnosis. which has been demonstrated with real data sets from rolling element bearings. Moreover, sTFGPs provide a transparent way for fault diagnosis. By transforming the sTFGP construct procedure into a spectral temporal logic inference problem, the reinforcement learning framework can be used to find the optimal STL formula. The advantages of the proposed method have been demonstrated with experiments with real data sets. The experiment results indicate the supremacy of the formal language can guide the learning process and provide robustness to noises for the learned sTFPGs. Given the popularity of formal language in safety-critical systems, we believe this paper provides a necessary foundation for many future system monitoring frameworks.

Declarations

Conflicts of interest The authors declare that they have no conict of interest.

References

1. Abdelwahed S, Karsai G, Mahadevan N, Ofsthun SC (2008) Practical implementation of diagnosis systems using timed failure propagation graph models. IEEE Transactions on Instrumentation and Measurement 58(2):240–247
2. Alba, E., Garcia-Nieto, J., Jourdan, L., Talbi, E.G.: Gene selection in cancer classification using pso/svm and ga/svm hybrid algorithms. In: 2007 IEEE Congress on Evolutionary Computation, pp. 284–290. IEEE (2007)
3. Bartocci E, Deshmukh J, Gigler F, Mateis C, Ničković D, Qin X (2020) Mining shape expressions from positive examples. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 39(11):3809–3820
4. Bittner, B., Bozzano, M., Cimatti, A.: Automated synthesis of timed failure propagation graphs. In: 25th International Joint Conference on Artificial Intelligence, pp. 972–978 (2016)
5. Bittner, B., Bozzano, M., Cimatti, A., De Ferluc, R., Gario, M., Guiotto, A., Yushtein, Y.: An integrated process for fdir design in aerospace. In: International Symposium on Model-Based Safety and Assessmemt, pp. 82–95. Springer (2014)
6. Bowman, S., Gupta, R., Gauthier, J., Manning, C.D., Rastogi, A., Potts, C.: A fast unified model for parsing and sentence understanding. In: 54th Annual Meeting of the Association for Computational Linguistics, pp. 1466–1477. Association for Computational Linguistics (2016)
7. Bozzano, M., Cimatti, A., Gario, M., Micheli, A.: SMT-based validation of timed failure propagation graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29, pp. 3724–3730 (2015)
8. Camacho, A., McIlraith, S.A.: Learning interpretable models expressed in linear temporal logic. In: Proceedings of the

International Conference on Automated Planning and Scheduling, vol. 29, pp. 621–630 (2019)
9. Chen G, Lin X, Kong Z (2021) Data-driven real-timed-failure-propagation-graph refinement for complex system fault diagnosis. IEEE Control Systems Letters 5(3):1049–1054
10. Chen G, Liu M, Chen J (2020) Frequency-temporal-logic-based bearing fault diagnosis and fault interpretation using bayesian optimization with bayesian neural networks. Mechanical Systems and Signal Processing 145:106951
11. Chen G, Liu M, Kong Z (2020) Temporal-logic-based semantic fault diagnosis with time-series data from industrial internet of things. IEEE Transactions on Industrial Electronics. https://doi.org/10.1109/TIE.2020.2984976
12. Chen G, Wei P, Jiang H, Liu M (2020) Formal language generation for fault diagnosis with spectral logic via adversarial training. IEEE Transactions on Industrial Informatics. https://doi.org/10.1109/TII.2020.3040743
13. Chomsky, N., Lightfoot, D.W.: Syntactic structures. Walter de Gruyter (2002)
14. Dubey, A., Karsai, G., Mahadevan, N.: Fault-adaptivity in hard real-time component-based software systems. In: Software Engineering for Self-adaptive Systems II, pp. 294–323. Springer (2013)
15. Fritz, C.: Constructing büchi automata from linear temporal logic using simulation relations for alternating büchi automata. In: International Conference on Implementation and Application of Automata, pp. 35–48. Springer (2003)
16. Hu C, He S, Wang Y (2021) A classification method to detect faults in a rotating machinery based on kernelled support tensor machine and multilinear principal component analysis. Applied Intelligence 51(4):2609–2621
17. Hu C, Wang Y, Gu J (2020) Cross-domain intelligent fault classification of bearings based on tensor-aligned invariant subspace learning and two-dimensional convolutional neural networks. Knowledge-Based Systems 209:106214
18. Jian, Z., Li, X.B., SHI, X.z., Wei, W., WU, B.b.: Predicting pillar stability for underground mine using fisher discriminant analysis and SVM methods. Transactions of Nonferrous Metals Society of China **21**(12), 2734–2743 (2011)
19. Jiang H, Chen J, Dong G, Liu T, Chen G (2015) Study on hankel matrix-based svd and its application in rolling element bearing fault diagnosis. Mechanical systems and signal processing 52:338–359
20. Kong Z, Jones A, Belta C (2016) Temporal logics for learning and detection of anomalous behavior. IEEE Transactions on Automatic Control 62(3):1210–1222
21. Kumar A, Ahuja K, Vadapalli R, Talukdar P (2020) Syntax-guided controlled generation of paraphrases. Transactions of the Association for Computational Linguistics 8:330–345
22. Lei Y, Jia F, Lin J, Xing S, Ding SX (2016) An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data. IEEE Transactions on Industrial Electronics 63(5):3137–3147
23. Li Y, Liang X, Zuo MJ (2017) Diagonal slice spectrum assisted optimal scale morphological filter for rolling element bearing fault diagnosis. Mechanical Systems and Signal Processing 85:146–161
24. Long, Z., Calin, G., Majumdar, R., Meyer, R.: Language-theoretic abstraction refinement. In: International Conference on Fundamental Approaches to Software Engineering, pp. 362–376. Springer (2012)
25. Ma, M., Huang, L., Zhou, B., Xiang, B.: Dependency-based convolutional neural networks for sentence embedding. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pp. 174–179 (2015)
26. Meng Z, Zhan X, Li J, Pan Z (2018) An enhancement denoising autoencoder for rolling bearing fault diagnosis. Measurement 130:448–454
27. Priesterjahn, C., Heinzemann, C., Schäfer, W.: From timed automata to timed failure propagation graphs. In: 16th IEEE International Symposium on Object/component/service-oriented Real-time Distributed Computing, pp. 1–8. IEEE (2013)
28. Strasser, S., Sheppard, J.: Diagnostic alarm sequence maturation in timed failure propagation graphs. In: 2011 IEEE AUTOTESTCON, pp. 158–165. IEEE (2011)
29. Su R, Van Schuppen JH, Rooda JE (2011) The synthesis of time optimal supervisors by using heaps-of-pieces. IEEE Transactions on Automatic Control 57(1):105–118
30. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems, pp. 1057–1063 (2000)
31. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 1556–1566 (2015)
32. Ware S, Su R (2016) Time optimal synthesis based upon sequential abstraction and its application to cluster tools. IEEE Transactions on Automation Science and Engineering 14(2):772–784
33. Wen, T.H., Gašic, M., Mrkšic, N., Rojas-Barahona, L.M., Su, P.H., Vandyke, D., Young, S.: Toward multi-domain language generation using recurrent neural networks. In: NIPS Workshop on Machine Learning for Spoken Language Understanding and Interaction. Citeseer (2015)
34. Zhang, Z., Wu, Y., Zhao, H., Li, Z., Zhang, S., Zhou, X., Zhou, X.: Semantics-aware bert for language understanding. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 9628–9635 (2020)
35. Zhang, Z., Wu, Y., Zhou, J., Duan, S., Zhao, H., Wang, R.: Sg-net: Syntax-guided machine reading comprehension. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 9636–9643 (2020)