Formal Language Generation for Fault Diagnosis with Spectral Logic via Adversarial Training

Gang Chen, *Member, IEEE*, Peng Wei, *Member, IEEE*, Huiming Jiang, and Mei Liu

***Abstract*—Fault diagnosis with formal languages can be performed in an interpretable way. However, traditional formal languages cannot deal with noisy environments. Additionally, finding the optimal formal language for fault diagnosis is still a challenge due to the sparse reward issue. This paper presents a novel method to find formal languages, written with signal spectral logic (SSL), to describe the fault behaviours among frequency domain for fault diagnosis. The formal language defined by SSL is robust to noise, acts as the fault diagnoser, and provides interpretabilities for human operators. Moreover, the fault diagnoser construction procedure has been formulated as a language generation process and an adversarial training technique is used to find the optimal formal language and avoid sparse reward issue existing in language generation problems. Some experiments with real rolling element bearing data and simulated signals demonstrate that our method is able to find formal languages to diagnose faults efficiently and accurately under noisy environments.**

***Index Terms*—Adversarial training, fault diagnosis, language generation, long short-term memory network, spectral logic.**

I. Introduction

Modern industrial systems grow increasingly complex and large-scale, and because of that, the fault diagnosis becomes more critical than the past [1]–[3] as the systems' reliabilities and availabilities suffer when faults accumulate. The goal of a fault-diagnosis system is to rapidly detect and determine the root causes of faults based on the obtained information, such as sensor data and the system model. Real-time fault detection and diagnosis have many benefits, such as preventing catastrophic failures, enhancing safety and reliability, positive economic and environmental impacts. Moreover, the interpretabilities of fault-diagnosis process is significantly important for timely responses to the faults occurring in the system [4]. For example, for the task of bearing condition monitoring for a wind turbine in [5], the monitoring system was able to detect deterioration of bearings and found that it's the high temperature causing the faults. Thus human experts can then deduce that the high temperature was due to insufficient or inefficient lubricant properties with his domain knowledge.

Manuscript received Month xx, 2xxx; revised Month xx, xxxx; accepted Month x, xxxx. This work was supported in part by the National Natural Science Foundation of China under Grant 52005335 and Grant 62003235 (Corresponding author: Mei Liu).

Gang Chen and Mei Liu are with the Department of Automation, School of Electrical and Information Engineering, Tianjin University, Tianjin, 200072 (e-mail:megangchen@gmail.com, liumeimei@tju.edu.cn).

Peng Wei is with the Department of Mechanical and Aerospace Engineering, University of California, Davis, Davis, CA, 95616 (e-mail: penwei@ucdavis.edu).

Huiming Jiang is with the School of Mechanical Engineering, University of Shanghai for Science and Technology, Shanghai, 200093ds (e-mail: hmjiang@usst.edu.cn).

In this work, we define a formula language, called signal spectral logic (SSL), to describe the fault events among a system for fault diagnosis with real time-series data. SSL is a formal language to specify frequency domain behaviours of the systems to be diagnosed, which is inspired by signal temporal logic (STL). SSL is defined over spectral kurtosis, which is robust to noise [6], and provides interpretability for the fault diagnosis results. It gives an explanation about the diagnosis process with spectral events occurring among the system. For instance, SSL can explain the fault diagnosis process for a rolling element bearing as "the out race of the bearing is faulty because whenever there exists a frequency resonance mode, whose amplitude is larger than 0.3, there exists another frequency resonance mode, whose amplitude is larger than 0.3 within the next 150 Hz." Additionally, we present a language generation framework with an adversarial training technique to generate the optimal formal languages for fault diagnosis, which borrows the idea of language generation in natural language processing field to avoid the sparse reward issue during formal language construction process. Moreover, using the adversarial training technique will increase the robustness of the learning results with respect to noise and improve the fault diagnosis accuracy [7], [8].

A. Related Works

Recently, there has been increasing interests in applying temporal logic, a kind of formal language for fault detection and diagnosis and have obtained good performance [9], [10]. Fault diagnosis with temporal logic belongs to data-driven approaches, but the diagnosis procedure is interpretable, thus avoids the interpretability issues of traditional data-driven methods. Similar to natural language, formal languages defined by temporal logic formulas (can be seen as sentences) describe temporal patterns between events in a form close to human's way of reasoning. Therefore, many temporal logic-based formal languages have been applied to monitoring tasks, such as first-order temporal logic [11], metric temporal logic [12], linear temporal logic [13] and signal temporal logic (STL) [14]. Among these methods, STL is defined over the continuous state, makes it suitable for describing continuous-state systems. The basic idea of monitoring with STL is learning an STL formula to describe the normal behaviour of the system. The status of the system is checked by passing the signal of the system to the learned formula. When the signal satisfies the formula, we say the system is normal. Otherwise, a fault alarm is triggered. However, since real systems usually work in a noisy environment, and STL is brittle to noise, applying STL to systems with noise is a challenge.

Another challenge of applying formal language to fault diagnosis is to find the optimal formula (sentence). There are mainly two categories of techniques to learn a formula defined by a formal language. The first category assumes the structure of the formula was given, usually defined by experts, but with unknown parameters [15]. Under this assumption, the problem is transformed into an optimization problem. The goal is to find the optimal parameters, such that the generated formula could classify the time-series data for monitoring tasks. However, this assumption is too conservative for many real complex systems, since it is hard for experts to define the formula structures. Therefore, the second category relaxes the assumption and attempts to infer a discriminating formula without specifying its structure [9]. The investigation of this scenario is similar to natural language generation: Given a set of atom formulas (words in natural language), the goal is to choose a set of words in a right order, such that the words can construct a formula (sentence in natural language) that classifies the time-series data correctly. Many approaches have been proposed to generate STL formulas for monitoring tasks. For example, the method in [16] tried all combinations of basic formulas according to a predefined order and selected the best one. Nevertheless, this method suffered from a combinatorial explosion issue. To reduce the computational complexity, the author in [9] formulated the formula generation problem as a Markov decision process and solved it with a reinforcement learning (RL) algorithm. A main drawback of learning based methods is that the reward is only available when the entire formula is generated. For example, when we need to generate a sentence with length of N, we do not receive any reward when we generate the first $N-1$ words until we compete the sentence. In the other words, the language generation problem has a sparse reward, and this issue becomes extremely serious when we need to generate a long formula (will be discussed in detail in Section II-B). Also, the scalar reward received when the formula has been completed is non-informative as it does not necessarily preserve the picture about the intermediate syntactic structure and semantics of the formula that is being generated for the learning algorithm.

B. Contributions and Advantages

This paper proposes a novel formal language, namely SSL, to define the properties of signals in frequency domain for fault diagnosis. Compared with STL, SSL describes the spectral properties of the signals instead of temporal properties. Since many systems' faults signals are contaminated by noises and can only be detected in frequency domain, the proposed formula language is suitable for these systems. To apply SSL to fault diagnosis scenarios, we focus on generating SSL formulas to discriminate different faulty scenarios with time-series data via adversarial training, in which we allow the discriminator to provide richer information about the current incomplete formula to the generator, thus the learning algorithm can get reward at every step of extending the current formula, not only at the last step, thus avoid the sparse reward issue. In summary, the major contributions of this paper are as follows:

1) *Signal spectral logic for fault diagnosis*: We propose a novel formal language, called SSL, which is suitable to describe the spectral properties of time-series data and diagnose the fault for rotational machines. The proposed language provides interpretation for the fault diagnosis results, and is robust to noisy environments.
2) *Formal language generation framework*: We propose an adversarial training framework to generate SSL formulas (sentences) for fault diagnosis, which addresses the sparse reward issue of formal language generation problems. Some experimental results demonstrate our method outperforms state-of-the-art methods for generating long sentences in terms of efficiency and accuracy.

This paper is organized as follows. Section II defines the signal spectral logic and formulates the problem solved in this paper. Section III introduces the adversarial training framework and the solution to the problem. Section IV applies the proposed method to real data sets and shows the comparison results, and section V draws the conclusions.

II. Problem Formulation

A. Signal Spectral Logic

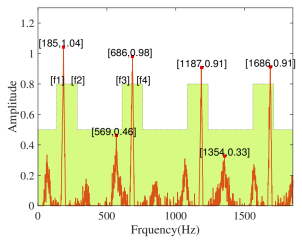
Signal spectral logic is inspired by signal temporal logic [9], which is defined over *continuous-frequency, continuous-valued spectral* of a signal. Given two sets A and B, $\mathcal{F}(A, B)$ denotes the set of all functions from A to B. Given a frequency domain $\mathbb{R}^+ := [0, \infty)$, the continuous-frequency, continuous-valued spectral is defined as a function $x \in \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n)$. When the signal is multi-dimensional, its spectral is calculated for each dimension independently using spectral kurtosis [6]. Therefore, the spectral of a multi-dimensional signal is also multi-dimensional. In the rest of this paper, we use $x(f)$ to denote the value of spectral for signal x at frequency f.

Definition 1. Signal spectral logic (SSL) is a spectral logic defined over a signal's spectral. Its syntax is defined recursively as:

$$\varphi ::= \mu | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | \Box_{[a,b]} \varphi | \Diamond_{[a,b]} \varphi, \tag{1}$$

where a and b are non-negative finite real numbers. μ is a predicate over a signal, which can be defined as $l(x(f)) \bowtie \pi$ with $l \in \mathcal{F}(\mathbb{R}^n, \mathbb{R})$ being a function, $\bowtie \in \{<, \geq\}$, and $\pi \in \mathbb{R}$ being a constant. The Boolean operators $\vee$ and $\wedge$ are disjunction ("or") and conjunction ("and"), respectively. The spectral operators $\Diamond$ and $\Box$ stand for "eventually" and "always", respectively.

The above syntaxes define how to use basic words, e.g., $\mu, \varphi_1, \varphi_2$, to construct a sentence (or formula). With these syntaxes, SSL is a powerful language to express the spectrum pattern of signals. The exists of spectral operators $\Box$ (always) and $\Diamond$ (eventually) make SSL suitable for depicting the periodical properties of a spectrum. For instance, the spectrum of a faulty rolling element bearing signal contains a cyclic impulse energy. Fig. 1(left) gives an example of such spectrum, and the periodical impulse energy, has the pattern written in English "whenever the spectrum has a energy density smaller than 0.5 for 325 Hz, the energy density for the spectrum will be bigger than 0.8 in the next 15 Hz." If we denote $x(f)$ to be the spectrum of the signal at frequent f, the spectral pattern can be easily described by an SSL formula $\varphi := \Box\Diamond_{[0,475]}(\Box_{[0,325]}(x(f) < 0.5) \rightarrow \Diamond_{[0,15]}(x(f) > 0.8))$, where $\Box$ without any frequency bound means a frequency bound $[0, \infty]$. SSL is interpretable and can be understood by human users. In the formula, $\Box\Diamond_{[0,475]}$, read as always eventually within 475 Hz, denotes that after the pattern occurs, then within 475 Hz, the pattern will occur again. $\Box_{[0,325]}(x(f) < 0.5)$ denotes the amplitude of the spectrum will be always smaller than 0.5 for 325 Hz, $\rightarrow$ denotes implication relationship. $\Diamond_{[0,15]}(x(f) > 0.8)$ denotes the energy density of spectrum $x(f)$ can be over 0.8 at least once within next 15 Hz. As shown in Fig. 1(left), at frequency f_2, the spectrum is smaller than 0.5 until frequency f_3, and $f_3 - f_2 = 325$ Hz. After keeping the value being smaller than 0.5 for 325 Hz, the spectrum is bigger than 0.8 between frequency f_3 to f_4, and $f_4 - f_3 = 15$ Hz.



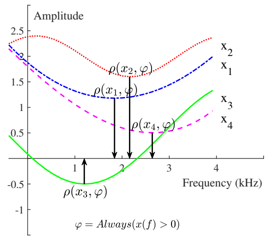
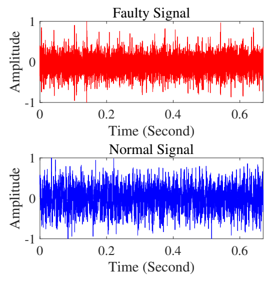


Fig. 1: (left) A signal with frequency pattern described with SSL formula $\varphi := \Box\Diamond_{[0,475]}(\Box_{[0,325]}(x(f) < 0.5) \rightarrow \Diamond_{[0,15]}(x(f) > 0.8))$; (right) Robustness degrees for different signals with formula $\phi = \Box(x(f) > 0)$ indicated by black directed line.

SSL is incorporated with a quantitative semantics called *robustness degree* $\rho : \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n) \times \Psi \rightarrow \mathbb{R}$, which maps a spectral $x \in \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n)$ and an SSL formula $\varphi \in \Psi$ to a real value. $\rho(x, \varphi)$ indicates how far the spectral of a signal x at frequency f is away from satisfying an SSL formula φ. It is defined as [9]:

$$\begin{aligned}
\rho(x,(l(x)<\pi),f) &= \pi - l(x(f))\\
\rho(x,(l(x)\geq\pi),f) &= l(x(f)) - \pi\\
\rho(x,\varphi_1\wedge\varphi_2,f) &= \min\big(\rho(x,\varphi_1,f),\rho(x,\varphi_2,f)\big)\\
\rho(x,\varphi_1\vee\varphi_2,f) &= \max\big(\rho(x,\varphi_1,f),\rho(x,\varphi_2,f)\big)\\
\rho(x,\Box_{[a,b)}\varphi,f) &= \min_{f'\in[f+a,f+b)}\rho(x,\varphi,f')\\
\rho(x,\Diamond_{[a,b)}\varphi,f) &= \max_{f'\in[f+a,f+b)}\rho(x,\varphi,f').
\end{aligned}$$

The robustness is sound, meaning that $\rho(x, \varphi, f) \geq 0$ implies that signal x satisfies φ at frequency f, and denoted as $x[f] \models \varphi$, whereas $\rho(x, \varphi, f) < 0$ implies that signal x violates φ at frequency f, and denoted as $x[f] \not\models \varphi$. In the rest of this paper, we denote the robustness of specification φ at frequency 0 with respect to signal x by $\rho(x, \varphi)$ for short. Fig. 1(right) shows four spectrums and their robustnesses with respect to SSL formula $\phi = \Box(x(f) > 0)$, where the length of a directed line indicates the magnitude of the robustness, up direction indicates negative robustness and down direction indicates positive robustness. Fig. 1(right) shows spectrum x_1, x_2 and x_4 have positive robustnesses and satisfy the formula, while spectrum x_3 has negative robustness and violates the formula. Moreover, spectrum x_2 has the largest robustness, thus the satisfaction of x_2 is more robust to noise, i.e., it is harder for disturbances to change the satisfaction state.



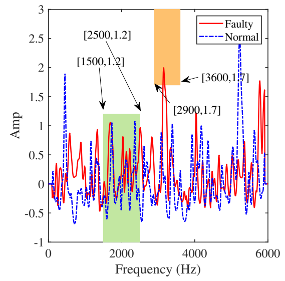


Fig. 2: left: The waveforms of two signals collected from a normal and a faulty rolling element bearing; right: The spectral kurtosis of the two signals' wavelet package transform based decomposed components.

B. Motivation Example

SSL formulas are quite expressive and interpretable. The following example will show the signals can be classified with spectrum patterns, but it is hard to be classified with temporal patterns. The spectrum patterns are written with SSL formulas, such that a human user can interpret SSL formulas easily. Moreover, we will show when learning an SSL formula with reinforcement learning algorithm as presented in [9], the sparse reward issue will occur.

Example 1. For instance, we can use an SSL formula

$$\varphi_n = \Box_{[1500,2500]}(e_s^1 < 1.2) \wedge \Diamond_{[2900,3600]}(e_s^2 \geq 1.7)$$

to distinguish the normal and faulty signals shown in Fig. 2 (left), where e_s^i in the formula is the spectral kurtosis for ith dimension of the signal s. φ_n can be dissected as follows:

$$\begin{aligned}
&\varphi_n := \varphi_1 \wedge \varphi_2,\\
&\varphi_1 := \Box_{[1500,2500]}(e_s^1 < 1.2), \varphi_2 := \Diamond_{[2900,3600]}(e_s^2 \geq 1.7).
\end{aligned}$$

φ_n can be interpreted with natural language and read as "properties φ_1 and φ_2 must both be true at the same frequency f to satisfy φ_n"; φ_1 can be read as "e_s^1 must always be smaller than 1.2 within the next 1500 and 2500 Hz"; and φ_2 can be read as "e_s^2 must be no smaller than 1.7 at least once within the next 2900 and 3600 Hz". As shown in Fig.2 (right), the green region indicates the requirement of φ_1, where the trajectory should always stay in that region. The yellow region indicates the requirement of φ_2, where the trajectory should occur in that region at least once. We can see that the faulty (red) signal satisfies the formula φ_n whereas the normal (blue) signal violates φ_n. Moreover, the spectrums of the signals are smooth, thus robust to noise. While the waveforms are contaminated by noise and hard to be distinguished with traditional formal languages defined over time domain using temporal patterns.

In this example, we assume φ_n is learned with a reinforcement learning algorithm. During the learning process, the algorithm should choose atom formulas (words) from a set of atom formulas (dictionary). The dictionary has two classes of words. The first one includes words constructed with a spectral operator and a predicate, e.g., $\Box_{[1,20]}(x(f) > 3.4)$, and the second one includes words constructed with a Boolean operator, a spectral operator and a predicate, e.g., $\wedge\Box_{[1,20]}(x(f) > 3.4)$. To construct an SSL sentence, the algorithm should choose one word from the first class for the initial word and choose other words from the second class to extend the formula. For example, to construct φ_n, the first word is φ_1 and the second word is $\wedge\varphi_2$. In order to choose the optimal words sequentially, reinforcement learning algorithm can be applied [9]. However, to construct a sentence, assuming the sentence has N words, the algorithm should take N actions to choose the words, while it does not receive any reward until the last step when a sentence has been completed. Namely, the algorithm comes up with sparse reward issue. This issue becomes serious when the length of the sentence is long. In the rest of this paper, we will try to address this issue.

C. Problem Formulation

In this paper, we are given a set of signals from the system to be diagnosed, and they are labelled with positive examples and negative examples, i.e., $X = X^+ \cup X^-$. The goal is to generate a formula φ, such that all positive examples satisfy the formula, whereas all negative examples violate the formula. Note that here we consider two conditions of the system (i.e., normal and faulty), and the system can have more than two fault types, which, in this case, one SSL formula will be generated for each fault type. To better relate the formula language with natural language, we use *word* to denote the atomic formula (formulas with a spectral operator and a predicate), *sentence* to denote the generated formula, and *vocabulary* to denote the set of all atomic formulas.

In this paper, the words have the forms of $\Box_{[a,b]}\mu$, $\Diamond_{[a,b]}\mu$, $\Box_{[a_1,b_1]}\Diamond_{[a_2,b_2]}\mu$ and $\Diamond_{[a_1,b_1]}\Box_{[a_2,b_2]}\mu$, where a, b, a_1, b_1, a_2 and b_2 are frequency parameters. Obviously, SSL is expressive since the parameters can be adjusted according to the signals.

During the language generation procedure, we can assume that there is an agent choosing words from vocabulary $\mathcal{V}$ and Boolean operators that connect the words. In order to reduce the agent's actions and complexity, we embed the words in $\mathcal{V}$ with Boolean operators in advance. The embedded vocabulary is a tube $\hat{\mathcal{V}} =< \hat{V}_0, \hat{V} >$, where $\hat{V}_0$ is the set of words from the original vocabulary $\mathcal{V}$, and $\hat{V}$ is the set of Boolean operators embedded words that have Boolean operators embedded which have the forms of $\wedge\Box_{[a,b]}\mu$, $\wedge\Diamond_{[a,b]}\mu$, $\wedge\Box_{[a_1,b_1]}\Diamond_{[a_2,b_2]}\mu$, $\wedge\Diamond_{[a_1,b_1]}\Box_{[a_2,b_2]}\mu$, $\vee\Box_{[a,b]}\mu$, $\vee\Diamond_{[a,b]}\mu$, $\vee\Box_{[a_1,b_1]}\Diamond_{[a_2,b_2]}\mu$ and $\vee\Diamond_{[a_1,b_1]}\Box_{[a_2,b_2]}\mu$. Therefore, the agent first chooses a word from $\hat{V}_0$, then chooses a sequence of words from $\hat{V}$ to construct a sentence, without the need to choose Boolean operators. The problem solved in this paper can be defined formally as:

Problem 1. (*Formal Language Generation*) Given a vocabulary $\hat{\mathcal{V}}$, a positive integer T and two sets of signals, X^+ and X^- , find an optimal policy $\pi(\cdot)$, which decides a sequence of words $(\varphi_0, \varphi_1, \cdots, \varphi_T)$, where $\varphi_0 \in \hat{V}_0$ and $\varphi_t \in \hat{V}$ for $1 \leq t \leq T$, to construct a sentence s_T, such that

$$\rho(X, s_T) = \min(\min_{x\in X^+}(\rho(x, s_T)), \min_{x\in X^-}(\rho(x, \neg s_T))) \quad (2)$$

is expected to be maximized, where $X = X^+ \cup X^-$. $\rho(x, s_T)$ and $\rho(x, \neg s_T)$ denote the robustness degrees of x with respect to s_T and its negation, respectively.

We solve a language generation problem in this paper. The obtained sentence s_T can be regarded as an interpretable classifier which has two roles. The first role is a classifier, and the second role is a decision explanator. It classifies the conditions of the system and gives explanations to the decision-making process with its semantics. Since we use labelled data to train a model, it is a supervised learning problem previously addressed in [17] with a reinforcement learning algorithm. However, in the previous work, the learning efficiency to generate long sentence is low due to the sparse rewards. In this paper, we address this issue with adversarial training by allowing the discriminator to provide richer information about the current incomplete formula to the generator, and the information will guide the language generation process. The following section introduces the adversarial training framework in detail.

III. Spectral Logic Formula Generation via Adversarial Training

A. Word to Vector

Before we generate the formal language, we need to map words and sentences to vectors and matrices, such that the adversarial training can be applied. Similar to the natural language process, the word should be first encoded into a vector. In this paper, we encode the atomic formula to a nine-dimensional vector (Note that this vector is used to define the word, but not the vector representation of the word used in the learning process, in which the representation is learned automatically). As illustrated in Fig. 3, Boolean operator embedded word in $\hat{\mathcal{V}}$ is encoded to a nine-dimensional vector. The first dimension decides the Boolean connector; the second dimension decides the type of temporal operators; the third dimension decides the signal's dimension index; the fourth dimension decides the type of inequality sign; the fifth dimension decides the scalar for the predicate; the sixth and seventh dimensions decide the frequency interval for the first spectral operator; the eighth and ninth dimensions decide the frequency interval for the second spectral operator. If there is only one spectral operator, the last two dimensions will be zeros. To map the words in vocabulary $\hat{\mathcal{V}}$ to vectors, we design the encoding table as shown in Table I. With the encoding table, every word in $\hat{\mathcal{V}}$ can be mapped to a ninth-dimensional vector uniquely. With the encoding of different words, the sentence can be mapped to a matrix, which is the stack of the words. Example 2 shows how to map words and sentences to vectors and matrices.

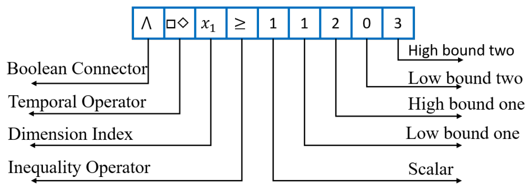


Fig. 3: Example encoding of a Boolean operator embedded word $\varphi = \wedge\Box_{[1,2]}\Diamond_{[0,3]}(x_1 \geq 1)$ to a nine-dimensional vector $[1,3,1,1,1,1,2,0,3]$.

TABLE I: Encoding table for words in $\hat{\mathcal{V}}$

Category	Symbol	Value	Category	Symbol	Value
Boolean	$\wedge$	1	Boolean	$\vee$	2
Boolean	none	0	Temporal	$\Box$	1
Temporal	$\Diamond$	2	Temporal	$\Box\Diamond$	3
Temporal	$\Diamond\Box$	4	Inequality	$\geq$	1
Inequality	$<$	2	Scalar	π	π
Lower Bound	a_1	a_1	Upper Bound	b_1	b_1
Lower Bound	a_2	a_2	Upper Bound	b_2	b_2
Dimension	i	i			

Example 2. Consider a sentence $\varphi_n = \varphi_1 \wedge \varphi_2$ with two words, $\varphi_1 = \Box_{[1500,2500]}(e_s^1 < 1.2)$ and $\varphi_2 = \wedge\Diamond_{[2900,3600]}(e_s^2 \geq 1.7)$. Based on the encoding table in Table I, the words can be mapped to vectors, denoted as $v_{\varphi_1} = [0,1,1,2,1.2,1500,2500,0,0]$ and $v_{\varphi_2} = [1,2,2,1,1.7,2900,3600,0,0]$, respectively. The sentence can be mapped to a matrix $M_{\varphi_n} = [v_{\varphi_1}, v_{\varphi_2}]^T$.

B. Adversarial Training Framework

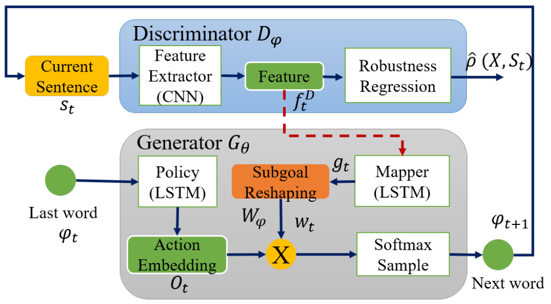


Fig. 4: An overview of the formal language generation framework. While the generator is responsible to generate the next word, the discriminator adversarially judges the generated sentence once it is complete. The major novelty lies in that, unlike the conventional method, during the sentence generation process, the discriminator reveals its internal state (feature f_t^D) in order to guide the generator more informatively, and tries to approximate the true robustness degree of the sentence.

We formulate the formal language generation problem as a sequential decision-making process [18]. To solve the problem, we apply the adversarial training technique to generate formulas for fault diagnosis. As illustrated in Fig.4, the overall framework is a *generative adversarial network* (GAN), which has two functions called *Generator* and *Discriminator*. The *Generator* generates sentences for fault diagnosis, while the *Discriminator* evaluates the performance of the generated sentences. In Fig.4, the θ-parametrized generator has a hierarchical structure, which consists of a high-level *Mapper* module and a low-level *Policy* module. The Mapper and Policy modules are comprised of long short-term memory networks (LSTM) [19] and serve as mediators. In summary, the generator G_θ, which corresponds to a stochastic policy, maps the current sentence s_t and last word to a distribution over the whole vocabulary, i.e., $G_\theta(\cdot|s_t)$, from which the action φ_{t+1}, i.e. the next word to select, is sampled. The discriminator, D_φ, consists of a feature extractor which is a convolutional neural network (CNN), and a robustness regression module which approximates the robustness degree of the sentence with respect to signals in X. At each step, the generator receives high-level feature representation from the discriminator D_φ, i.e., the feature comes from the CNN, and uses it to set up the guiding goal for the Policy module with the help from the Mapper.

During the generation process, the Policy module encodes the current generated word with an LSTM, and combines the output with the feature sub-goal generated from the Mapper. The Mapper takes feature f_t^D as the input and passes it through another LSTM. Then we take the final action at current state by choosing a word from $\hat{V}$ through a Softmax function. In such a way, the guiding signals from D_φ is available to the generator G_θ in terms of a goal embedding vector during the generation process and teaches G_θ how to get improved. To realize the information sharing between the discriminator and generator, we find out that a hierarchical RL architecture is an effective way to incorporate such shared information into the generation procedure of G_θ. We describe the detailed generator model in the following subsection.

1) Generator: The generator G have two LSTM networks. For the rest of this paper, we use θ_p to denote all parameters in the Policy LSTM and use θ_m to denote all parameters in the Mapper LSTM. Then the two LSTMs can be defined with functions $\mathcal{M}(f_t^D, h_{t-1}^M, \theta_m)$ and $\mathcal{P}(\varphi_t, h_{t-1}^P, \theta_p)$, respectively. During the generation process, the Policy and Mapper modules both start with all-zero hidden states, denoted as h_0^P and h_0^M. At each step, the Mapper receives feature vector f_t^D from the discriminator D_φ, which is further combined with the current hidden state h_{t-1}^M to produce the sub-goal vector g_t, such as

$$\hat{g}_t, h_t^M = \mathcal{M}(f_t^D, h_{t-1}^M, \theta_m), g_t = \hat{g}_t/||\hat{g}_t||. \tag{3}$$

To incorporate the output from Mapper with the output from the Policy, a linear transformation is applied to reshape the summation over previous C goals with a matrix W_φ, and the result is an embedding vector w_t, where $w_t = W_\varphi(\sum_{i=1}^{C}(g_{t-i})$.

The Policy module takes the current word φ_t as the input and outputs a matrix O_t, which is further combined with w_t by matrix product to determine the final action space distribution under current state s_t through a softmax function,

$$\begin{aligned} O_t, h_t^P &= \mathcal{P}(\varphi_t, h_{t-1}^P, \theta_p), \\ G_\theta(\cdot|s_t) &= softmax(O_t \cdot w_t/\alpha), \end{aligned} \tag{4}$$

where α is a parameter to control the generation entropy.

1551-3203 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Nanyang Technological University. Downloaded on November 28,2020 at 02:02:37 UTC from IEEE Xplore. Restrictions apply.

2) Discriminator: The discriminator measures the performance of the generated sentence, which tries to approximate the true robustness degree of the sentence based on Eqn. (2). The CNN with 8 convolutional layers is used to extract an 18 dimensional feature vector f_t^D, which is further fed to a feed-forward network to approximate the real robustness degree. We denote the network as

$$f_t^D = \mathcal{F}_\varphi(s_t, W_f), r_t = D_\varphi = \mathcal{R}(f_t^D, W_D), \qquad (5)$$

where f_t^D is the feature vector, r_t is the estimated robustness, W_f is the parameter matrix for the CNN, W_D is the parameter matrix for the discriminator and s_t is the current sentence. With s_t, the real robustness degree can be calculated with Eqn.(2) with respect to the training dataset, since the sentence is also an SSL formula. In summary, the discriminator tries to extract the optimal features, which can be used to estimate the real robustness degree and guide the training process in the following section.

C. Training Process

We train the above procedure by policy gradient algorithm, since the functions $\mathcal{D}_\varphi(s_t, W_d)$, $\mathcal{M}(f_t^D, h_{t-1}^M, \theta_m)$ and $\mathcal{P}(\varphi_t, h_{t-1}^P, \theta_p)$ are fully differentiable. The training process is shown in Algorithm 1. The algorithm first pre-trains the generator and discriminator, then the generator G_θ and discriminator D_φ are alternatively trained until convergence. Line 13 trains the discriminator to minimize the estimation error for robustness. The structure of the discriminator is classical and the training process is omitted here. To simplify the training process for the generator, here we follow the method in [20] and train the Policy and Mapper modules separately. In the generator, the Policy and Mapper LSTMs are alternatively trained while fixing the other. Next, we will introduce how to train the Policy network and Mapper network, respectively.

Line 11 in Algorithm 1 trains the Mapper and fixes the policy network, such that the Mapper will predict advantageous directions in the discriminative feature space and the Policy will obtain good performance by following the directions. The gradient of the Mapper is defined as,

$$\nabla_{\theta_m} = -Q(s_t, g_t)\nabla_{\theta_m} d(f_{t+R}^D - f_t^D, g_t(\theta_m)), \qquad (6)$$

where $Q(s_t, g_t) = \mathbb{E}[r_t]$ is the expected robustness degree obtained under the current policy, which can be approximately estimated via Monte Carlo search [21]. d is the distance measure between two vectors, Here we use the cosine similarity to measure the distance between the change of feature representation after $C-$step transitions and the goal vector $g_t(\theta_m)$ produced by the Mapper as in Eqn. (3). Intuitively, the training process tries to find the parameters θ_m, such that the goal vector will match the transition in the feature space and achieves high robustness for the generated formula.

Line 12 trains the Policy network to maximize the reward using the REINFORCE algorithm as is shown in [21],

$$\begin{aligned}\nabla_{\theta_p} &= \mathbb{E}_{s_{t-1}\sim G}[\textstyle\sum_{\varphi_t} r_t^I \mathcal{P}(\varphi_t|s_t, \theta_p)] \\ &= \mathbb{E}_{s_{t-1}\sim G, \varphi_t\sim\mathcal{P}(\varphi_t|s_{t-1})}[r_t^I \nabla_{\theta_p}\log\mathcal{P}(\varphi_t|s_{t-1}, \theta_p)],\end{aligned} \qquad (7)$$

where r_t^I is the intrinsic reward for the policy and

$$r_t^I = \frac{1}{C}\sum_{i=1}^{C} d(f_t - f_{t-i}, g_{t-i}). \qquad (8)$$

Then the gradient of the policy parameters, ∇_{θ_p} can be approximated by sampling the state s_{t-1} and the action φ_t taken by the Policy network.

Note that the proposed framework do not have sparse reward issue, since the expected robustness degree r_t and intrinsic reward r_t^I can be obtained at every step t, not only the last step when the entire formula is generated. Moreover, the adversarial learning in this paper means that the samples generated by the generator are expected to be adversarial examples for the discriminator, since the samples are not fitted by the discriminator, which is slightly different from the traditional adversarial learning algorithm.

Algorithm 1 Adversarial Training Framework

```
Input: A set of labelled signals $X = X^+ \cup X^-$, a vocabulary $\hat{\mathcal{V}}$ and a positive integer $T$.
Output: The optimal formal language $s_T$.
1: Initialize $G_\theta$ and $D_\varphi$ with random weights $\theta_m$, $\theta_p$, $W_D$.
2: Pre-train $D_\varphi$ with output from the generator.
3: Pre-train $G_\theta$ with the feature vectors from $D_\varphi$.
4: repeat
5:     for 5-steps do
6:         Generate a sequence of sentence $s_0, s_1, \cdots, s_T$.
7:         for t in $1 : T + 1$ do
8:             Store $f_t^D$ from $D_\varphi$.
9:             Compute $Q(s_t, g_t)$ by Monte Carlo Search.
10:            Compute $g_t$ by Eqn.(3).
11:            Update Mapper parameters by Eqn.(6).
12:            Update Policy parameters by Eqn.(7).
13:    for 5-steps do
14:        Use current generator $G_\theta$ to generator samples and calculate their real robustnesses.
15:        Train discriminator $D_\varphi$ for 50 epochs by Eqn.(5).
16: until Convergence
```

IV. Case Study

A. Fault Diagnosis for Rolling Element Bearings

In this section, we investigate the properties of the proposed method with real experimental data. The real data sets are from a set of rolling element bearings and we want to diagnose the faults. To get the fault signals of rolling element bearings for training and testing, the single pitting was introduced to the surface of the race or the rolling body of the bearings by electrical-discharge machining method.

1) Experiment Setup: The data collection test rig is similar to [22] as shown in Fig.5, where an a.c. motor drove the shaft of the rotational machine through a rub belt and a shaft coupling. The tested bearing is assembled on the shaft. The data acquisition (DAQ) system for the test rig is based on NI PXI system (a NI PXI-1042 chassis with NI PXI-4472 modules), where the data is sampled with an accelerometer

(Kistler 8791A250) sensor located on a bracket by an adhesive mounting. During the test, a series of GB203 rolling element bearings are used to conduct the experiment and the faults are introduced to the surface of the race or the rolling element by the electrical-discharge machining method. Each bearing has only one fault and each fault has been introduced to at least 5 bearings. During the experiment, we increase the difficulty of fault diagnosis by varying the speed of the shaft from 10 HZ to 30 Hz and the sampling rate is set to 12 kHz.

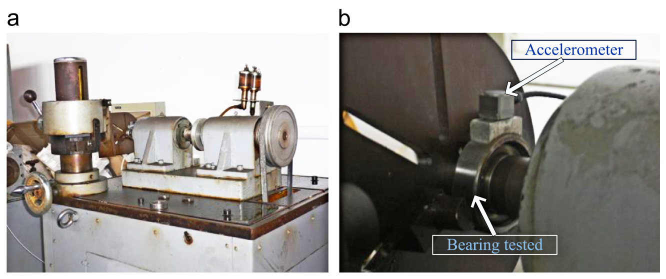


Fig. 5: (a) The rolling element bearing test rig and (b) the location of the accelerometer.

We introduced four kinds of states for the bearings, which are normal, rolling element fault, inner race fault, and outer race fault. 200 pieces of signals with length 8192 for each (sampled within 0.68 second) were collected for each condition. To obtain the spectrum of each signal, we applied the wavelet package transform (WPT) with a depth of level 3 to the signals in order to obtain their wavelet packet transform coefficients, which has been proven to be a powerful tool for signal processing [23]. There are 8 components at level 3 and each of them has a length of 1024. Then, we calculate the spectral kurtosis for each component (Fig.6) shows an example of the spectrums). Therefore, we have 200 signal pieces for each bearing condition, and each signal has 8 dimensions. To construct the positive set for inner fault, 150 pieces of inner fault signals are used, and the negative set from the other three conditions' signals (50 signal pieces for each condition and 150 in total). Therefore, the size of X is 300, and the labelled sets for normal, outer race fault and rolling element fault are constructed accordingly. To construct the testing set for the inner fault conditions, the positive test examples are the rest 50 pieces, and the negative test examples come from the other three bearing conditions (50 pieces for each condition that are un-used for training). Other testing sets are constructed accordingly.

Before we start the training process, we generate 1000 words randomly, including 100 initial words in $\hat{V}_0$. In this paper, we choose a CNN as the feature extractor. The extracted features are different from the features in [9], which are designed manually. The length of the sentence is set to 10 (i.e., $T = 9$ in problem 1). The CNN kernel size ranges from 1 to 9. The number of each kernel is between 20 to 50. Dropout with a keep rate of 0.5 and L2 regularization is performed to avoid over-fitting. For the generator, the Mapper produces a 18-dimensional goal embedding feature vector g_t using the features extracted from CNN. Moreover, the goal duration C is set to 3 after some preliminary experiments.

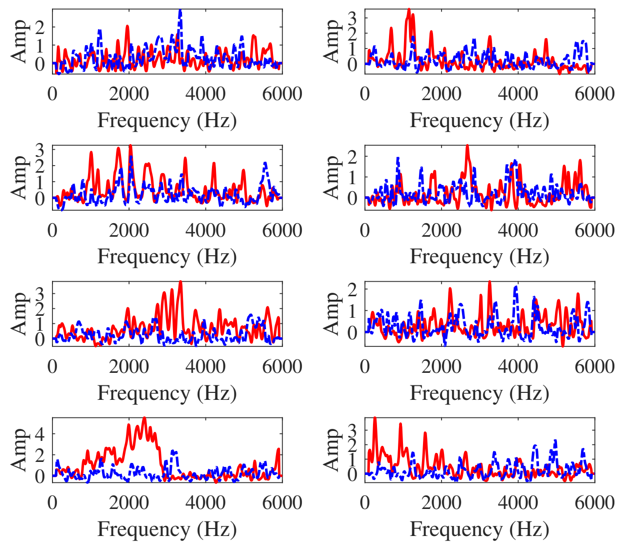


Fig. 6: The spectral kurtosis under variable speed conditions from normal (blue) and faulty (red) rolling element bearings, respectively. The signal is decomposed with WPT and has 8 components.

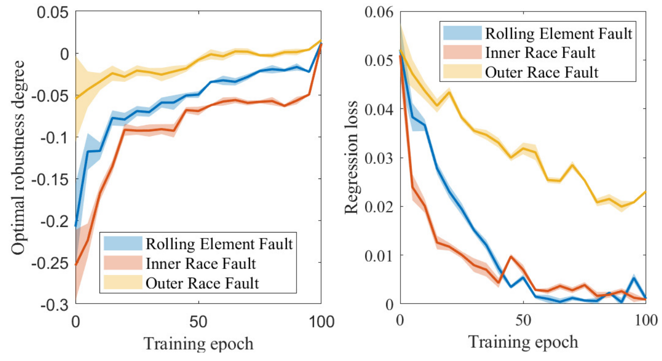


Fig. 7: (left): Average robustness and its standard deviation obtained; (right): Regression loss (root mean square) and its standard deviation during the training process.

During the training, the optimal robustness degree over the testing set obtained by the generated sentences (we generate many sentences with the trained network and choose the sentence that leads to maximum robustness degree) and the regression loss by the discriminator are used as the metrics. The training curves for each fault are depicted in Fig.7, where the results come from 10 trials. Based on the results, the proposed method can find a satisfactory (positive robustness) sentence for all the fault types within 100 epochs. Moreover, in order to demonstrate the efficiency of the proposed method in formal language generation for fault diagnosis, we compare it with the temporal logic inference methods in [9], [16], [17]. In these comparison experiments, the length of the sentence is set to 10 and the training time is limited for each experiment. We run the training on a 64bit Linux computer with a 16-core

CPU at 3.8 GHz, GeForce GTX 1070 GPU, and 64GB RAM.

In the first experiment setting, we check the performance at the 20th, 40th, 60th, and 80th minute. Table II-III show the average fault diagnosis error rate and its standard derivation among 10 trails for each method for inner race fault. The results in Table II show that the reinforcement learning method in [9] is not stable since the standard derivation keeps as a high value. Additionally, the method in [17] and our proposed method are stable, since the standard derivations decrease with respect to the increasing of time.

TABLE II: Average error rate and its standard deviation among 10 trails for four methods for inner race fault diagnosis.

	Average Error Rate(%)/ σ_e			
Time (min)	20	40	60	80
Our Method	43.65/8.577	23.2/6.041	15.4/4.131	**3.00/0.500**
Method in [16]	37.00/0.000	25.00/0.000	17.70/0.000	11.00/0.000
Method in [9]	36.55/8.252	23.3/7.059	11.65/5.794	13.15/7.056
Method in [17]	42.25/9.205	31.7/7.711	24.65/6.634	12.30/3.293

TABLE III: Average robustness and its standard deviation among 10 trails for four methods for inner race fault diagnosis.

	Average Robustness/ σ_ρ			
Time (min)	20	40	60	80
Our Method	-0.156/0.041	-0.081/0.012	-0.065/0.008	**0.002/0.003**
Method in [16]	-0.141/0.000	-0.119/0.000	-0.098/0.000	-0.053/0.000
Method in [9]	-0.242/0.121	-0.223/0.082	-0.065/0.010	-0.071/0.015
Method in [17]	-0.245/0.102	-0.144/0.076	-0.087/0.032	-0.013/0.009

The results in Table II-III indicate that our proposed method has the best performance, since our method can find the satisfactory sentences quickly. The reason for this is due to the scalability and the representative of the networks used in this paper. In contrast, the method in [16] needs to explore all combinations of the words to construct the optimal sentence. When the vocabulary is large or the length of the sentence is long, the computation time will increase dramatically. The methods in [9], [17] define the representation of sentences manually, which is not representative. The length of the sentence is important, which affects the efficiency of the algorithm. A too short sentence will not have enough expressiveness capacity, while a too long sentence will waste the computational source. In order to further investigate the properties of our method, we conduct another experiment over the inner fault case, where the sentence length is set to 4, 6, 8, 10, and we use the average error rate among 10 trails at 80th minute as the metrics. The results in Table IV reveal that when the sentence length is small, the method in [16] will obtain a better performance, since it can find the sentences with optimal parameters within 80 minutes. However, when the length of the sentence increases, [16] has worse performance, since the algorithm does not converge yet. Moreover, when the length increases further, the comparison performance among training set and testing set indicate the method in [9] cannot obtain good performance neither, since the algorithm has not converged or been over-fitting, while our method and the method in [17] can avoid over-fitting. The reason for this is that our method and the method [17] extend the sentence based on some estimated distribution about the state-space, which is robust to noise. In summary, the method in [16] is suitable for short sentences and small vocabulary, the method in [9] is suitable for short sentences and large vocabulary, the method in [17] has an average performance for all cases, and our method outperform the others for long sentences with large vocabulary.

TABLE IV: Average error rate for our method and the methods in [9], [16], [17] with different sentence lengths

	Training set/Testing set.			
Sentence length T	4	6	8	10
Our Method	35.40/36.15	15.10/15.30	4.90/4.80	**2.50/3.00**
Method in [16]	20.65/21.70	19.00/19.75	12.90/13.50	11.00/12.70
Method in [9]	23.65/25.90	18.50/19.55	13.75/14.70	13.15/14.50
Method in [17]	22.50/26.50	20.90/21.95	15.80/16.50	12.30/13.70

TABLE V: Comparison with other data-driven methods.

	Error Rate(%)			
Method	Our method	method in [24]	method in [25]	method in [26]
Inner Race	3.00	4.25	4.50	**1.00**
Outer Race	1.50	2.50	2.50	**0.00**
Rolling Element	**0.00**	2.00	0.00	3.25

Table V shows the comparison results between our method and other data-driven methods, which indicates that our method has comparable performance with other methods. Even though the Fisher based method in [26] and supper vector machine based method in [25] have comparable performance in some cases, their interpretabilites are limited. The reason for this is that these data-driven methods have the feature extraction process, which maps the original data to a high dimensional feature space and suppresses the physical meaning of the original data.

B. Simulation Analysis

In this section, we further investigate the properties of the proposed method with different levels of noise and proportion settings of training-test samples. Since we need to manipulate the signal to noise ratio (SNR), we simulate the faulty rolling element bearing signals based on the bearing fault model in [27]. The faulty signal can be described as

$$\begin{aligned} x(t) &= \textstyle\sum_i A_i s(t - iM - \tau_i) + n(t), \\ s(t) &= e^{-Bt}\sin 2\pi f_n t, A_i = cos(2\pi f_A t + \varphi_A) + C_A, \end{aligned}$$

where τ_i is the tiny fluctuation around M during the i^{th} impact of the fault, B the coefficient of resonance damping attenuation, f_n the natural frequency related to bearing or system, φ_A and C_A are constant, f_A the modulation frequency, which can be 0 (outer race fault), shaft rotation frequency f_r (inner race fault), or cage rotation frequency (rolling element fault). In this experiment, the parameters of simulated bearing signal is set as follows: the natural frequency is 2000 Hz, sampling rate 12 kHz. The characteristic defect frequency for out race, inner race are $f_{bpo} = 20$ Hz and $f_{bpi} = 30$ Hz, respectively. The rotation frequency $f_r = 100$ Hz.

In the first simulation setting, we investigate the performance of the proposed method with different levels of noise by changing the signal to noise ratio (SNR). The training set, testing set and the network structure are constructed with the same way in Section IV-A. Since the rotational speed is fixed, and the frequency pattern of the signal is simply compared with the data from real experiments. The length of the sentence is set to 3, the training time is set to 80 minutes and other parameters of the networks are the same with Section IV-A. The average error rate and robustness of 10 trails are shown in Table VI, which show that the proposed method can obtain good performance even though the SNR is -10 dB, indicating the method is robust to noise.

TABLE VI: Results for different signal to noise ratio.

	Error Rate(%)/ $\rho(X, \varphi_T)$			
SNR (dB)	-20	-10	0	10
Inner Race	13.50/-0.322	3.70/-0.081	1.5.00/0.022	0.00/0.132
Outer Race	14.65/-0.343	3.50/-0.075	0.00/0.093	0.00/0.112
Rolling Element	17.00/-0.232	4.90/-0.054	2.00/0.128	0.00/0.213

In the second simulation setting, we investigate the performance of the proposed method with different proportion settings of training-test samples. The SNR for this experiment is set to -10 dB, and 200 pieces of signals with length 8192 for each (sampled within 0.68 second) were simulated for each condition with random starting time. Then the training and testing sets were constructed with different proportion settings of training-test samples. The network parameters and the length limit for the formula are the same as the first simulation setting. The results are shown in Table VIII, which indicate that a larger training set will lead to a better performance. When the proportion is larger than 40%/60%, a good performance can be obtained. Table VII shows the obtained SSL formula with proportion 60%/40%, where the formulas are interpretable and can be understood by human users. For example, formula φ_I can be read with plain English as "when inner race fault occurs, the 3rd component's spectral kurtosis should be always smaller than 1.45 between 1278 and 2220 Hz, and no smaller than 0.63 between 1278 and 2220 Hz, and the 5th component's spectral kurtosis should be no smaller than 2.38 between 2903 and 4412 Hz at least once." Other formulas can be interpreted accordingly. The interpretation indicates that the spectrum should be bounded within a frequency interval, then there exists at least a peak after or before the bounded interval, which is consistent with the failure mechanism of the bearing that fault signals are caused by a series of impacts or impulses, which excite the entire system where the bearing is mounted. Therefore, the fault diagnosis with formal methods may reveal the failure mechanism.

C. Statistical Testing

Considering the machine learning algorithm is usually robust to noise, the formula is generated based on the policy of the generator, which chooses the next word with a probability approach and it should have a high probability to overcome the effect of noise. Next, we illustrate the noise resistance property of the generated language with statistical testing.

Let us first consider a t-test for a given SSL formula. The t-test compares the mean values of two fragments of spectrum, thereby to test a whole spectrum fragment for a fault occurrence. In other words, if the null hypothesis is rejected, the current spectrum cannot be categorized as fault in the baseline samples.

In several cases, however, robustness degree does not imply a normal distribution, due to the min and max operators in the semantics of SSL formula. Therefore, the two-sample Kolmogorov-Smirnov test (K-S test) is used and the new objective is to test the equality of value distributions in the robustness from the baseline and tested operating modes. As a result, a piece of time series is considered an outlier if a null hypothesis of value distribution equality is rejected with a level of significance.

During the test, we set the significance level to 0.05, and sampled the signals with a window length of 8192 for fault diagnosis. We calculated the robustness value with respect to the signal's decomposed spectral kurtosis. Then we moved the window 200 steps forward along time axis and repeated the calculation. The resulting trace of robustness values was used to determine the distribution. For example, to test inner race fault formula φ_i, 100 inner race fault signals are used, plus the signals under the other states (i.e., normal, rolling element fault, outer race fault) as baseline samples. The results are provided in Table IX, which are the average results among 10 trails of learning process. The results show the formula learned with the proposed method is robust to noise.

D. Discussion

Even though the proposed method can efficiently find the optimal formulas for fault diagnosis, there are some issues that should be further studied. First, the performance of the formula for fault diagnosis is affected by the noise level, which is the fundamental shortage for logic based method. Further study will focus on signal processing techniques that suitable for formal method to suppress the effect of noise. Second, the training process is conducted off-line, which is not good for on-line applications. In our further work, we will develop unsupervised learning algorithm to update the formula, such that it can be used for on-line applications. Third, the length of the sentence affects the performance of the algorithm, which should deal with the trade-off between computational time and classification accuracy. Many experiments are needed to set this parameter. More efficient algorithm should be developed to set this parameter in our future research.

The adversarial training is a static defence method using a supervised learning strategy. However, it can be applied to dynamic state systems for two reasons. First, the training and testing data sets come from dynamic systems, which are assumed to cover enough states information of the systems for fault diagnosis. For example, the real bearing data sets come from rolling element bearings under changing speed operations. Second, SSL is an expressive language, it allows the variation of the fault patterns among frequency domain

TABLE VII: SSL formulas for the simulated signals for inner race, outer race and rolling element faults.

Fault Type	Formula
Inner Race	$\varphi_I = \Box_{[1278,2220]}(x_3(f) < 1.45) \wedge \Box_{[1278,2220]} x_3(f) \geq 0.63)) \wedge \Diamond_{[2903,4412]}(x_5(f) \geq 2.38)$
Outer Race	$\varphi_O = \Diamond_{[985,1196]}(x_2(f) \geq 2.49)) \vee \Box_{[4979,5777]} x_8(f) \geq 0.34)) \wedge \Box_{[4979,5777]}(x_8(f) < 1.41)$
Rolling Element	$\varphi_R = \Diamond_{[4011,5144]}(x_5(f) \geq 1.47)) \wedge \Box_{[1243,1929]} x_3(f) \geq 0.65)) \wedge \Box_{[958,2826]}(x_3(f) < 1.23)$

TABLE VIII: Results for different proportion settings of training-test samples.

	Error Rate(%)/ $\rho(X, \varphi_T)$			
training/testing	20%/80%	40%/60%	60%/40%	80%/20%
Inner Race	11.95/-0.143	4.65/-0.044	4.10/-0.064	2.50/-0.059
Outer Race	13.65/-0.246	5.10/-0.038	3.20/-0.019	0.00/0.078
Rolling Element	14.90/-0.126	5.20/-0.023	3.65/0.012	2.10/0.006

TABLE IX: Average Statistical Testing Results.

Test Method	**t-test**	**K-S test**
Reject null hypothesis	94.1	95.0
Accept null hypothesis	5.9	5.0

with spectral operators. For example, "eventually" ($\Diamond$) operator allows the fault patterns occur within a frequency interval, which can capture the fault patterns of a dynamic system with changing operation scenarios.

Focusing on the space complexity of Algorithm 1, the discriminator solves a regression problem, which the required space is in the order of $\mathcal{O}(T)$, where $T+1$ is the length of the formula. Regarding the generator, which is trained with reinforcement strategy, the searching space complexity of the Policy modular is in the order of $\mathcal{O}(N^T)$ based on the complexity analysis in [9], where N is the size of the vocabulary. However, the space complexity of the generator is in the order of $\mathcal{O}(N+T)$, since the structure of the networks is fixed and the space required is related to the vocabulary size and length of the formulas.

V. Conclusions

This paper introduces a novel method to generate formal languages for fault diagnosis. The formal language is called signal spectral logic, defining over signals' spectral kurtosis. The formal languages can be seen as interpretable classifiers, which provide interpretability for the fault diagnosis procedure. Experimental results indicate our method is robust to noise. To find the optimal formal languages, the adversarial training technique is used to address the sparse reward issue during language generation process, and the results indicate our method outperforms the state-of-the-art methods on generating long formal languages.

References

[1] P. Wang, C.-M. Chen, S. Kumari, M. Shojafar, R. Tafazolli, and Y.-N. Liu, "HDMA: Hybrid D2D message authentication scheme for 5g-enabled VANETs," *IEEE Transactions on Intelligent Transportation Systems*, DOI: 10.1109/TITS.2020.3013928, 2020.

[2] M. Kordestani, M. F. Samadi, M. Saif, and K. Khorasani, "A new fault prognosis of MFS system using integrated extended kalman filter and bayesian method," *IEEE Transactions on Industrial Informatics*, dio: 10.1109/TII.2018.2815036, 2018.

[3] E. K. Wang, F. Wang, S. Kumari, J.-H. Yeh, and C.-M. Chen, "Intelligent monitor for typhoon in iot system of smart city," *The Journal of Supercomputing*, pp. 1–20, 2020.

[4] J. Gonzales and E. Mombello, "Fault interpretation algorithm using frequency-response analysis of power transformers," *IEEE Transactions on Power Delivery*, vol. 31, no. 3, pp. 1034–1042, 2015.

[5] H. D. M. de Azevedo, A. M. Araújo, and N. Bouchonneau, "A review of wind turbine bearing condition monitoring: State of the art and challenges," *Renewable and Sustainable Energy Reviews*, vol. 56, pp. 368–379, 2016.

[6] J. Antoni and R. B. Randall, "The spectral kurtosis: application to the vibratory surveillance and diagnostics of rotating machines," *Mechanical systems and signal processing*, vol. 20, no. 2, pp. 308–331, 2006.

[7] R. Taheri, R. Javidan, M. Shojafar, P. Vinod, and M. Conti, "Can machine learning model with static features be fooled: an adversarial machine learning approach," *Cluster Computing*, pp. 1–21, 2020.

[8] J. Liu, F. Qu, X. Hong, and H. Zhang, "A small-sample wind turbine fault detection method with synthetic fault data using generative adversarial nets," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 3877–3888, 2018.

[9] G. Chen, M. Liu, and Z. Kong, "Temporal-logic-based semantic fault diagnosis with time-series data from industrial internet of things," *IEEE Transactions on Industrial Electronics*, DOI: 10.1109/TIE.2020.2984976, 2020.

[10] Z. Xu and A. A. Julius, "Robust temporal logic inference for provably correct fault detection and privacy preservation of switched systems," *IEEE Systems Journal*, vol. 13, no. 3, pp. 3010–3021, 2019.

[11] K. Havelund, D. Peled, and D. Ulus, "First-order temporal logic monitoring with bdds," *Formal Methods in System Design*, pp. 1–21, 2019.

[12] H. Gunadi and A. Tiu, "Efficient runtime monitoring with metric temporal logic: A case study in the android operating system," in *International Symposium on Formal Methods*. Springer, 2014, pp. 296–311.

[13] X. C. Ding, C. Belta, and C. G. Cassandras, "Receding horizon surveillance with temporal logic specifications," in *49th IEEE Conference on Decision and Control*. IEEE, 2010, pp. 256–261.

[14] O. A. Beg, L. V. Nguyen, T. T. Johnson, and A. Davoudi, "Signal temporal logic-based attack detection in dc microgrids," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 3585–3595, 2018.

[15] J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia, "Robust online monitoring of signal temporal logic," *Formal Methods in System Design*, vol. 51, no. 1, pp. 5–30, 2017.

[16] Z. Kong, A. Jones, and C. Belta, "Temporal logics for learning and detection of anomalous behavior," *IEEE Transactions on Automatic Control*, vol. 62, no. 3, pp. 1210–1222, 2016.

[17] G. Chen, M. Liu, and J. Chen, "Frequency-temporal-logic-based bearing fault diagnosis and fault interpretation using bayesian optimization with bayesian neural networks," *Mechanical Systems and Signal Processing*, vol. 145, p. 106951, 2020.

[18] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3540–3549.

[21] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[22] H. Jiang, J. Chen, G. Dong, T. Liu, and G. Chen, "Study on hankel matrix-based svd and its application in rolling element bearing fault

diagnosis," *Mechanical Systems and Signal Processing*, vol. 52, pp. 338–359, 2015.

[23] Y. Wang, G. Xu, L. Liang, and K. Jiang, "Detection of weak transient signals based on wavelet packet transform and manifold learning for rolling element bearing fault diagnosis," *Mechanical Systems and Signal Processing*, vol. 54, pp. 259–276, 2015.

[24] J. ZHOU, X. Li, X. Shi, W. Wei, and B. Wu, "Predicting pillar stability for underground mine using fisher discriminant analysis and svm methods," *Transactions of the Nonferrous Metals Society of China*, vol. 21, no. 12, pp. 2734–2743, 2011.

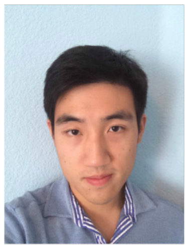
[25] G. Chen and J. Chen, "A novel wrapper method for feature selection and its applications," *Neurocomputing*, vol. 159, pp. 219–226, 2015.

[26] E. Alba, J. Garcia-Nieto, L. Jourdan, and E.-G. Talbi, "Gene selection in cancer classification using pso/svm and ga/svm hybrid algorithms," in *2007 IEEE Congress on Evolutionary Computation*. IEEE, 2007, pp. 284–290.

[27] F. Hou, J. Chen, and G. Dong, "Weak fault feature extraction of rolling bearings based on globally optimized sparse coding and approximate svd," *Mechanical Systems and Signal Processing*, vol. 111, pp. 234–250, 2018.

Gang Chen received the B.S. degree and M.S. degrees in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2012 and 2015, respectively and the Ph.D. degree in mechanical and aerospace engineering from the University of California, Davis, Davis, CA, in 2020.

His research interests include machine learning, formal methods, control, signal processing and fault diagnosis.



Peng Wei received the B.S. in thermal and power engineering from Huazhong University of Science & Technology, China, and the M.S. in mechanical engineering from Arizona State University, AZ, USA, in 2015 and 2016, respectively. He is currently working towards his Ph.D. in mechanical and aerospace engineering from the University of California, Davis, Davis, CA.

His research interests include UAV control, optimal control and reinforcement learning.

Huiming Jiang received the B.S. degree in mechanical engineering from Xi'an Jiaotong University, in 2011, and the Ph.D. degree in mechanical engineering from Shanghai Jiao Tong University, in 2017. She is currently a Lecturer with the School of Mechanical Engineering, University of Shanghai for Science and Technology. Her current research interests include machinery condition monitoring and fault diagnosis, intelligent fault diagnostics, and performance degradation assessment.

Mei Liu received her B.Sc.degree in Mthematics from China University of Mining and Technology in 2012, and her Ph.D degree in Control Science and Engineering from University of Science and Technology of China in 2017. From August 2017 to February 2019, she worked with The University of Hong Kong and The Hong Kong Polytechnic University as a Research Associate/Assistant. She is currently an Associtae Professor in the Department of Automation at Tianjin University, Tianjin, China. Her current research interests include negative imaginary systems, positive real systems, state-space symmetric systems and fault diagnosis.