Journal of Dynamics, Monitoring and Diagnostics, 2023, 2, 30-41
https://doi.org/10.37965/jdmd.2023.143

ISTP
RESEARCH ARTICLE

Intelligent Fault Diagnosis for Planetary Gearbox Using Transferable Deep Q Network Under Variable Conditions with Small Training Data

Hui Wang,[2] Jiawen Xu,[2] and Ruqiang Yan[1,2]
[1]School of Mechanical Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[2]School of Instrument Science and Engineering, Southeast University, Nanjing 210096, China

(Received 24 October 2022; Revised 25 October 2022; Accepted 18 January 2023; Published online 18 January 2023)

Abstract: Effective fault diagnosis of planetary gearboxes is critical for ensuring the safety and dependability of mechanical drive systems. Nevertheless, variable conditions and inadequate fault data bring huge challenges to its practical fault diagnosis. Taking this into account, this study presents a new intelligent fault diagnosis (IFD) approach for planetary gearbox using a transferable deep Q network (TDQN) that merges deep reinforcement learning (DRL) and transfer learning (TL). First, a DRL environment simulation is designed by a predefined classification Markov decision process. Then, leveraging varied-size convolutions and residual learning, a multiscale residual convolutional neural network agent for TDQN is created to automatically learn meaningful features directly from vibration signals while avoiding model degradation. Next, a large source dataset is obtained from complex conditions, and this agent learns an IFD policy via autonomous interaction with the data environment. Finally, a parameter-based TL strategy is adopted to retrain the model on target datasets with variable conditions and small training data, which is conducted by fine-tuning the model parameters gained from the source task to accomplish target tasks. The results show that this TDQN outperforms not only state-of-the-art methods in a source task with an accuracy of 98.53% but also in two target tasks with 99.63% and 98.37%, respectively.

Keywords: convolutional neural network; deep reinforcement learning; gearbox; fault diagnosis; transfer learning

ABBREVIATIONS

A	Action space
CMDP	Classification Markov decision process
CNN	Convolutional neural network
DBN	Deep belief network
DL	Deep learning
DQN	Deep Q network
DRL	Deep reinforcement learning
D_s	Source domain
D_t	Target domain
IFD	Intelligent fault diagnosis
Main-Net	Current network
MDP	Markov decision process
MK-ResCNN	Multiscale kernel-based residual CNN
MRCNN	Multiscale residual CNN
MSCTN	Multiscale convolutional transfer network
OA	Overall accuracy
P	State transition probability
PGB	Planetary gearbox
R	Reward function
ReLU	Rectified linear unit
RL	Reinforcement learning
RNN	Recurrent neural network
S	State space
SAE	Stacked autoencoder
SVM	Support vector machine
Target-Net	Target network
TCNN	Transferable CNN
TDQN	Transferable deep Q network
TL	Transfer learning
WDCNN	CNN with wide first-layer kernels
WT	Wavelet transform
γ	Discount factor

I. INTRODUCTION

Planetary gearbox (PGB), as a vital life-limited transmission component, has been broadly used in mechanical equipment, such as wind turbines, ships, and helicopters. However, it is prone to various failures such as gear cracks, tooth breakage, and bearing damage due to harsh working conditions [1,2]. Such failures that directly affect the reliability of equipment may cause huge financial losses and even casualties [3]. Hence, it is vital to investigate advanced fault diagnosis methods to detect PGB failures effectively.

With the fast growth of artificial intelligence, intelligent fault diagnosis (IFD) has piqued the interest of industry and academia [2,4]. Traditional IFD methods that combine shallow models (*e.g.*, support vector machine (SVM)) with feature extractors (*e.g.*, wavelet transform (WT)) are difficult to meet current IFD requirements. Recently, deep learning (DL) techniques, such as deep belief network (DBN) [5], recurrent neural network (RNN) [6], and convolutional neural network (CNN) [3], have overcome the limits of traditional shallow models and have been applied in the IFD field. In particular, CNN, which has the merits of locality, shift-invariance, and hierarchical representations, is popularly favored by researchers [2,7,8]. Zhang *et al.* proposed a CNN with wide first-layer kernels (WDCNN)

Corresponding author: Ruqiang Yan (e-mail: yanruqiang@xjtu.edu.cn).

for bearing fault diagnosis with good anti-noise and adaptation, obtaining accuracy near 100% [7]. Azamfar *et al.* presented an IFD model for PGB based on 2D CNN and multi-sensor data that beats SVM, decision tree, etc., [8]. Liu *et al*. designed a multiscale kernel-based residual CNN (MK-ResCNN) to realize fault diagnosis of rotating motors with an accuracy of 94.47% [9]. Although these CNN-based methods have achieved good results, there are still inherent limits such as static learning way and lacking decision-making ability [10,11], which restrict the generalization ability and intelligence level of fault diagnosis.

Deep reinforcement learning (DRL) [12], which unites DL perception with reinforcement learning (RL) decision-making ability, offers a novel idea for IFD and may tackle the limitations mentioned above. In the DRL, an agent can autonomously learn the knowledge or policy by dynamic self-exploration and reward returned from the environment. Similar to the human cognitive process, DRL has strong generality and intelligence level, and thus, it is deemed the future of general artificial intelligence [13]. Until now, DRL has been used successfully in the field of games, robotics, and automatic driving [12,14]. Recently, researchers have attempted to use DRL algorithms to address machinery fault diagnosis problems. Ding *et al.* created a stacked autoencoder (SAE)-based deep Q network (DQN) for IFD of rotating machinery, which illustrated the feasibility and effectiveness of the DRL for fault diagnosis [10]. Wang *et al.* presented a fault diagnostic framework for PGB using time-frequency representation and a 2D CNN-based DQN, which achieved better generalization than regular CNN [15]. Wang *et al.* proposed a DRL-based bearing fault diagnosis model using the actor-critic algorithm-based 1D CNN, which obtained more accurate results than SVM and CNN [16]. Despite achieving excellent performance, these approaches are primarily used for fault diagnosis in stationary working conditions and rely on sufficient training data with the same distribution. It means that each time a new diagnosis task appears, a large quantity of data is required to train the model from scratch, resulting in high computational costs. It is likewise unrealistic to collect enough fault samples for model training each time. In addition, since PGB typically works under variable conditions that result in data collected with significant internal variability, deep-rich features need to be extracted for fault diagnosis. Unfortunately, the above DRL models with ordinary CNN can only capture fixed-scale fault features and may encounter a degradation problem as the model deepens [9], making it challenging to meet IFD needs under complex variable conditions. These shortcomings hinder the wide use of current DRL-based IFD methods in variable conditions with small training data.

Currently, transfer learning (TL) can well overcome the difficulties of training models from scratch under insufficient samples and sample distribution differences, and various TL strategies have been used for fault diagnosis [7,17,18]. Shao *et al.* proposed a deep TL model for machine fault diagnosis that combined WT images and a pre-trained CNN, achieving over 99.60% accuracy and speeding up the training [18]. He *et al.* presented a deep transfer multi-wavelet autoencoder to diagnose gearbox faults with small target training samples [19]. Chen *et al.* suggested a transferable CNN (TCNN) for rotary machinery fault diagnosis that used prior knowledge to enhance the target domain performance in the absence of adequate training data [20]. Although these methods have the limitations of traditional DL methods, they perform excellently in fault diagnosis tasks under variable working conditions. This prompts us to further explore the fusion of DRL and TL for PGB fault diagnosis.

In this work, we provide a DRL method using a transferable deep Q network (TDQN) for PGB fault diagnosis under varying conditions with small training data. Specifically, a multiscale residual CNN (MRCNN)-based DQN fault diagnosis model is developed to learn the diagnostic policy from a complex source task via dynamic interaction with the data environment. The well-trained DRL model is then fine-tuned by using small target training samples through a parameter TL strategy to realize fault diagnosis under variable conditions. The main contributions of this paper are listed below.

(1) In the DRL framework, a new MRCNN agent for the DQN algorithm is presented to capture rich multiscale features from vibration signals without handcrafted features. The diagnostic policy is learned in a weak feedback dynamic interaction via a classification Markov decision process (CMDP), which enhances the generalization ability.

(2) By incorporating parameter TL, the proposed TDQN can well realize PGB fault diagnosis under variable conditions with small target training samples. It is promising to promote the actual IFD application of DRL.

(3) This method is first verified by a complex source task, and then, its transferability is evaluated by two different target working conditions. Compared with existing methods, the results show the superiority of the proposed method.

The remainder of this paper is arranged as follows. Section II briefly reviews the relevant theories. Section III gives the diagnostic framework based on the TDQN. Section IV describes datasets and experimental details. Section V shows results and analysis. Section VI draws some conclusions.

II. RELATED THEORIES

Since our work is mainly based on the DQN algorithm and parameter TL, thus we first briefly review the DQN model. Following that, the fundamentals associated with parameter TL are introduced.

A. BRIEF INTRODUCTION OF DQN

RL is based on a Markov decision process (MDP), denoted by a tuple $\{S, A, P, R, \gamma\}$, where S, A, P, R, and γ are the state space, action space, transition probability, reward, and discount factor, respectively. In RL, the agent learns a policy π through dynamic interaction with the environment, where π is the probability mapping from a state $s \in S$ to an action $a \in A$. Typically, the policy is evaluated by the state-action value function $Q^{\pi}(s, a)$ as follows:

$$Q^{\pi}(s,a) = E_{\pi}\left[R_t = \sum_{t'=1}^{T} \gamma^{t'-1} r_{t'} \middle| s_t = s, a_t = a, \pi\right] \quad (1)$$

where $E_{\pi}(\cdot)$ denotes the expectation under the policy π, R_t signs the cumulative reward, and T is the end-time step.

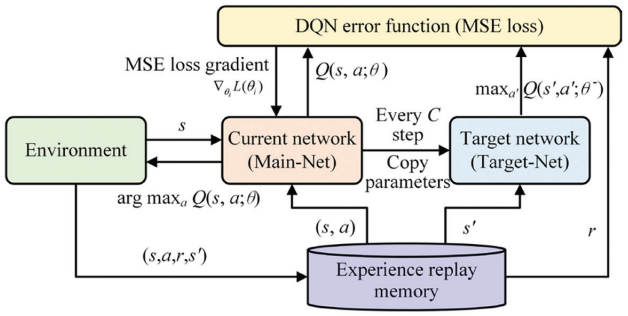

Fig. 1. The basic flow of the DQN algorithm.

The optimal policy π^* guides the agent to select an optimal action at a state, and it is expressed as

$$\pi^* = \arg\max_{a\in A} Q^{\pi}(s,a). \tag{2}$$

Unfortunately, Eq. (1) is difficult to calculate in practice. For this, Mnih *et al*. [12] proposed the DQN algorithm, which used deep neural networks (DNN) to estimate the state-action value, represented as $Q(s, a; \theta) \approx Q(s, a)$. Here, DNN is called Q network, θ is its parameters. The DQN algorithm is illustrated in Fig. 1, where two key techniques are used to ensure training stability. One is the experience replay mechanism, which stores interaction data $e = (s, a, r, s')$ between agent and environment in experience replay memory ε, and then, mini-batch experience trajectories are randomly sampled from replay memory ε for updating the network. The other is to estimate target state values by setting an additional target network (Target-Net), which is updated by copying the weights of the current network (Main-Net) every delayed C step. Notably, the network updating is constrained by the mean-square error (MSE) loss.

$$L(\theta_i) = E_{e\sim\varepsilon}\left[\left(r + \gamma\max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right] \tag{3}$$

where $Q(s,a;\theta_i)$ is the Main-Net, $Q(s',a';\theta_i^-)$ is the Target-Net, and $y_i = r + \gamma\max_{a'} Q(s',a';\theta^-)$ is the target Q value. Ultimately, the stochastic gradient descent technique is used to update the network parameters.

$$\nabla_{\theta_i} L(\theta_i) = E_{e\sim\varepsilon}[(Y_i - Q(s,a;\theta_i))\nabla Q(s,a;\theta_i)] \tag{4}$$

In addition, the ϵ-greedy strategy [12,13] is used to balance the exploration and exploitation during interactive learning. Based on DQN, this work will develop a new framework for gearbox fault diagnosis.

B. PARAMETER TL STRATEGY

Variable conditions cause changes in the distribution across the training and testing data. The disparity in data distribution will have a great influence on the model performance [21,22]. With auxiliary training data, parameter transfer as an effective TL tool can well address domain mismatch problems, especially with the support of a fine-tuning algorithm [18,19,21]. More specifically, it fine-tunes a well-trained model with small auxiliary training data to make this model fit for new tasks. Via parameter transfer, knowledge learned in the source domain (D_s) can be transferred to the target domain (D_t) and used to fulfill target tasks. Notably, while having different data distributions, D_s and D_t exist in a certain association. Generally, D_s has sufficient training data, whereas D_t has a little amount of training data. Parameter transfer mainly consists of two ordinal steps: 1) D_s is used to pre-train a source model; 2) Weight parameters of the well-trained source model are copied to the target model as its initial parameters, and then, the target model is fine-tuned by using small data from D_t. Accordingly, deep models can be trained faster and more easily through parameter transfer than by learning from scratch [18].

Therefore, this work introduces parameter transfer to enhance DQN and created a TDQN model to realize PGB fault diagnosis under variable conditions with small training data. The IFD framework is illustrated in section III specifically.

III. METHODOLOGY

This section begins with the creation of a CMDP, which is a base for DRL environment simulation. On this premise, a new MRCNN for the DRL agent is developed. Finally, an IFD framework based on TDQN is provided.

A. CMDP DEFINITION

The first step in using DRL is to describe the task by MDP. Essentially speaking, fault diagnosis is a classification task; thus, a new CMDP is created to formalize it, inspired by [15,23]. Let a training dataset is $D = \{(x_1, l_1), (x_2, l_2), \ldots, (x_n, l_n)\}$, where x_i is the ith sample and l_i is the ith label related to x_i. A CMDP likewise contains a tuple $\{S, A, P, R, \gamma\}$, and related elements in this work are illustrated as follows:

S: It is made up of sample states produced by the training dataset D. Initial environment state s_1 is associated with the sample x_1. Likewise, state s_t at time step t matches the sample x_t. Each time the environment is initialized, the order of the samples in dataset D is randomly shuffled.

A: It comprises all classification actions allowing for agent taking, and action a_t is associated with a label l_t. For a classification task, $A = \{0, 1, \ldots, K-1\}$, where K is categories.

P: The state transition probability $p(s_{t+1}|s_t, a_t)$ is deterministic, which means that the agent receives the next state s_{t+1} from the environment in the sample order of dataset D.

R: A reward r_t is environment feedback that assesses the success or failure of the agent's classification action and aids the agent in learning the optimal classification strategy [15]. Hence, we stipulate that if the agent correctly selects a recognition action matching the reference label, the environment returns a positive reward $R((s_t, a_t), a_t = l_t) = +1$, or else, the environment gives a negative reward $R((s_t, a_t), a_t \neq l_t) = -1$.

γ: It belongs to the range [0,1] and is to balance the immediate and future rewards. Since the classification agent is obsessed with instant rewards, γ is often small.

Based on the above CMDP, a data environment simulation can be created for dynamic interaction [24]. Once the agent selects a failed action during the interaction, the environment ends the episode and restarts a round of interactive learning. The goal of the agent is to find an optimum policy π^*_θ, which assists the agent in performing

optimal recognition action given states. Because the agent structure is critical to the sensing environment state, a new MRCNN will be proposed.

B. MRCNN STRUCTURE

Vibration signals obtained from variable conditions have high complexity and internal variability. To effectively diagnose PGB faults, rich features from raw signals must be captured. Therefore, this work proposes a new MRCNN structure as the network of the DQN agent (Fig. 1). Via multiscale convolutions and residual learning, MRCNN automatically learns abstract and rich fault features directly from complicated raw signals. It is a complete end-to-end workflow that does not rely on manual features. Figure 2 depicts the overall architecture of MRCNN, where the symbols "L," "C_i," "Conv," "MaxP," "GAP," and "Res-$B_{1\text{-}2}$" refer to sample length, branch channel number, convolutional layer, max-pooling layer, global average pooling, and residual blocks, respectively. Overall, MRCNN mainly has a multiscale fusion (MSF) layer, a MaxP layer, and residual units. The following are the specifics.

(1) *MSF Layer:* Generally, a fixed wide-kernel convolution is used in the first layer of CNN to extract low-frequency features directly from raw vibration signals for fault diagnosis [7,9]. However, fixed single-scale filters cannot fully learn discrepant features directly from raw signals, and signal information may be underutilized, resulting in a loss in overall model diagnostic performance, especially for fault diagnosis under complicated variable conditions. Given this, in MRCNN, the first stage of the MSF layer is designed to capture rich discriminant features from input signals via three various filter sizes. For the input $\boldsymbol{X}_{in} \in \mathbf{R}^{L\times 1}$, the first multiscale output of the MSF layer is given by

$$\boldsymbol{X}^{I}_{MSF} = Concat\Big(Conv_{k=25,64,112}(\boldsymbol{X}_{in})\Big). \tag{5}$$

where $Conv_k$ denotes a 1D convolutional layer with a kernel size of k, and *Concat* represents the concatenation operation along the channel axis. The output of $Conv_k$ is calculated by

$$\boldsymbol{X}_k = \sigma(\boldsymbol{W}_k * \boldsymbol{X}_{in} + \boldsymbol{b}_k). \tag{6}$$

where "*" is the convolution operation, $\boldsymbol{W}_k$ and $\boldsymbol{b}_k$ denote the weight and bias term, respectively, and $\sigma(\cdot)$ is the activation function, which is often the rectified linear unit (ReLU) [25].

Features of three filter scales are extracted preliminary from raw signals using Eq. (5). Next, the second stage of the MSF layer adopts a 1D multiscale module inspired by the 2D Inception architecture [26] to further refine multiscale features from the previous layer. Convolutional kernels with 1×1, 3×1, and 5×1 sizes are mostly used in this stage. Concretely, its output is realized by four parallel branches, as shown in Fig. 2, and these multiscale features are concatenated as the input of the next stage. The output of the MSF layer is described by

$$\boldsymbol{X}^{II}_{MSF} = \begin{bmatrix} \sigma(\boldsymbol{X}^{I}_{MSF} * \boldsymbol{W}_{1\times1} + \boldsymbol{b}_{1\times1}) \\ \sigma(\sigma(\boldsymbol{X}^{I}_{MSF} * \boldsymbol{W}_{1\times1} + \boldsymbol{b}_{1\times1}) * \boldsymbol{W}_{3\times1} + \boldsymbol{b}_{3\times1}) \\ \sigma(\sigma(\boldsymbol{X}^{I}_{MSF} * \boldsymbol{W}_{1\times1} + \boldsymbol{b}_{1\times1}) * \boldsymbol{W}_{5\times1} + \boldsymbol{b}_{5\times1}) \\ \sigma(MaxP(\boldsymbol{X}^{I}_{MSF}) * \boldsymbol{W}_{1\times1} + \boldsymbol{b}_{1\times1}) \end{bmatrix}. \tag{7}$$

(2) *MaxP Layer:* Following the MSF layer, a max pooling layer is used as a down-sampling operation to reduce feature size and computation cost while increasing the generalization. The output of the MaxP layer is denoted as

$$\boldsymbol{X}_{MaxP} = \max_{(j-1)\boldsymbol{W}_p+1\le k\le j\boldsymbol{W}_p} \Big\{\boldsymbol{X}^{II}_{MSF}(k)\Big\}. \tag{8}$$

where $\boldsymbol{X}^{II}_{MSF}(k)$ indicates the kth value in each feature vector, $\boldsymbol{W}_p$ means the pooling window. Notably, an overlapping MaxP layer in which a pooling window is greater than a pooling stride is used as a feature reduction operation in MRCNN.

(3) *Res-$B_{1\text{-}2}$ unit:* Complex working situations make PGB fault identification more challenging; thus, deep features should be retrieved to improve the performance. Nevertheless, as the network deepens, the model performance may degrade, making model training difficult. Residual learning [27] can well solve this issue; thus, it is embedded in the proposed MRCNN structure to facilitate model training [9]. As depicted in Res-B_1 of Fig. 2, the core of residual learning is the residual subblock.

Based on the residual learning concept [27], Res-B_1 employs a shortcut connection that connects input $\boldsymbol{X}_{MaxP}$ to stacked weight layers. The Res-B_1 output is obtained by

$$\boldsymbol{X}_{Res\text{-}B_1} = F(\boldsymbol{X}_{MaxP}) + \boldsymbol{X}_{MaxP}, \tag{9}$$

$$F = \boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{X}_{MaxP}). \tag{10}$$

where $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are the weights, and $F(\boldsymbol{X}_{MaxP})$ is the residual mapping, which often adopts two stacked Conv layers. $F(\boldsymbol{X}_{MaxP})$ and $\boldsymbol{X}_{MaxP}$ must have identical dimensions

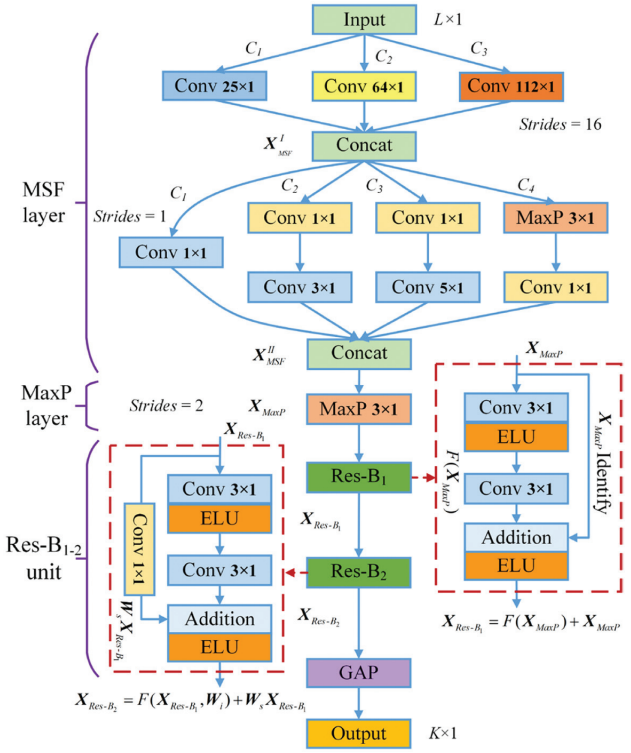

Fig. 2. The MRCNN structure for the DQN agent.

since they are fused by element-wise addition. Otherwise, a linear projection is used to match the output size. For example, a 1×1 convolution along the shortcut connection is used to expand the dimension for the input $\boldsymbol{X}_{Res\text{-}B1}$ in the second residual block (Res-B_2). The output of the residual block Res-B_2 can be written as

$$\boldsymbol{X}_{Res\text{-}B_2} = F(\boldsymbol{X}_{Res\text{-}B_1}, \boldsymbol{W}_i) + \boldsymbol{W}_s \boldsymbol{X}_{Res\text{-}B_1}. \tag{11}$$

where $\boldsymbol{W}_i$ is the weight layer and $\boldsymbol{W}_s$ is a linear projection.

Algorithm 1: Interaction Learning Process

1: Initialize replay buffer with the maximal capacity N;
2: Initialize Main-Net and Target-Net with weights θ and θ^{-}, respectively;
3: Initialize ϵ-greedy strategy, learning rate, mini-batch size, terminal flag, etc.;
4: **for** episode = 1, M **do**
5: Initialize data environment, randomly shuffle training dataset;
6: **for** $t = 0$, T-1 **do**
7: Observe the current state s_t from the data environment;
8: Select action at by the ϵ-greedy strategy: Randomly select a_t with the probability ϵ, or by maximizing Q value, that is, $argmax\ Q(s_t, a_t; \theta)$;
9: Environment returns agent a reward r_t based on the reward strategy;
10: Transition to next an environment state s_{t+1} by the order of samples;
11: Store transition trajectory (s_t, a_t, r_t, s_{t+1}) in replay buffer;
12: **if** the current replay buffer capacity reaches a certain number **do**
13: Sample random mini-batch experience data from the replay buffer;
14: Conduct a gradient descent procedure using the RAdam [28] on eq. (4) to update the weight parameters θ of the Main-Net;
15: Every C delayed step, update Target-Net by copying $\theta^{-} = \theta$;
16: **end for**
17: **end for**
18: **return** The parameter θ^{*} on the Main-Net for optimal action policy π_{θ}^{*}.

As a result, MRCNN adopts two sequential residual blocks to extract deep features for PGB fault diagnosis. After that, a GAP layer is adopted for feature aggregation, and a K-size fully connected layer outputs Q values linearly. Based on the above layers, an MRCNN is designed as the Main-Net and Target-Net network structure of DQN (Fig. 1). Finally, Algorithm 1 exhibits the agent interaction learning process according to the created CMDP and the DQN flow (Fig. 1).

C. TDQN-BASED DIAGNOSTIC FRAMEWORK

As shown in Fig. 3, a new TDQN-based framework for PGB fault diagnosis is constructed according to Algorithm 1 and the parameter transfer strategy. Specifically, this framework covers five main steps listed below.

Step 1: Data acquisition and division. Vibration signals of health types collected from different conditions are segmented and normalized into the range [−1, 1]. The datasets of the source domain (D_s) and target domains (D_t) are created, and the quantity of training and testing samples between them is confirmed.

Step 2: Model building. The data environment simulation is created according to the defined CMDP, and the agent structure of the TDQN model based on an MRCNN is developed. Details are depicted in Fig. 2.

Step 3: Model learning. According to the reward mechanism, the agent autonomously learns the diagnostic policy from the source domain D_s via interaction with the data environment, as illustrated in Algorithm 1.

Step 4: Model transfer. The model parameters learned from D_s are transferred and used to realize target tasks (D_t). Exactly, the well-trained TDQN model is fine-tuned on a small training dataset of D_t by only retraining Res-$B_{1\text{-}2}$, GAP, and the output layer via Algorithm 1, whereas low-level multiscale layers are frozen.

Step 5: Fault diagnosis. The fine-tuned TDQN is used to realize fault diagnosis on the testing dataset of D_t, and then, the recognition results are further analyzed.

IV. EXPERIMENT AND DATA DESCRIPTION

This section first assesses the proposed framework on a multi-speed source dataset (D_s). Subsequently, two target datasets (D_t) are collected to verify the effectiveness of this method in two different scenarios. Data descriptions are offered first, followed by experimental details and comparative methods.

A. SOURCE DOMAIN DATASET

PGB often operates under complex variable conditions. Thus, a source domain dataset (D_s^1) is collected from the drivetrain dynamics simulator (DDS) under variable speed conditions. Figure 4 depicts the experimental bench, which consists primarily of a driving motor, a speed controller, a PGB, a parallel gearbox, and a brake. During the experiments, vibration data are acquired using PCB 608A11 accelerometers installed at the radial input end of PGB, with a sampling rate of 5120 Hz. Moreover, a speed controller is used to adjust the rotational speed of the driving motor, which ranges from 0 to 2400 r/min, and the details are depicted in Fig. 5. Nine health types are physically simulated in PGB [15], including one healthy state (HEA), four gear faults (*i.e.*, CTF, MTF, RTF, and SWF), and four bearing faults (*i.e.*, BWF, CWF, IRF, and ORF), as shown in Table I. For each health type, vibration signals are collected under a speed ranging from 0 to 2400 r/min and a load of 0 N.m. In the above settings, ten tests were performed for each PGB health type to acquire abundant vibration data.

Collected vibration signals are segmented into samples using a sliding window with a step of 1024 data points, and each sample comprises 2048 data points. Finally, a source dataset D_s^1 is created with 2550 samples per type, totaling 22,950 samples in nine categories. During the test, we adopt five-fold cross validation as an evaluation strategy.

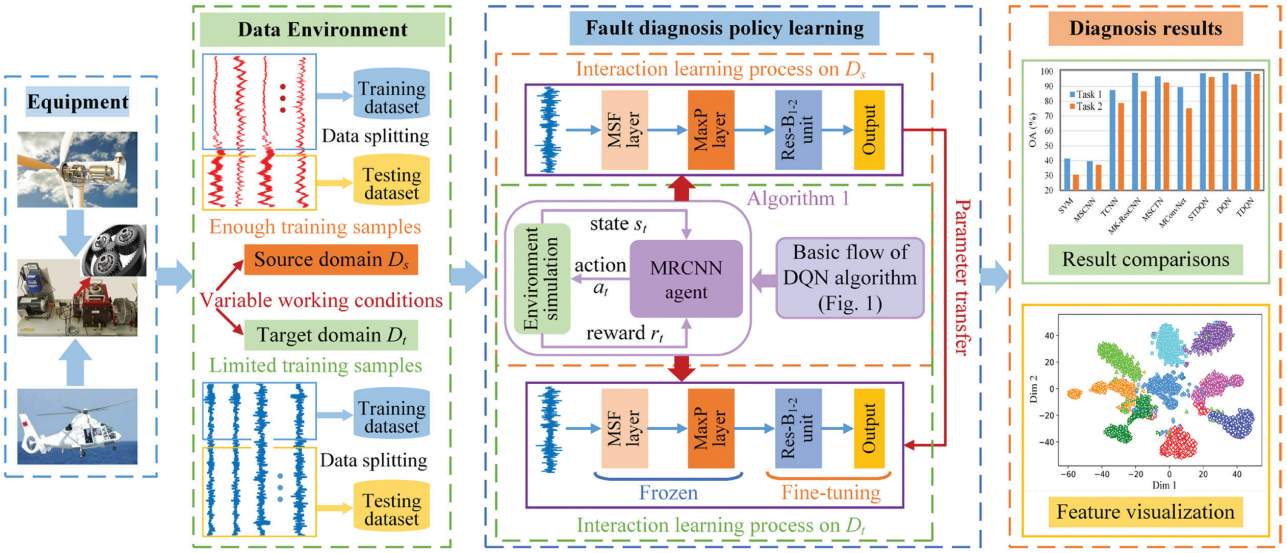

Fig. 3. The overall framework of TDQN for PGB fault diagnosis.

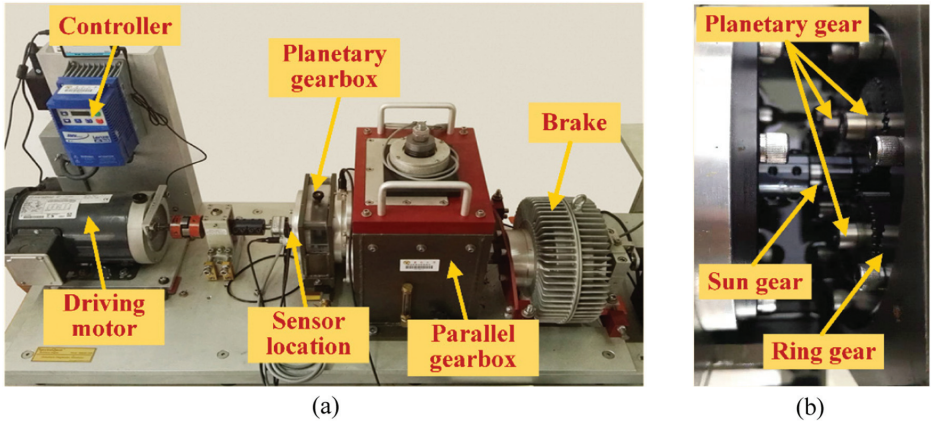

Fig. 4. Experimental system: (a) DDS platform; (b) PGB inner structure.

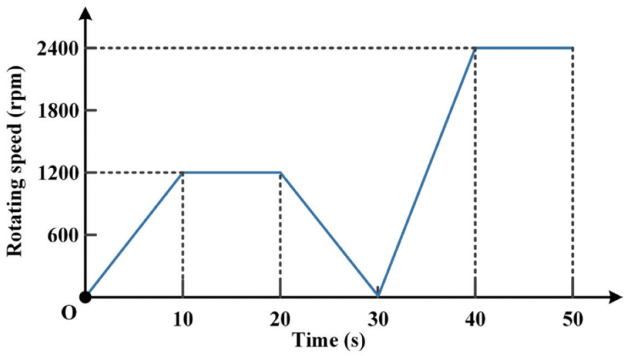

Fig. 5. The change of rotating speed.

The whole dataset D_s^1 is randomly and equally divided into five sub-sample sets, so each validation experiment has one sub-sample set containing 4590 samples for testing, while the remaining four sub-sample sets are utilized to train the model. The details of the source domain dataset are listed in Table I.

B. TARGET DOMAIN DATASETS

Target domain datasets are also gathered from the aforesaid DDS platform but in two different situations: one from the same speed but different loads between two cases and the other from two different speed-load cases, thereby creating two target sub-datasets D_t^1 and D_t^2, as illustrated in Table II.

Three running conditions on the DDS test rig are emulated by controlling the motor speed and brake: 1800 rpm-3.66 N.m, 1800 rpm-10.98 N.m, and 1200 rpm-0 N.m, respectively. Fault configurations and sensor settings are the same as the source domain. Vibration signals are segmented into samples in a non-overlapping way, and each sample comprises 2048 data points. Target sub-dataset D_t^1 is obtained from two working conditions with the same speed of 1800 r/min but differing loads of 3.66 N.m and 10.98 N.m. For the dataset D_t^1, each health type has 60 training samples and 500 testing samples, and they are acquired evenly from both running conditions. Similarly, a target dataset D_t^2 consists of two different speed loads, 1800 rpm-3.66 N.m and 1200 rpm-0 N.m. Its settings are similar to the dataset D_t^1. More details can be found in Table II.

Table I. Descriptions of the source domain dataset (D_s^1)

Type	Health description	Sample size	Speed/load (rpm/N.m)	Action label
HEA	Healthy PGB	2550	[0,2400]/0	0
CTF	Chipped tooth fault in gear	2550	[0,2400]/0	1
MTF	Missing tooth fault in gear	2550	[0,2400]/0	2
RCF	Root crack fault in gear	2550	[0,2400]/0	3
SWF	Surface wear fault in gear	2550	[0,2400]/0	4
BWF	Ball wear fault in bearing	2550	[0,2400]/0	5
CWF	Combo wear fault in bearing	2550	[0,2400]/0	6
IRF	Inner race fault in bearing	2550	[0,2400]/0	7
ORF	Outer race fault in bearing	2550	[0,2400]/0	8

Table II. Descriptions of the target domain datasets D_t^1 AND D_t^2

Dataset D_t	Type	Speed (rpm)	Load (N.m)	Training/testing sample size	Action label
D_t^1	HEA	1800	3.66 & 10.98	30 & 30/250 & 250	0
	CTF	1800	3.66 & 10.98	30 & 30/250 & 250	1
	MTF	1800	3.66 & 10.98	30 & 30/250 & 250	2
	RCF	1800	3.66 & 10.98	30 & 30/250 & 250	3
	SWF	1800	3.66 & 10.98	30 & 30/250 & 250	4
	BWF	1800	3.66 & 10.98	30 & 30/250 & 250	5
	CWF	1800	3.66 & 10.98	30 & 30/250 & 250	6
	IRF	1800	3.66 & 10.98	30 & 30/250 & 250	7
	ORF	1800	3.66 & 10.98	30 & 30/250 & 250	8
D_t^2	HEA	1800 & 1200	3.66 & 0	30 & 30/250 & 250	0
	CTF	1800 & 1200	3.66 & 0	30 & 30/250 & 250	1
	MTF	1800 & 1200	3.66 & 0	30 & 30/250 & 250	2
	RCF	1800 & 1200	3.66 & 0	30 & 30/250 & 250	3
	SWF	1800 & 1200	3.66 & 0	30 & 30/250 & 250	4
	BWF	1800 & 1200	3.66 & 0	30 & 30/250 & 250	5
	CWF	1800 & 1200	3.66 & 0	30 & 30/250 & 250	6
	IRF	1800 & 1200	3.66 & 0	30 & 30/250 & 250	7
	ORF	1800 & 1200	3.66 & 0	30 & 30/250 & 250	8

Further, the feasibility and transferability of the TDQN model are evaluated on the above source dataset D_s^1 and target datasets $D_t^1$and D_t^2. The implementation specifics of this method and certain comparison methods are given in the next section.

C. EXPERIMENTAL CONFIGURATIONS AND COMPARISONS

The parameter details of the TDQN method are set as follows.

In the first multiscale layer of MRCNN, three 1D convolutions with kernel sizes of 25×1, 64×1, and 112×1, respectively, are conducted on input vibration signal. Each branch of the MSF layer has 32 filters. The output channel of Res-$B_{1\text{-}2}$ units is 128 and 256, respectively. Detailed structure is depicted in Fig. 2. Notably, all convolutions use an exponential linear unit (ELU) activation function [29], and the convolutional filling way is "SAME." In the CMDP, the discount factor γ is set as 0.1 by considering the low correlation between samples, implying that the agent cares more about the immediate reward during autonomous learning. RAdam optimizer with a learning rate of 1e-3 is used to update Main-Net, and the delayed updating step of Target-Net is 100. The capacity of the experience replay buffer is 100000, and the interaction steps between the agent and environment are 140000 in the source domain. Besides, a linear annealing policy is used to update a greedy parameter ϵ from 1 to 0.1 for a total of 100000 steps, and the batch size is 128.

To validate the superiority of TDQN, it is compared with other methods that are run on the same datasets. The primary parameters of these methods are listed as follows:

- SVM [4]: 28 time and frequency domain features (*e.g.*, mean, crest factor, kurtosis, etc.) are extracted manually and SVM is used as a classifier.
- WDCNN [7]: The network is a slightly modified version of WDCNN, where the dropout technique is adopted in the final fully connected layer to avoid overfitting.
- TCNN [20]: A transferable CNN, its structure is a variant of WDCNN, and it uses a fine-tuning transfer strategy to retrain the model on the target domain.
- MK-ResCNN [9]: MK-ResCNN is a multi-branch multiscale kernel deep residual network; here its input

is samples reshaped to a 512×4, and other details are listed in [9].

- MSCNN [2]: MSCNN contains a multiscale coarse-grained layer and multi-branch structures. Also, a dropout operation is placed after the last fully connected layer.
- MSCTN [21]: A multiscale convolutional transfer network (MSCTN), which uses dilation convolution to learn multiscale features. Its input is samples reshaped to a 512×4 shape and a dropout operation is added after a GAP layer.
- MConvNet: Multiscale convolutional network (MConvNet) whose network structure is similar to TDQN except that the output layer activation function is Softmax, and the training way uses supervised learning.
- STDQN: Single-scale TDQN is similar to TDQN only saves the branch with a kernel size of 64 in the MSF layer first stage and replaces the second multiscale block of the MFS layer with a 3×1 convolution with a filter number of 64.
- DQN: DQN has the same network as TDQN, but it is trained from scratch in the target domain without fine-tuning.

When the above deep learning models are evaluated on the source domain dataset D_s^1, the batch size is also 128, and the same RAdam optimizer is used to update models. Additionally, 10% of training samples are randomly chosen as a validation dataset, and an early stopping operation is implemented to save optimal models to avoid overfitting.

All methods are coded using python in the Keras framework and the open-source library Keras-rl [30] and conducted on a computer with Intel® Xeon(R) CPU E5-2603 v4@1.70GHz × 12, four GTX 1080 Ti (11GB) graphics cards, and Ubuntu 16.04 operating system. Each experiment is carried out five times, and the overall accuracy (OA) is adopted as an assessment indicator. The average OA and standard deviation (STD) in source and target tasks are reported and compared with each other.

V. EXPERIMENTAL RESULTS AND COMPARISONS

In this section, the effectiveness of TDQN is first evaluated on a source domain (D_s). Then, this model is migrated to the target domain (D_t) and fine-tuned with small target training samples to assess the model's transferability. Lastly, the impact of training sample size on performance is discussed.

A. RESULTS ON SOURCE DOMAIN

To analyze the performance of these methods on the source dataset D_s^1, 5-fold cross-validation results including average accuracy and elapsed time are recorded in Table III. From the results, SVM has worse training and testing OA than others. It shows that features manually extracted are hard to describe fault characteristics. In contrast to single-scale WDCNN, TCNN, and STDQN, multiscale models have better performance. It means that multiscale models can extract rich features for improving accuracy and are more suitable for fault diagnosis in complex conditions. Especially, the testing accuracy of MConvNet, MK-ResCNN, and TDQN reaches 96.39%, 98.28%, and 98.53%, respectively. TDQN outperforms SVM, WDCNN, TCNN, etc., significantly. Compared with MK-ResCNN, this method still slightly surpasses it. MConvNet has relatively high accuracy, which further shows the effectiveness of the multiscale network. Importantly, compared with MConvNet and STDQN, the testing OA of TDQN increases by 2.14% and 0.62%, respectively. Since model interaction learning requires numerous trial-and-error steps and the updating process needs to store empirical trajectories and copy model parameters, DRL models often have longer elapsed time than models directly training. As a result, STDQN and TDQN take a long training time. TDQN, in particular, has the longest training time, reaching about 2862 s. Moreover, the network lightweight also affects model elapsed time. This is why TDQN is more time-consuming than STDQN. Even so, due to the relatively lightweight design, TDQN has a testing time of only 0.61 s, which is faster than the MK-ResCNN.

Figure 6 depicts a normalized 5-fold cross-validation confusion matrix for the proposed TDQN. From Fig. 6, this method has an accuracy of more than 97% for each health type. Especially for HEA and MTF states, the accuracies exceed 99%. It is inferred that this method has better fault diagnosis consistency. To graphically illustrate the model learning ability, the t-distributed stochastic neighbor embedding (t-SNE) method [31] is applied to reduce the feature dimension of high-level GAP layers of TDQN and MConvNet from 256 to 2 and visualize the learned features, respectively, as shown in Fig. 7. It can be seen that features learned from TDQN have stronger clustering and separability among nine health types than MConvNet.

Table III. Results of different methods on source dataset (D_s^1)

	Diagnostic accuracy (%)		Average time (s)	
Methods	**Training OA**	**Testing OA**	**Training time**	**Testing time**
SVM	50.09 ± 0.10	49.27 ± 0.36	41.09	7.46
WDCNN	93.37 ± 2.37	74.46 ± 0.98	107.82	0.65
TCNN	97.79 ± 0.43	90.90 ± 0.61	421.65	0.97
MK-ResCNN	100.00 ± 0.00	98.28 ± 0.21	1036.76	2.57
MSCNN	99.91 ± 0.08	91.31 ± 0.64	204.67	0.69
MSCTN	98.64 ± 0.73	93.84 ± 0.79	202.90	0.65
MConvNet	100.00 ± 0.00	96.39 ± 0.48	231.32	0.84
STDQN	99.91 ± 0.01	97.91 ± 0.19	2361.50	0.46
TDQN	99.96 ± 0.01	98.53 ± 0.26	2861.92	0.61

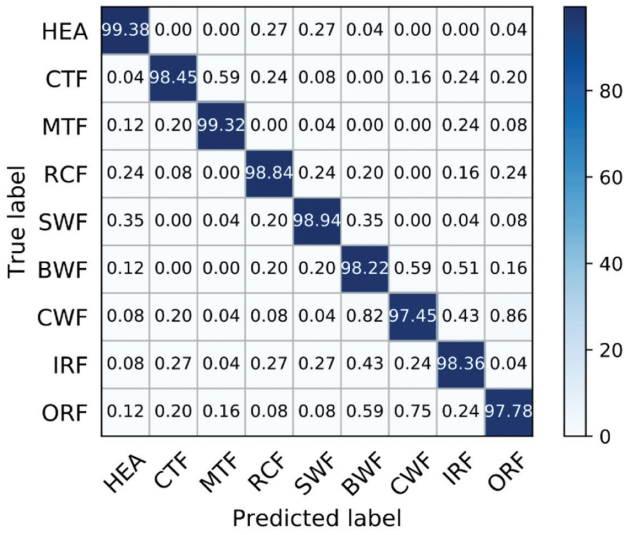

Fig. 6. Normalized confusion matrix on the source domain dataset.

In addition, a sample from the dataset D_s^1 is used to verify feature representations of the first multiscale layer of TDQN. Figure 8 depicts reshaped feature maps from three convolutional scales (*i.e.*, 25, 64, and 112).

From Fig. 8, different scale kernels can capture various time scale information, which is slightly similar to time-frequency representation. It can be seen along the vertical axis that small-scale kernels focus on more local details, and it can be seen along the horizontal axis that each wide-kernel filter easily captures more low-frequency feature information. Therefore, it can be concluded that the designed first multiscale module in TDQN can capture more underlying physics information directly from raw signals than a single-scale kernel, which lays the groundwork for high-level feature extraction in subsequent modules and is useful to increase diagnostic performance.

Based on the analysis above, results show that the presented TDQN can better learn meaningful discriminative features that can effectively enhance the fault diagnosis accuracy of PGB. The effectiveness and superiority of this method in the source domain dataset D_s^1 is verified by the analysis above.

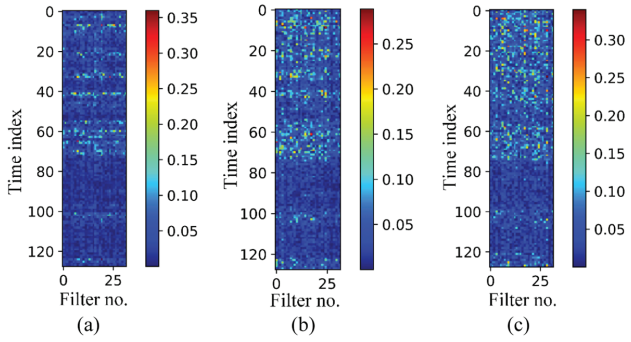

Fig. 8. Feature extractions of a vibration signal from three scales in Stage I: (a) Scale 1 for kernel 25; (b) Scale 2 for kernel 64; (c) Scale 3 for kernel 112.

B. TRANSFERABILITY EVALUATION ON TARGET DOMAIN

In this section, two target datasets D_t^1 and D_t^2, are employed to evaluate the transferability and viability of the proposed method under variable conditions with small target samples. The interaction steps in DRL models between the agent and the environment are 40000, and the step for updating parameter ϵ with a linear annealing policy from 1 to 0.1 is a total of 20,000. Due to small training samples in the target domain, the batch size for other DL models is 10, and the training epoch is 100. Besides them, the experimental settings are the same as in the preceding section. All methods were executed five times to obtain average evaluation results, as recorded in Table IV.

From Table IV, the accuracies of the proposed method on two target datasets are all greater than 95%. Concretely, the testing OA of TDQN for datasets D_t^1 and D_t^2 are 99.63% and 98.37%. TDQN can outperform others even with small target training samples. Among others, SVM performs worst. Or rather, handcrafted features are hard to ensure their consistency under variable conditions, resulting in inferior accuracy. Deep models all have high training OA of over 99%. Nevertheless, there are great differences among them in testing sets. Especially, MSCNN likewise performs badly, with a testing accuracy of less than 40%, because of a lack of adequate training data. Similarly, while

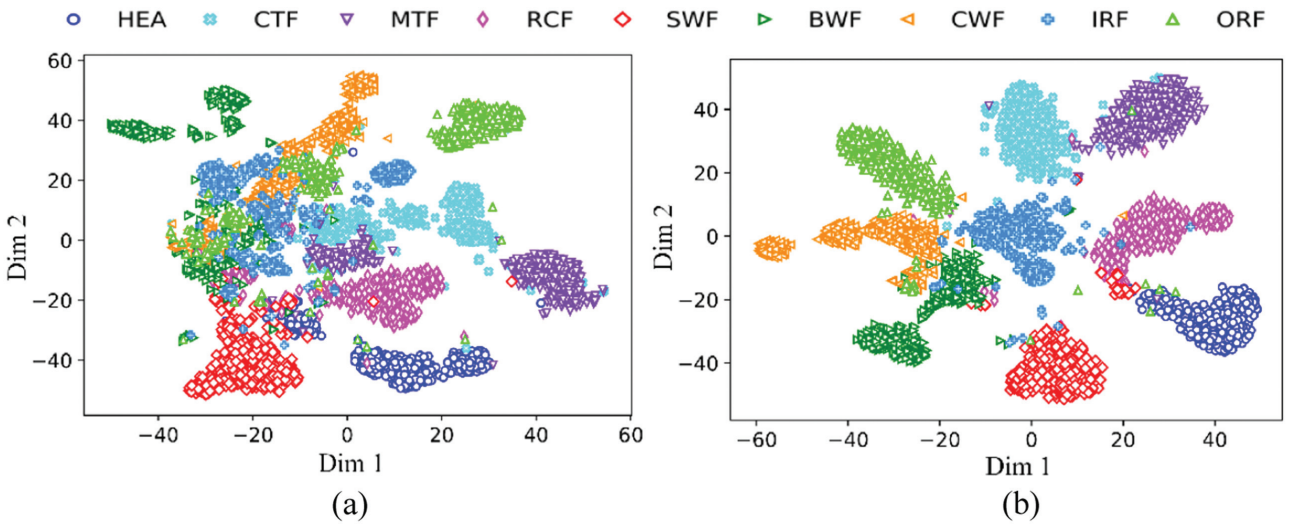

Fig. 7. High-layer feature visualization on the source domain: (a) MConvNet for D_s^1; (b) TDQN for D_s^1.

Table IV. Result comparisons on target domain (D_t^1 AND D_t^2)

	Dataset D_t^1 (%)		Dataset D_t^2 (%)	
Methods	**Training OA**	**Testing OA**	**Training OA**	**Testing OA**
SVM	44.81 ± 0.00	41.49 ± 0.00	40.56 ± 0.00	30.58 ± 0.00
MSCNN	99.92 ± 0.09	39.62 ± 5.45	99.85 ± 0.22	37.25 ± 0.86
TCNN	99.89 ± 0.09	87.55 ± 2.37	100.00 ± 0.00	78.75 ± 4.14
MK-ResCNN	100.00 ± 0.00	99.15 ± 0.27	99.96 ± 0.08	86.67 ± 6.40
MSCTN	99.96 ± 0.08	96.73 ± 1.00	100.00 ± 0.00	92.47 ± 1.05
MConvNet	100.00 ± 0.00	89.56 ± 1.80	100.00 ± 0.00	75.21 ± 1.11
STDQN	100.00 ± 0.00	98.78 ± 0.22	100.00 ± 0.00	96.22 ± 0.26
DQN	100.00 ± 0.00	99.18 ± 0.16	100.00 ± 0.00	91.19 ± 1.44
TDQN	100.00 ± 0.00	99.63 ± 0.06	100.00 ± 0.00	98.37 ± 0.20

MK-ResCNN and MConvNet outperform MSCNN, they are still inferior to TDQN. In contrast to single-scale TCNN, MSCTN performs better, with testing OA of 96.73% and 92.47% on target datasets D_t^1 and D_t^2, respectively. This is because that single-scale TCNN cannot well learn useful features from the source domain, resulting in low accuracy when the pre-trained model is transferred to target datasets. Other methods perform better than 90% in the target testing domain, especially for the testing OA of over 95% on the dataset D_t^1, which are better than SVM, TCNN, MConvNet, etc., but still worse than TDQN.

Specifically, the proposed TDQN outperforms STDQN and DQN significantly. Especially on the dataset D_t^2, the testing OA of TDQN rose by 2.15%, and 7.18%, respectively. Additionally, the performance of all methods on the target task D_t^2 is weaker than that on the target task D_t^1. It indicates that variable speed cases are more difficult to diagnose than variable load cases because failure frequency is changed. Despite this, TDQN still has an OA of 98.37%. To further demonstrate the superiority of TDQN, its high-level GAP features are visualized and compared with DQN, as depicted in Fig. 9.

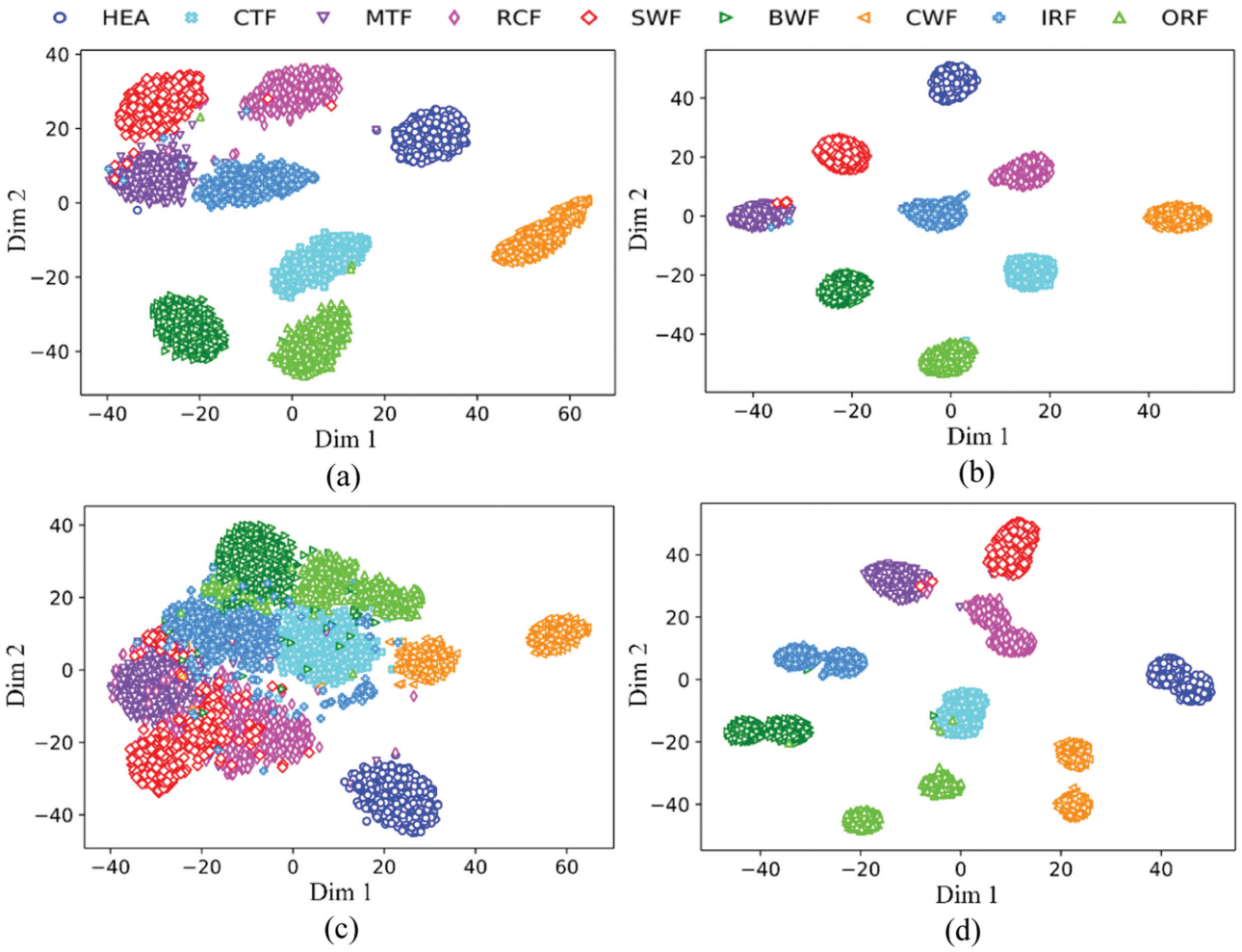

Fig. 9. High-layer feature visualization on target domain dataset: (a) DQN for D_t^1; (b) TDQN for D_t^1; (c) DQN for D_t^2; (d) TDQN for D_t^2.

From Fig. 9, feature clustering and separability of TDQN among nine classes outperform DQN considerably. Especially for D_t^2, most of the features from DQN are heavily mixed and only health types HEA and CWF are well distinguished. As a result, it is reasonable to explain that the performance of DQN is weaker on D_t^2. From Table IV, and Fig. 9(a) and (b), even the testing accuracy of DQN on the dataset D_t^1 is 99.18%, its discriminant features learned are worse than TDQN. From Fig. 9(d), while TDQN performs better, there is an evident intra-class gap between BWF, CWF, IRF, and ORF. This is since the target set D_t^2 contains two speed-load cases, existing in obvious diversities within the class. Importantly, while DQN requires more samples to learn from scratch to realize fault diagnosis, TDQN can achieve high accuracy with small training samples, greatly reducing computation costs. In sum, the transferability and effectiveness of the proposed approach have been demonstrated in the above datasets D_t^1 and D_t^2.

C. EFFECT OF TARGET TRAINING SET SIZE ON RESULTS

Furthermore, a comparison of five diagnosis methods (*i.e.*, TCNN, MSCTN, DQN, STDQN, and TDQN) under different numbers of target training samples is carried out. Experimental configurations are as same as in the previous section B. Detailed comparisons are presented in Fig. 10.

From Fig. 10(a), the testing OA of TDQN reaches 98.20% even with only 10 target training samples, and STDQN with the best performance is only 94.23% among comparison methods. DQN, in particular, performs worst, with only 65.30% accuracy. The advantages of TDQN in the case of small target samples are further verified. Also, it can be seen that these methods with a model transfer perform better. The accuracy of these methods gradually improves as the number of training samples grows. When the training sample reaches 20, the proposed method can achieve a 99.07% recognition rate on the dataset D_t^1. At the same time, the accuracies of TCNN, MSCTN, DQN, and STDQN are 85.58%, 92.97%, 89.00%, and 97.09%, respectively. When the quantity of training samples rises to 40, the test accuracy of DQN reaching 98.11% exceeds that of TCNN and MSCTN. It further shows the superiority of the proposed method, especially TDQN. Among them, the accuracy improvement of single-scale TCNN is relatively slowest as the number of target training samples increases. It may be that TCNN has a relatively great complexity, which results in overfitting in the event of small training data. Meanwhile, because the single-scale structure is difficult to capture ample features in the complex source domain, the transfer effect of the model becomes worse.

Similarly, these conclusions are inferred from Fig. 10(b). Compared with the target dataset D_t^1, these phenomena are more obvious in the target dataset D_t^2. When the training sample size is set to 20, for example, TDQN achieves an accuracy of 96.18%, which is 18.62%, 9.18%, 23.79%, and 5.13% higher than TCNN, MSCTN, DQN, STDQN, and TDQN, respectively. Furthermore, TDQN has a low standard deviation, indicating strong stability. To sum up, in the case of small training samples, the TDQN outperforms significantly others. Therefore, TDQN has the huge potential to provide a generic and reliable diagnostic framework for PGB fault identification.

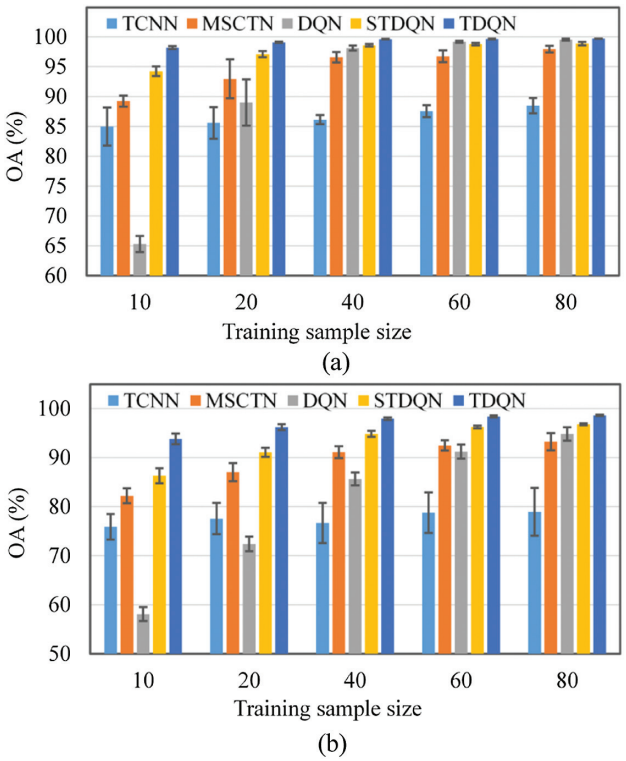

Fig. 10. Comparisons of five methods in different training sample sizes: (a) OA comparisons on D_t^1; (b) OA comparisons on D_t^2.

VI. CONCLUSION

This work proposes a transferable DRL framework based on TDQN for PGB fault diagnosis under variable conditions with small training data. It first creates an MRCNN structure as a DRL agent to enhance feature learning ability while avoiding model degradation. This agent can autonomously and effectively learn rich features directly from vibration signals for fault recognition via weak feedback interaction learning, improving the generalization performance. Importantly, this method combines parameter TL, which utilizes knowledge obtained from a source domain to enhance the accuracy of the DRL model under variable conditions with small training data. The results show that this method is more impactful and superior to existing methods such as SVM, TCNN, MSCTN, and others.

In future work, we will refine this method by optimizing agent structure from a physics-informed perspective, reward reshaping, and conjoining other TL strategies to make it suited for more challenging scenarios (*e.g.*, strong noise and data imbalance). Also, we will investigate its scalability on other equipment to enlarge the potential applications.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (52275130) and the National Key Research and Development Program of China (2018YFB1702400).

CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

References

[1] M. Kordestani, M. Saif, M. E. Orchard, R. Razavi-Far, and K. Khorasani, "Failure prognosis and applications—a survey of recent literature," *IEEE Trans. Rel.*, vol. 70, no. 2, pp. 728–748, Jun. 2021.

[2] G. Jiang, H. He, J. Yan, and P. Xie, "Multiscale convolutional neural networks for fault diagnosis of wind turbine gearbox," *IEEE Trans. Ind. Electron.*, vol. 65, no. 4, pp. 3196–3207, Apr. 2019.

[3] T. Wang, Q. Han, F. Chu, and Z. Feng, "Vibration based condition monitoring and fault diagnosis of wind turbine planetary gearbox: a review," *Mech. Syst. Signal Process.*, vol. 126, pp. 662–685, Jul. 2019.

[4] R. Liu, B. Yang, E. Zio, and X. Chen, "Artificial intelligence for fault diagnosis of rotating machinery: a review," *Mech. Syst. Signal Process.*, vol. 108, pp. 33–47, Aug. 2018.

[5] S. Xing, Y. Lei, S. Wang, and F. Jia, "Distribution-invariant deep belief network for intelligent fault diagnosis of machines under new working conditions," *IEEE Trans. Ind. Electron.*, vol. 68, no. 3, pp. 2617–2625, Mar. 2021.

[6] K. N. Ravikumar, A. Yadav, H. Kumar, K. V. Gangadharan, and A. V. Narasimhadhan, "Gearbox fault diagnosis based on multi-scale deep residual learning and stacked LSTM model," *Measurement*, vol. 186, pp. 1–13, Dec. 2021.

[7] W. Zhang, G. Peng, C. Li, Y. Chen, and Z. Zhang, "A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals," *Sensors*, vol. 17, no. 2, pp. 425, Jan. 2017.

[8] M. Azamfar, J. Singh, I. Bravo-Imaz, and J. Lee, "Multisensor data fusion for gearbox fault diagnosis using 2-D convolutional neural network and motor current signature analysis," *Mech. Syst. Signal Process.*, vol. 144, Oct. 2020, Art. no. 106861.

[9] R. Liu, F. Wang, B. Yang, and S. J. Qin, "Multiscale kernel based residual convolutional neural network for motor fault diagnosis under non-stationary conditions," *IEEE Trans. Ind. Inform.*, vol. 16, no. 6, pp. 3797–3806, Jun. 2020.

[10] Y. Ding, L. Ma, J. Ma, M. Suo, L. Tao, Y. Cheng, and C. Lu, "Intelligent fault diagnosis for rotating machinery using deep Q-network based health state classification: a deep reinforcement learning approach," *Adv. Eng. Inform.*, vol. 42, Oct. 2019, Art. no. 100977.

[11] G. Li, J. Wu, C. Deng, X. Xu, and X. Shao, "Deep reinforcement learning-based online domain adaptation method for fault diagnosis of rotating machinery," *IEEE/ASME Trans. Mechatron.*, vol. 27, no. 5, pp. 2796–2805, Oct. 2022.

[12] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–541, Feb. 2015.

[13] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[14] B. R. Kiran *et al.*, "Deep reinforcement learning for autonomous driving: a survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp.4909–4926, Jun. 2022.

[15] H. Wang, J. Xu, C. Sun, R. Yan, and X. Chen, "Intelligent fault diagnosis for planetary gearbox using time-frequency representation and deep reinforcement learning," *IEEE/ASME Trans. Mechatron.*, vol. 27, no. 2, pp. 985–998, Apr. 2022.

[16] Z. Wang and J. Xuan, "Intelligent fault recognition framework by using deep reinforcement learning with one dimension convolution and improved actor-critic algorithm,". *Adv. Eng. Inform.*, vol. 49, Aug. 2021, Art. no. 101315.

[17] M. Azamfar, J. Singh, X. Li, and J. Lee, "Cross-domain gearbox diagnostics under variable working conditions with deep convolutional transfer learning," *J. Vib. Control*, vol. 27, no. 7–8, pp. 854–864, Apr. 2021.

[18] S. Shao, S. McAleer, R. Yan, and P. Baldi, "Highly accurate machine fault diagnosis using deep transfer learning," *IEEE Trans. Ind. Inform.*, vol. 15, no. 4, pp. 2446–2455, Apr. 2019.

[19] Z. He, H. Shao, P. Wang, J. J. Lin, J. Cheng, and Y. Yang, "Deep transfer multi-wavelet auto-encoder for intelligent fault diagnosis of gearbox with few target training samples," *Knowl.-Based Syst.*, vol. 191, Mar. 2020, Art. no. 105313.

[20] Z. Chen, K. Gryllias, and W. Li, "Intelligent fault diagnosis for rotary machinery using transferable convolutional neural network," *IEEE Trans. Ind. Inform.*, vol. 16, no. 1, pp. 339–349, Jan. 2020.

[21] B. Zhao, X. Zhang, Z. Zhan, and S. Pang, "Deep multi-scale convolutional transfer learning network: a novel method for intelligent fault diagnosis of rolling bearings under variable working conditions and domains," *Neurocomputing*, vol. 407, pp. 24–38, Sep. 2020.

[22] Y. Tamaazousti, H. Le Borgne, C. Hudelot, M.-E.-A. Seddik, and M. Tamaazousti, "Learning more universal representations for transfer-learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 9, pp. 2212–2224, Sep. 2020.

[23] E. Lin, Q. Chen, and X. Qi, "Deep reinforcement learning for imbalanced classification," *Appl. Intell.*, vol. 50, pp. 2488–2502, Mar. 2020.

[24] H. Wang, J. Xu, and R. Yan, "Multi-scale attention based deep reinforcement learning for intelligent fault diagnosis of planetary gearbox," *J. Mech. Eng.*, vol. 58, no.11, pp.133–142, Jun. 2022.

[25] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML-10)*, 2010, pp. 807–814.

[26] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[27] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5353–5360.

[28] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," *arXiv: 1908.03265*, Aug. 2019.

[29] D. A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," arXiv:1511.07289, Nov. 2015.

[30] M. Plappert, "Keras-rl," *GitHub Reposit.*, 2016. [Online]. Available: https://github.com/keras-rl/keras-rl.

[31] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learning Res.*, vol. 9, pp. 2579–2605, Nov. 2008.