scientific reports

Check for updates

OPEN

# Lightweight bearing fault diagnosis via decoupled distillation and low rank adaptation

Ovanes Petrosian[1,2], Pengyi Li[1,2]✉, Yulong He[1], Jiarui Liu[1], Zhaoruikun Sun[1], Guofeng Fu[1] & Liping Meng[1]

**Rolling bearing fault detection has developed rapidly in the field of fault diagnosis technology, and it occupies a very important position in this field. Deep learning-based bearing fault diagnosis models have achieved significant success. At the same time, with the continuous improvement of new signal processing technologies such as Fourier transform, wavelet transform and empirical mode decomposition, the fault diagnosis technology of rolling bearings has also been greatly developed, and it can be said that it has entered a new research stage. However, most of the existing methods are limited to varying degrees in the industrial field. The main ones are fast feature extraction and computational complexity. The key to this paper is to propose a lightweight bearing fault diagnosis model DKDL-Net to solve these challenges. The model is trained on the CWRU data set by decoupling knowledge distillation and low rank adaptive fine tuning. Specifically, we built and trained a teacher model based on a 6-layer neural network with 69,626 trainable parameters, and on this basis, using decoupling knowledge distillation (DKD) and Low-Rank adaptive (LoRA) fine-tuning, we trained the student sag model DKDL-Net, which has only 6838 parameters. Experiments show that DKDL-Net achieves 99.48% accuracy in computational complexity on the test set while maintaining model performance, which is 0.58% higher than the state-of-the-art (SOTA) model, and our model has lower parameters.**

**Keywords** Bearing fault diagnosis, Convolutional neural network, Model compression, Low-Rank Adaptation

Rolling bearings are common components in rotating machinery, designed to reduce friction during rotation and thereby enhance the safety of the equipment. Studies in the industrial sector indicate that approximately 40%-70% of mechanical failures are caused by bearing faults[1–4]. However, traditional bearing fault detection methods are time-consuming. In the current era of artificial intelligence, it is crucial to use deep learning techniques for fault detection tasks. Training an efficient and accurate bearing fault detection model can not only improve detection efficiency but also significantly reduce economic losses[5].

Bearing fault diagnosis is typically based on acoustic signals and bearing vibration signals[6]. Bearing faults cause abnormal vibrations and sounds, making it possible to detect faults by diagnosing these anomalies. In practical detection tasks, sensors are usually installed on machine tools to capture the bearing's sound signals.

There are various approaches to vibration fault detection based on sound signals and bearings, including methods using deep belief networks (DBN)[7–9], support vector machine(SVM)[10], convolutional neural network (CNN)[11–14], deep autoencoders (DAE)[15–17], generative adversarial networks (GAN)[18–20], and deep transfer learning (DTL)[21–24]. What's common among these models is that they differ only in the way neural networks are constructed and training strategies, while the data type remains the same. Additionally, they all require a large amount of data.

Over the past 5 years, bearing fault detection tasks have commonly used models such as Convolutional Neural Networks (CNN)[11–13], Recurrent Neural Networks (RNN)[25–27], and Long Short-Term Memory Networks (LSTM)[28–31]. Experiments have shown that lightweight models fail to achieve the desired results, with the number of model parameters affecting the final outcome. For instance, LEFE-Net[32], WDCNN[33], and MCNN-LSTM[31] models can achieve an accuracy of over 98.50% on the CWRU[34,35] dataset, but their trainable parameters exceed 50,000. In contrast, the CLFormer[36] model has 4,980 parameters but an accuracy of less than 95%.

[1]St.Petersburg State University, 7-9 Universitetskaya Embankment, 199034 St Petersburg, Russia. [2]Ovanes Petrosian and Li Pengyi: These authors contributed equally to this work. ✉email: st112719@student.spbu.ru

Heavyweight models are usually able to achieve higher accuracy due to having more parameters, benefiting from their stronger fitting ability. However, this also comes with a higher inference overhead. In contrast, lightweight models, although slightly lacking in accuracy, significantly reduce inference costs and are suitable for resource-constrained application scenarios. The key to whether a model can be applied in industry lies in its lightweight nature, high accuracy, and robustness.

Models based on acoustic signals typically transform time-domain signals into frequency-domain signals using techniques such as Short-Time Fourier Transform (STFT)[37], Empirical Mode Decomposition (EMD)[38,39], and envelope spectrum analysis. Neural network models are then built to extract features, and finally, classification is performed through fully connected layers to achieve bearing fault diagnosis. Based on this process, we have constructed a lightweight bearing fault detection model.

In this paper, first, we trained a Teacher model with a large number of parameters, and then trained a lightweight Student model using DKD[40]. This Student model has only one convolutional layer, one pooling layer, and one fully connected layer. The Student model, guided by the Teacher model during training, has a significantly lower number of parameters. However, we found that the accuracy of the Student model was 2% lower than that of the Teacher model. To address this, we introduced a Low-Rank Adaptation (LoRA)[41] to fine-tune the Student model, which improved accuracy by 1.5% with a relatively short training time. Compared to traditional knowledge distillation and fine-tuning methods, the combination of DKD and LoRA fine-tuning ensures model performance while significantly reducing the number of training parameters.

We propose a model for industrial application in rolling bearing fault detection and contributions are fourfold:

- We developed a lightweight rolling bearing fault detection model based on DKD, characterized by a single-layer neural network, It is compressed by 90.20% compared to the teacher model (6-layer neural network).
- We improved the model's performance after knowledge distillation by using a LoRA fine-tuning method, addressing the performance degradation issue.
- Compared to other lightweight models, our approach demonstrates superior performance on the CWRU dataset, with average accuracy, precision, recall, and F1-Score all higher than those of other models. Our F1-Score reached 99.50%, and the trainable parameters of our model are only 6,838.
- Compared to the SOTA model our model improved the F1-Score by 0.58%.

## Related work

**Bearing fault detection.** The fault detection of rolling bearings typically involves checking whether the three components of the bearing (outer race, inner race, rolling elements) are faulty. The structure of the bearing is visualized as shown in Fig. 1. There are 10 categories of bearing sounds[42], where one category represents health components, three categories represent damaged rolling elements, three categories represent damaged inner races (IR), and three categories represent damaged outer races (OR). Once abnormal vibrations occur in the bearing, it indicates a fault.

**Convolutional neural network.** There has been extensive work on bearing fault detection models based on convolutional neural networks. The Adaptive Deep Convolutional Neural Network (ADCNN)[43] utilizes the distribution of key information in discrete frequency bands to diagnose the health status of rotating components. The 2D LeNet-5[44] is an improved bearing fault detection model based on LeNet, which performs one-dimensional convolution and pooling operations directly on the raw vibration signals without any

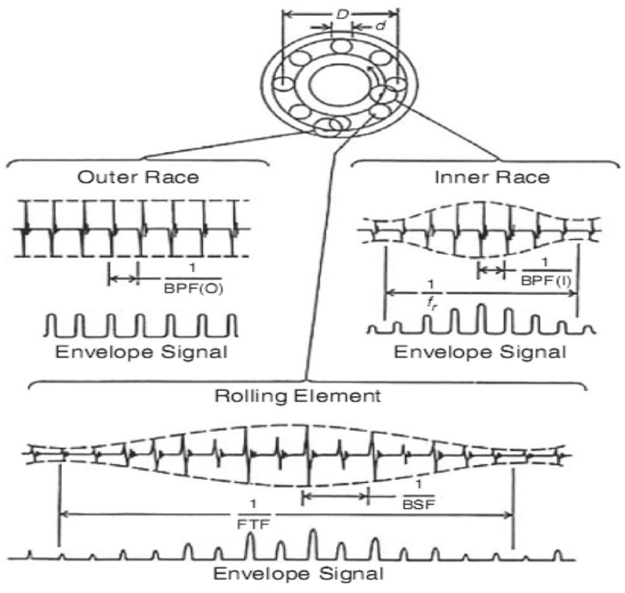


**Fig. 1**. Structure of rolling bearings.

preprocessing. The lightweight multi-scale CNN[45] introduces depthwise separable convolution in a multi-scale CNN to reduce the model's storage and computational costs. The hybrid deep neural network model integrated with Principal Component Analysis (PCA)[46] based model combines RNN and CNN, using bidirectional long short-term memory (BiLSTM) to enhance the extraction of time-series data features, and finally employs an Attention mechanism to improve the model's accuracy. KDSCNN[47] is a CNN model based on KD and model parameters are only 5890. BearingPGA-Net[48] is a lightweight model trained with DKD and applied on an FPGA module. MCNN-LSTM[31] is a model based on CNN and LSTM, which first uses CNN to extract features and then processes classification according to the time series information from LSTM. FaultNet[49] and WDCNN[33] are CNN models based on convolutional kernels of different scales.

While these models have achieved commendable performance, none of them offers a low-parameter model that combines high efficiency and accuracy.

**Model compression.** Neural network compression involves various techniques, including model pruning, parameter quantization, low-rank decomposition, knowledge distillation (KD), and lightweight model design[50]. Model pruning[51] can reduce the accuracy and lead to irregular structures. Parameter quantization[51] requires extensive training and fine-tuning. Low-rank decomposition[52] involves decomposing and training layer-by-layer parameters, which can be challenging for highly complex models. KD[53] is a method where a teacher model guides the training of a student model. The teacher model is typically complex, while the student model can have a simple structure, such as a single-layer network. This method achieves significant compression with minimal accuracy loss. Lightweight model design is challenging to combine with other compression methods[54]. If improperly combined, it can lead to lower performance and poor generalization.

All in all, KD, on the other hand, can significantly reduce model parameters while maintaining high accuracy. In fact, the algorithm mentioned in this article shows an accuracy drop of less than 2%.

DKD[40] is an algorithm that addresses the inefficiency problem of traditional knowledge distillation. It enhances the knowledge distillation process by decomposing and separately optimizing different components of the knowledge transfer from the teacher model to the student model. DKD divides traditional knowledge distillation into two independent parts: Target Class Knowledge Distillation (TCKD) and Non-Target Class Knowledge Distillation (NCKD), and defines separate loss functions to optimize the model. DKD is already a mature algorithm and has been widely applied in various tasks such as large language models LLMs[55], LVM[56], and object detection[57]. It is highly efficient for model compression.

**Fine-tuning.** LoRA was first proposed for fine-tuning LLMs[41] for downstream tasks, such as fine-tuning GPT for grammar correction[58], fine-tuning Llama for fire safety training[59], and fine-tuning Stable Diffusion to enhance image generation for specific tasks[60]. LoRA achieves this by decomposing the weight matrices into low-rank matrices. This approach significantly reduces the number of parameters that need to be fine-tuned, decreases the storage requirements of the model, and lowers the computational complexity, while enabling fast learning with less data[61]. As a result, both inference and training become more efficient.

In CNNs, LoRA can be used to compress the model by applying low-rank decomposition to the convolutional kernels[62], thereby reducing the number of parameters to be trained. This approach reduces the model's storage and computational costs. After fine-tuning a CNN model with LoRA, both inference speed and training time are greatly improved. Using low-rank decomposition for object detection[63] enhances the model's performance while reducing its parameters. Compared to the original CNN model, a LoRA fine-tuned model can operate under lower hardware resource conditions.

Our idea is to propose a model based on DKD training and LoRA fine-tuning. And model is not only fast in reasoning but also possesses high accuracy.

## Method

### CNN and classification

A CNN consists of $n$ $(n \in \{0, 1, 2, \ldots, n\})$ convolutional layers and $m$ $(m \in \{0, 1, 2, \ldots, m\})$ pooling layers. Convolutional layers use multiple filters to convolve over feature maps, extracting features. The extracted features are then passed through activation functions for non-linear transformations, helping to mitigate issues like vanishing gradients during training. Pooling layers are used to reduce the number of features, thus decreasing computational complexity.

The output value $a_j^l$ of the $j-th$ unit of the convolutional layer $l$ is given by Eq. (1).

$$a_j^l = f(b_j^l + \sum_{i \in M_j^l} a_i^{l-1} * k_{ij}^l) \tag{1}$$

The activation value $a_j^l$ in the pooling layer $l$ is given by Eq. (2).

$$a_j^l = f(b_j^l + \beta_j^l down(a_j^{l-1}, M^l)) \tag{2}$$

where $down(*)$ represents the pooling function. Common pooling functions include average pooling, max pooling, min pooling, and stochastic pooling. $b_j^l$ denotes the bias, $\beta_j^l$ represents the multiplier residual, and $M^l$ is the size of the pooling window used in the l-th layer.

After the model extracts features through convolutional layers, it needs to classify the feature data. Generally, classification tasks uses a cross-entropy (CE)[64] loss as a objective function, the expression for CE is as in Eq. (3). By optimizing the cross-entropy loss, the model's classification accuracy is improved.

$$L_{CE}=\sum_{n=1}^{N}-y_n log(p_n),\ p_n=\frac{e^{z_n}}{\sum_{i=1}^{N}e^{z_i}} \tag{3}$$

where $N$ denotes the number of categories, $y_n$ denotes true label, $log(p_n)$ denotes natural logarithm of the predict probability of n-th label from model, $z_n$ is raw scores from model output of n-th label, $p_n$ predicted probability for the n-th label.

## Decoupled knowledge distillation (DKD)

Although there are various types of knowledge distillation, we use DKD for improvement our model. First, in the classical KD, the *logit*, $l_i$, computed for each class is converted to a probability $p_i$ by using the $softmax(*)$ function as shown in Eq. (4).

$$p_i=\frac{\exp(l_i)}{\sum_{j=1}^{N}\exp(l_j)} \tag{4}$$

where $N$ denotes the number of classes.

$$b=[p_t,p_{\neg t}]\in\mathbb{R}^{1\times 2} \tag{5}$$

Then, we use binary probabilities $b=[p_t,p_{\neg t}]\in\mathbb{R}^{1\times 2}$ to distinguish between predictions related and unrelated to the target class, where $p_t$ denotes the target class and $p_{\neg t}$ denotes the non-target class, calculated as shown Eq. (6).

$$p_t=\frac{\exp(l_t)}{\sum_{j=1}^{N}\exp(l_j)},p_{-t}=\frac{\sum_{d=1,d\neq t}^{N}\exp(l_d)}{\sum_{j=1}^{N}\exp(l_j)}. \tag{6}$$

Meanwhile, we use $\tilde{p}_i=\frac{p_i}{p_t}$ to denote the probability between non-target categories (i.e., without considering the target category t) calculated as Eq. (7).

$$\tilde{p}_i=\frac{\exp(l_i)}{\sum_{j=1,j\neq i}^{N}\exp(l_j)} \tag{7}$$

Classical KD uses KL-Divergence as the loss function, and further, we re-represent KD using the binary probability $b$ and the non-target class $\tilde{p}$, T and S stand for teacher and student, respectively. represented as Eq. (8).

$$\text{KD}=\text{KL}(\text{p}^T\parallel\text{p}^S)=p_t^T\log\left(\frac{p_t^T}{p_t^S}\right)+\sum_{i=1,i\neq t}^{N}p_i^T\log\left(\frac{p_i^T}{p_i^S}\right) \tag{8}$$

Simplifying, we can rewrite Eq. (8) as Eq. (9) X by Eq. (4) and Eq. (7).

$$\begin{aligned}\text{KD}=&p_t^T\log\left(\frac{p_t^T}{p_t^S}\right)+p_{\neg t}^T\sum_{i=1,i\neq t}^{D}\tilde{p}_i^T\left(\log\left(\frac{\tilde{p}_i^T}{\tilde{p}_i^S}\right)+\log\left(\frac{p_{-t}^T}{p_{-t}^S}\right)\right)\\=&p_t^T\log\left(\frac{p_t^T}{p_t^S}\right)+p_{\neg t}^T\log\left(\frac{p_{-t}^T}{p_{-t}^S}\right)+p_{\neg t}^T\sum_{i=1,i\neq t}^{D}\tilde{p}_i^T\log\left(\frac{\tilde{p}_i^T}{\tilde{p}_i^S}\right)\end{aligned} \tag{9}$$

Simplifying, KD can then be rewrite as Eq. (10).

$$\text{KD}=\text{KL}(b^T\parallel b^S)+(1-p_t^T)\text{KL}(\tilde{p}^T\parallel\tilde{p}^S) \tag{10}$$

where $\text{KL}(b^T\parallel b^S)$ denotes the similarity between teacher and student probabilities in the target class, which we refer to as Target Class Knowledge Distillation (TCKD), and $\text{KL}(\tilde{p}^T\parallel\tilde{p}^S)$ denotes the similarity between teacher and student probabilities in the non-target class, which we refer to as Non-Target Class Knowledge Distillation (NCKD), and thus we can rewrite KD Eq. (11).

$$\text{KD}=\text{TCKD}+(1-p_t^T)\text{NCKD} \tag{11}$$

Observing the latest KD formulation, we find that on the one hand NCKD is coupled with $(1-p_t^T)$, which would suppress NCKD for well-predicted samples. On the other hand, the weights of NCKD and TCKD are coupled in the classical KD framework, which does not allow to change the weights of each term in order to balance the importance. Therefore DKD introduces two hyperparameters $\alpha$ and $\beta$ as weights for TCKD and NCKD, respectively. Thus the loss function of DKD can be written as Eq. (12).

$$\mathscr{L}_{DKD} = \alpha * \text{TCKD} + \beta * \text{NCKD} \tag{12}$$

where $\beta$ replaces $(1 - p_t^T)$ to prevent inhibiting the effectiveness of the NCKD, and secondly, $\alpha$ and $\beta$ can be allowed to be adjusted to achieve a balance of importance. By optimizing this decoupling loss, the knowledge gained by the teacher model is more easily transferred to the student model, thus improving the performance of the student network.

## Low-rank adaptation (LoRA)

LoRA is a method for efficiently fine-tuning pre-trained models on specific tasks. This algorithm reduces the fine-tuning parameters using a low-rank approach while enhancing the model's performance on the given task. Fine-tuning with fewer parameters can achieve over 90% of the performance of full fine-tuning. For a pre-trained model with a weight parameter matrix $W_0 \in \mathbb{R}^{d\times k}$, $\Delta W \in \mathbb{R}^{d\times k}$ represents the fine-tuning parameters for a specific task. $\Delta W$ is a lower-dimensional parameter matrix that can be expressed as $B \times A$, where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, with $r \ll k$. Thus, the parameter count of $\Delta W$ is smaller than that of $W_0$. The LoRA algorithm as show in the Eq. (13), with its key idea being to decompose the parameter matrix using a low-rank matrix decomposition.

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{13}$$

## Our DKDL-Net model

The DKDL-Net model is based on the DKD approach, where a Teacher model guides the training of a Student model. The model framework is illustrated in Fig. 2. The Teacher model is a large-scale model with a substantial number of parameters, and its increased depth enhances accuracy in bearing fault detection. However, the large parameter size of the Teacher model results in slow inference speed, making it unsuitable for efficient industrial tasks. Therefore, we trained the Student model using the DKD method. This model is a single-layer neural network, meaning it has fewer parameters and faster inference speed. However, since the Student model is derived from significant parameter compression, its accuracy decreases. In simple experimental analyses of bearing fault detection, the Student model's accuracy is approximately 2% lower compared to the Teacher model.

Through extensive research, we found that we can further fine-tune the Student model using the LoRA approach. The model framework is illustrated in Fig. 3. Typically, LoRA involves low-rank decomposition of the model's convolutional and fully connected layers. In this task, we cannot reduce the model's parameters further, as experiments have shown that this would decrease the model's accuracy. Therefore, we integrated the LoRA module into the single-layer network.

In the DKDL-Net network, we copied the parameters from the Student model. The parameters of the $A$ matrix in the LoRA module are initialized with data following a normal distribution $N(0, \sigma^2)$, while the $B$ matrix is initialized to 0. This approach allows us to enhance the model's accuracy by adding only a small number of parameters.

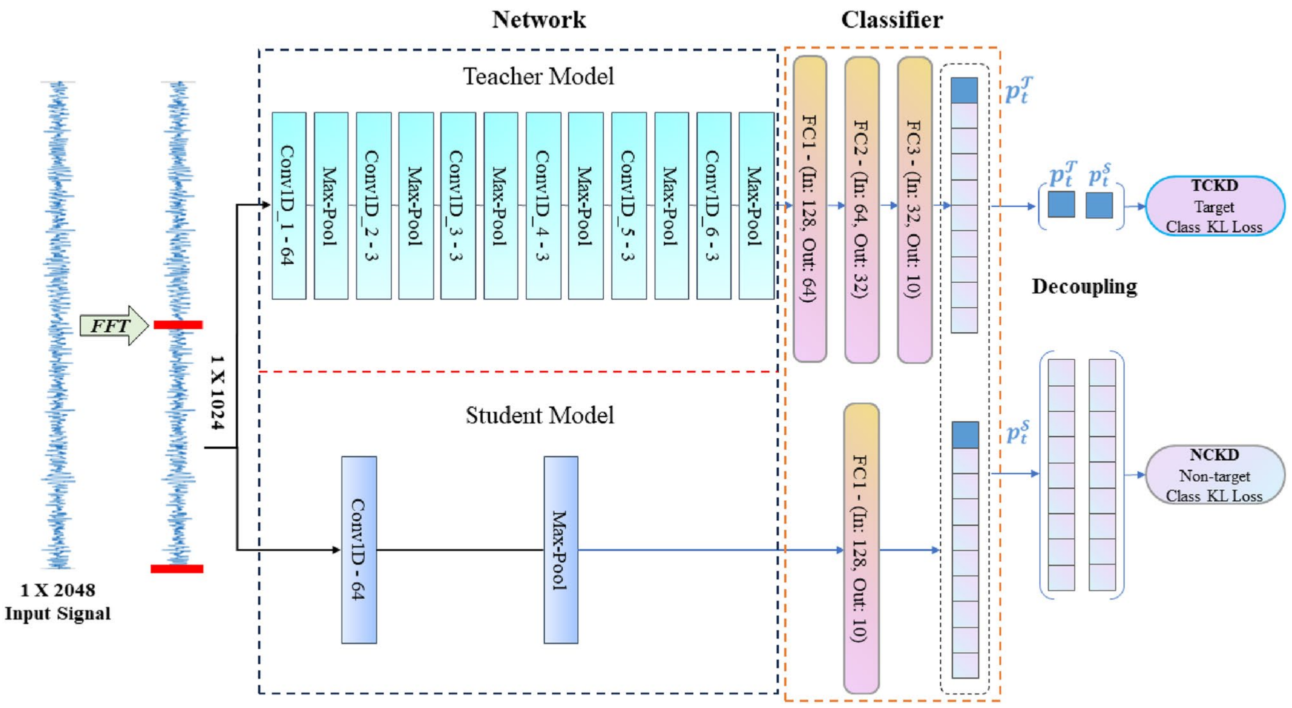


**Fig. 2**. Architecture of the Model. (top) Teacher Model, (below) Student Model.

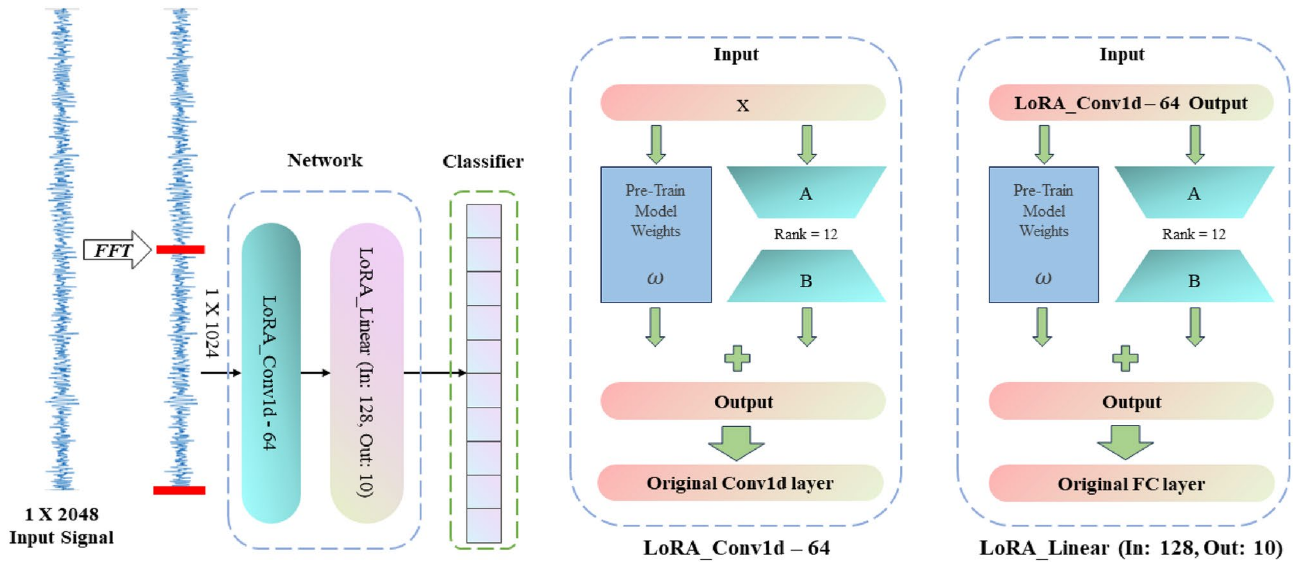


**Fig. 3**. Architecture of the DKDL-Net Model.

Regarding the training of the model, we utilize CE Loss as the objective function for training the Teacher model, defined mathematically as Eq. (3). For training the Student model, we employ a combination of TCKD Loss, NCKD Loss, and CE Loss as the loss functions. It is essential to balance CE Loss and DKD Loss during the training process, where DKD Loss is the sum of TCKD Loss, CE Loss and NCKD Loss, defined mathematically as Eq. (14).

$$\mathscr{L} = (1-\gamma)\underbrace{L_{CE}}_{\text{CE loss}} + \gamma\underbrace{\left(\alpha * \underbrace{\text{TCKD}}_{\text{TCKD Loss}} + \beta * \underbrace{\text{NCKD}}_{\text{NCKD Loss}}\right)}_{\text{DKD Loss}} \tag{14}$$

where $\gamma$ is a learnable parameter to balance CE and DKD Loss.

Finally, incorporating the LoRA plug-and-play module into the Student model, we fine-tune it using the CWRU dataset with CE Loss as the loss function, mathematically defined as Eq. (3). The pseudocode for the DKDL-Net algorithm is presented in Algorithm 1.

```
Require: Model parameters $W_{conv1d}^{freez}$, $W_{fc}^{freez}$, $B_{conv1d}^{freez}$, $B_{fc}^{freez}$ after training with DKD.
    Input:
        • CWRU dataset ($x_1^{CWRU}$, $x_2^{CWRU}$, ..., $x_n^{CWRU}$);
        • ($x_1^{CWRU}$, $x_2^{CWRU}$, ..., $x_n^{CWRU}$ ) of input to FFT get ($x_1^f$,...,$x_n^f$);
        • Use chunk get ($x_1^m$,$x_2^m$,...,$x_n^m$);
    Initialisation:
        • initial $A_{conv1d}$ , $B_{conv1d}$ , matrix ,Rank = r;
        • initial $A_{fc}$ , $B_{fc}$ , matrix ,Rank = r;
1: for $e^{th}$ training iteration in total epoch, $E$ do
2:     $\hat{F}_{conv1d}^{L} = B_{conv1d} \cdot A_{conv1d} \cdot (x_1^{m,e}, x_2^{m,e}, ..., x_n^{m,e})$;
3:     $\hat{F}_{conv1d}^{initial} = w_{conv1d}^{freez} \cdot \hat{F}_{conv1d}^{L} + B_{conv1d}^{freez}$;
4:     $\hat{F}_{fc}^{L} = B_{fc} \cdot A_{fc} \cdot \hat{F}_{conv1d}^{initial}$;
5:     $\hat{F}_{fc}^{initial} = w_{fc}^{freez} \cdot \hat{F}_{fc}^{L} + B_{fc}^{freez}$;
6:     $X_{output}^{e} = \hat{F}_{fc}^{initial}$;
7:     Calculate loss function and gradient, then use Adam optimizer, update parameters by Backpropagation algorithm;
8: end for
    Output: DKDL-Net model weights file (.pth);
```

**Algorithm 1**. The training phase of DKDL-Net model.

## Structural configuration of the model

The framework of the Teacher model is shown in Fig. 2 (top), and the relevant parameter configurations for the model's input, output, and convolutional kernel size are presented in Table 1.

The Student model is a single layer network and framework shown in Fig. 2 (below), and the relevant parameter configurations for the model's input, output, and convolutional kernel size are presented in Table 2.

The DKDL-Net model is a single layer network and framework shown in Fig. 3, adding LoRA module before convolutional and fully connected layers, and the relevant parameter configurations for the model's input, output, and convolutional kernel size are presented in Table 3.

| Name | Kernel size/stride | Input size | Output size | Activation function | #Parameters |
|---|---|---|---|---|---|
| Conv1D_1 | (64, ) / 8 | 1 × 1024 | 16 × 128 | ReLU | 1072 |
| Pooling_1 | 2 / 2 | 16 × 128 | 16 × 64 | | 0 |
| Conv1D_2 | (3, ) / 1 | 16 × 64 | 32 × 64 | ReLU | 1632 |
| Pooling_2 | 2 / 2 | 32 × 64 | 32 × 32 | | 0 |
| Conv1D_3 | (3, ) / 1 | 32 × 32 | 64 × 32 | ReLU | 6336 |
| Pooling_3 | 2 / 2 | 64 ×32 | 64 × 16 | | 0 |
| Conv1D_4 | (3, ) / 1 | 64 × 16 | 64 × 16 | ReLU | 12480 |
| Pooling_4 | 2 / 2 | 64 × 16 | 64 × 8 | | 0 |
| Conv1D_5 | (3, ) / 1 | 64 × 8 | 64 × 8 | ReLU | 12480 |
| Pooling_5 | 2 / 2 | 64 × 8 | 64 × 4 | | 0 |
| Conv1D_6 | (3, ) / 1 | 64 × 4 | 128 × 2 | ReLU | 24960 |
| Pooling_6 | 2 / 2 | 128 × 2 | 128 × 1 | | 0 |
| FC_1 | | 128 | 64 | ReLU | 8256 |
| FC_2 | | 64 | 32 | | 2080 |
| FC_3 | | 32 | 10 | | 330 |
| Total of trainable parameters | | | | | 69626 |

**Table 1**. Teacher model parameters applied.

| Name | Kernel size/stride | Input size | Output size | Activation function | #Parameters |
|---|---|---|---|---|---|
| Conv1D | (64, ) / 8 | 1 × 1024 | 4 × 128 | ReLU | 260 |
| Pooling | 2 / 2 | 4 × 128 | 4 × 64 | | 0 |
| FC | | 256 | 10 | | 2570 |
| Total of trainable parameters | | | | | 2830 |

**Table 2**. Student model parameters.

| Name | Kernel size/stride | Input | Output | Activation function | #Parameters |
|---|---|---|---|---|---|
| Conv1D_LoRA | | 1 × 1024 | 4 × 128 | | 816 |
| Conv1D | (64, ) / 8 | 1 × 1024 | 4 × 128 | ReLU | 260 |
| Pooling | 2 / 2 | 4 × 128 | 4 × 64 | | 0 |
| FC_LoRA | | 256 | 10 | | 3192 |
| FC | | 10 | 10 | | 2570 |
| Total of trainable parameters | | | | | 6838 |

**Table 3**. DKDL-Net model.

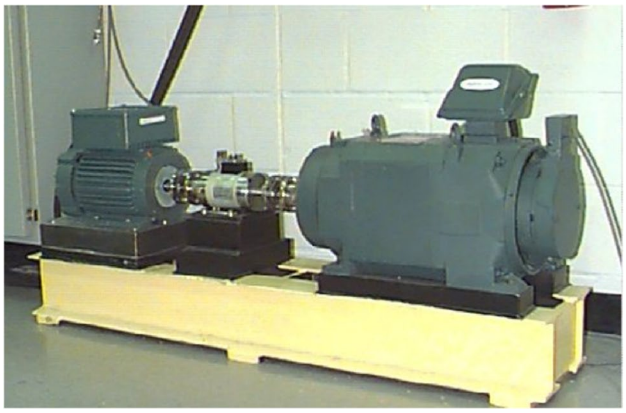


**Fig. 4**. Data collection machine tools.

## Experiments and results

### Experimental configurations

**Environment configuration.** All experiments for this model were conducted on a Windows 11 system with an Intel Core i7-9850H CPU at 2.60GHz and an NVIDIA GeForce GTX 1650 with Max-Q Design 4GB GPU. The code was run in an environment with Python 3.10.13 and PyTorch 2.0.1+cu117.

During the training of DKDL-Net, we utilized the Adaptive Moment Estimation (Adam) optimizer with a learning rate (LR) of 0.005 and a weight decay coefficient of 0.0001. We employed the cross-entropy loss function evaluate the loss between true and predicted labels.

**Baseline.** We selected MCNN-LSTM, FaultNet, BearingPGA-Net, KDSCNN, and WDCNN models as our baseline models. Among these, BearingPGA-Net and WDCNN are SOTA (state-of-the-art) models. However, BearingPGA-Net is more of a lightweight model, whereas WDCNN is a relatively large-scale model. KDSCNN is also a lightweight model and is comparable to our model.

**Benchmark.** Our benchmark is based on the CWRU dataset, curated by the Case Western Reserve University Bearing Data Center. The machine to generate the CWRU dataset is shown in Fig. 4. This dataset includes ten categories, comprising nine types of faulty bearings and one healthy bearing. Vibration data are collected at 12 kHz and 48 kHz. The fault types and labels of the CWRU dataset are shown in Table 4, waveforms as shown in Fg. 5.

**Evaluation metrics.** We use Accuracy, Precision, Recall and F1-Score to evaluate the performance of our model, which is calculated as in Eqs. (15) (16) (17) (18).

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{15}$$

| Faulty Mode | Fault size(mm) | Total dataset | class labels |
|---|---|---|---|
| Health | - | 280 | 0 |
| Ball cracking (Minor) | 0.18 | 280 | 1 |
| Ball cracking (Moderate) | 0.36 | 280 | 2 |
| Ball cracking (Severe) | 0.53 | 280 | 3 |
| OR cracking (Minor) | 0.18 | 280 | 4 |
| OR (Moderate) | 0.36 | 280 | 5 |
| OR (Severe) | 0.53 | 280 | 6 |
| IR (Minor) | 0.18 | 280 | 7 |
| IR (Moderate) | 0.36 | 280 | 8 |
| IR (Severe) | 0.53 | 280 | 9 |

**Table 4.** The labels in the CWRU dataset and their corresponding fault types.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{16}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{17}$$

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{18}$$

where *TP* represents True Positive, *TN* represents True Negative, *FP* represents False Positive, *FN* represents False Negative.

## Bearing fault detection experimental results

We conducted experiments on the CWRU dataset, and as shown in Table 5, the ratio of test data and training data is divided into 33.3% and 66.7%, the F1-Score of the DKDL-Net model is higher than that of the MCNN-LSTM, FaultNet, BearingPGA-Net, KDSCNN, and WDCNN models. Additionally, compared to the state-of-the-art (SOTA) model, our model achieved an improvement of 0.58%. Despite having only 6838 parameters, our model has a higher accuracy than BearingPGA-Net by 0.58%, with only 4008 more parameters. Compared to the KDSCNN model, our model has 948 more trainable parameters, yet it outperforms KDSCNN by 0.98%. In summary, the DKDL-Net (our) model achieves higher accuracy while maintaining fewer parameters and lower Flops.

As shown in Table 6, we evaluated the F1-Score, Precision, and Recall of the DKDL-Net model on the test dataset of CWRU. The DKDL-Net model outperforms the BearingPGA-Net, FaultNet, and MCNN-LSTM models in all three metrics. Compared to the best-performing BearingPGA-Net (SOTA) model, we achieved an improvement of nearly 0.55% across all three metrics. In conclusion, our model is the best-performing model on the CWRU dataset.

As shown in Table 7, under the same configuration, our student model trained using Decoupled Knowledge Distillation (DKD) has 2,830 parameters. Compared to the teacher model, the student model's trainable parameters are reduced by approximately 95.93%, but its F1-Score, Precision, and Recall decrease by 2.07%, 1.92%, and 2.08%, respectively. This indicates that while the DKD model can compress model parameters, its accuracy significantly decreases.

On the other hand, the DKDL-Net model, based on DKD compression and LoRA fine-tuning, has 6,838 trainable parameters. Compared to the teacher model, the DKDL-Net model's trainable parameters are reduced by approximately 90.20%, indicating that DKDL-Net can significantly reduce model complexity and resource requirements. The F1-Score, Precision, and Recall decrease by only 0.09%, 0.12%, and 0.12%, respectively. This demonstrates that the accuracy loss caused by the DKDL-Net model compared to the teacher model is negligible. Therefore, the DKDL-Net model effectively compresses parameters while maintaining high accuracy.

In summary, our model achieves a compression ratio of 90.20% with a negligible decrease in accuracy compared to the Teacher model, making it highly efficient in terms of compression while maintaining high accuracy.

Finally, we computed the confusion matrices for the DKDL-Net model as shown in Fig. 6b, Student model as shown in Fig. 6a, and Teacher model as shown in Fig. 6c on the CWRU dataset, with 2,500 test samples, 250 samples per class.

We plotted the ROC curves for both the DKDL-Net model and the student model show in Fig. 7. As shown in Fig. 7b for the DKDL-Net model and Fig. 7a for the student model, it can be observed that the ROC curve for the student model is more jagged, while the ROC curve for the DKDL-Net model is smoother. Additionally, the area under the curve (AUC) for each class in the DKDL-Net model is generally larger than that of the student model. This indicates that our algorithm outperforms the student model's algorithm, demonstrating better performance.

We tested the DKDL-Net model on the CWRU bearing fault detection dataset. Under the same training configuration as see Section Experimental Configurations, we tested 2500 samples, result as Shown in Table 8,

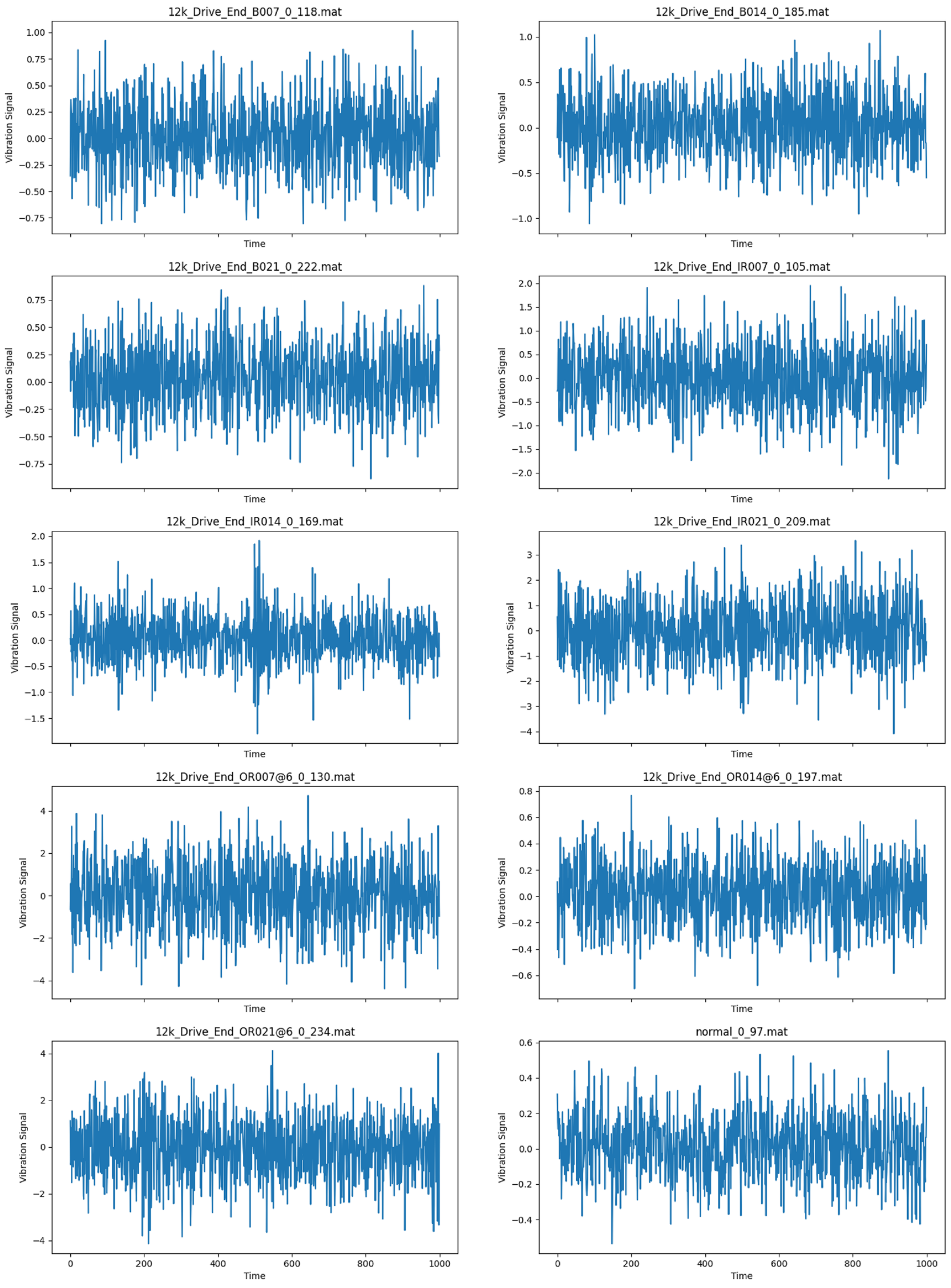


**Fig. 5**. Bearing abnormal and normal waveform format.

and the DKDL-Net model required an average of 1757 $\mu$s per sample. DKDL-Net model with teacher model have 1x faster inference. This figure also demonstrates the high efficiency of our model.

Overall, the DKDL-Net model outperformed the Student model, and its performance was comparable to that of the Teacher model. Therefore, our DKDL-Net model can maintain good results even under high compression.

**Ablation experiment.** We fixed the $\beta$ and $\gamma$ parameters and analyzed the influence of different alphas on the model. The results are presented in Table 9.

| Model | F1-Score(%) | #Parameters | #FLOPs |
|---|---|---|---|
| MCNN-LSTM | 98.46 | 73.48K | – |
| FaultNet | 98.50 | 627.05K | – |
| WDCNN | 98.39 | 66.79K | 1.61M |
| KDSCNN | 98.50 | 5.89K | 70.66K |
| BearingPGA-Net | 98.90 | 2.83K | 78.34K |
| **DKDL-Net (Ours)** | **99.50** | **6.38K** | **70.65K** |

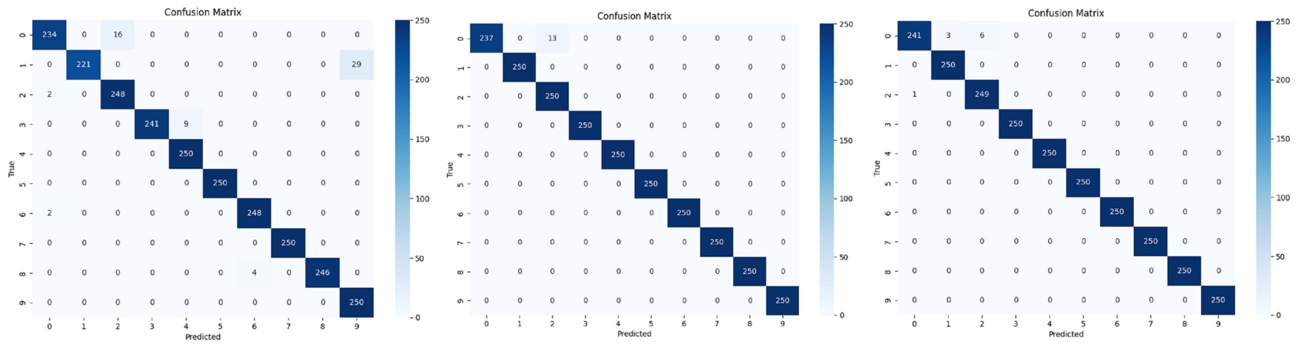
**Table 5**. Comparison of F1-Score, model size and computational cost among different bearing fault diagnosis algorithms.

| Model | Precision(%) | Recall(%) | F1-score(%) |
|---|---|---|---|
| MCNN-LSTM | 98.46 | 97.85 | 97.856 |
| FaultNet | 98.60 | 98.57 | 98.57 |
| BearingPGA-Net | 98.98 | 98.92 | 98.90 |
| **DKDL-Net(our)** | **99.48** | **99.48** | **99.50** |

**Table 6**. Comparison of bearing fault diagnosis precision, Recall and F1-Score with different algorithms.

| Model | Precision(%) | Recall(%) | F1-score(%) | #Parameters |
|---|---|---|---|---|
| Teacher | **99.60** | **99.60** | **99.59** | 69626 |
| Sudent | 97.68 | 97.52 | 97.52 | **2830** |
| **DKDL-Net(our)** | 99.48 | 99.48 | 99.50 | 6838 |

**Table 7**. Comparison of the parameters and assessment metrics of the teacher model, student model and DKDL-Net model.



**(a)** Student **(b)** DKDL-Net **(c)** Teacher

**Fig. 6**. Confusion matrix for student model and DKDL-Net model on CWRU dataset.

In terms of the results in Fig. 8, we fixed the alpha at 0.3 and analyzed the influence of beta and gamma on the model. We analyzed that when beta is greater than gamma, the performance ability of the model will improve. The experimental results show that the model is able to exhibit strong performance when $\beta$ is larger than $\gamma$.

## Discussion

In this article, we propose a CNN model named DKDL-Net, which is based on decoupled knowledge distillation training and Low-Rank Adaptation fine-tuning. DKDL-Net is a single-layer neural network with only 6,838 parameters and an inference speed of 1,767 $\mu$s. It achieved an F1-Score of 99.50% on the CRWU dataset, representing a 0.60% improvement in F1-Score compared to the state-of-the-art (SOTA) models. Therefore, our model is highly efficient in detection while maintaining high accuracy. Moreover, the model is extremely lightweight, making it suitable for practical industrial applications.

(limatations and future work)While our model excels on the CWRU dataset, its generalizability to other datasets (e.g., Paderborn or SEU) remains untested. Additionally, the 0.5% residual accuracy gap compared to

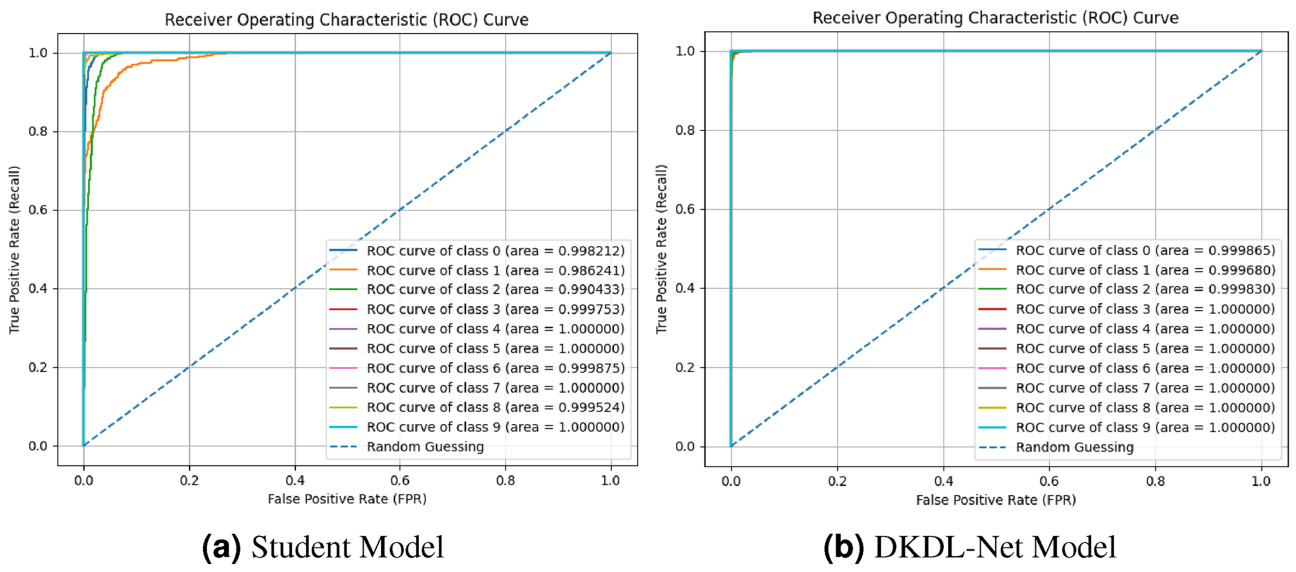


**Fig. 7**. ROC curves for student and DKDL-Net models on the CWRU dataset.

| **Model** | **Num. of test samples** | **Avg. inference time($\mu s$)** |
|---|---|---|
| Teacher | 2500 | 3816 |
| **DKDL-Net (Our)** | 2500 | 1757 |

**Table 8**. Inference time for the DKDL-Net model.

| $\alpha$ | **0.1** | **0.2** | **0.3** | **0.4** | **0.5** |
|---|---|---|---|---|---|
| Accuracy(%) | 96.09 | 95.93 | 96.13 | 95.76 | 94.04 |

**Table 9**. Accuracy (%) under different values of $\alpha$, where $\gamma$ and $\beta$ is fixed.

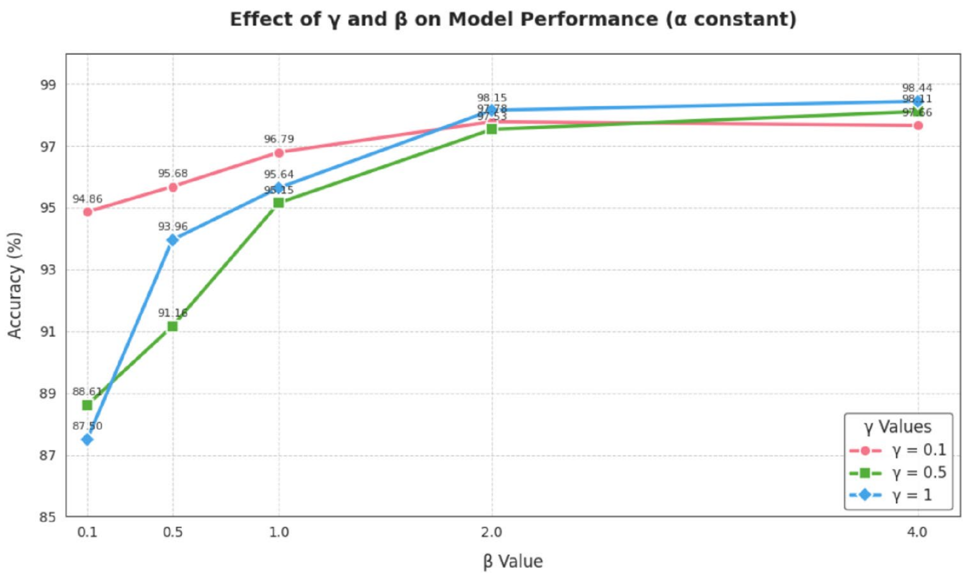


**Fig. 8**. Performance comparison under different combinations of $\gamma$ and $\beta$.

the teacher model suggests room for improvement in distillation efficiency. Future work could explore hybrid compression techniques (e.g., pruning + DKD) or adaptive LoRA ranks to further narrow this gap. Expanding evaluations to heterogeneous data sources and investigating cross-domain transferability would also validate broader applicability.

## Data availability
The dataset CWRU[35] used in this article comes from a public dataset, which can be accessed online, and of course by requesting corresponding authors.

## Code availability
The source code supporting the findings of this study is available at https://github.com/lipengyi0829/DKDL-Net and is currently under consideration for open-source licensing. Researchers are encouraged to contact the corresponding author for usage guidance.

Received: 19 June 2024; Accepted: 10 June 2025
Published online: 16 October 2025

## References
1. Bonnett, A. H. & Yung, C. Increased efficiency versus increased reliability. *IEEE Ind. Appl. Mag.* **14**, 29–36 (2008).
2. Benbouzid, M. E. H. A review of induction motors signature analysis as a medium for faults detection. *IEEE Trans. Ind. Electron.* **47**, 984–993 (2000).
3. Kumar, N. & Satapathy, R. Bearings in aerospace, application, distress, and life: a review. *JJ. Fail. Anal. Prev.* **23**, 915–947 (2023).
4. Nandi, S., Toliyat, H. A. & Li, X. Condition monitoring and fault diagnosis of electrical motors-a review. *IEEE transactions on energy conversion* **20**, 719–729 (2005).
5. Zhang, S., Zhang, S., Wang, B. & Habetler, T. G. Deep learning algorithms for bearing fault diagnostics-a comprehensive review. *IEEE Access* **8**, 29857–29881 (2020).
6. Pacheco-Cherrez, J., Fortoul-Diaz, J. A., Cortes-Santacruz, F., Aloso-Valerdi, L. M. & Ibarra-Zarate, D. I. Bearing fault detection with vibration and acoustic signals: Comparison among different machine leaning classification methods. *Engineering Failure Analysis* **139**, 106515 (2022).
7. Shao, H., Jiang, H., Zhang, X. & Niu, M. Rolling bearing fault diagnosis using an optimization deep belief network. *Meas.Sci. Technol.* **26**, 115002 (2015).
8. Che, C., Wang, H., Ni, X. & Fu, Q. Domain adaptive deep belief network for rolling bearing fault diagnosis. *Comput. &Ind. Eng.* **143**, 106427 (2020).
9. Tao, J., Liu, Y., Yang, D. *et al.* Bearing fault diagnosis based on deep belief network and multisensor information fusion. *Shock. vibration* **2016** (2016).
10. Yang, K., Zhao, L. & Wang, C. A new intelligent bearing fault diagnosis model based on triplet network and svm. *Sci. Rep.* **12**, 5234 (2022).
11. Zhang, W., Li, C., Peng, G., Chen, Y. & Zhang, Z. A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load. *Mech. Syst. Signal. Process.* **100**, 439–453 (2018).
12. Waziralilah, N. F., Abu, A., Lim, M., Quen, L. K. & Elfakharany, A. A review on convolutional neural network in bearing fault diagnosis. In *MATEC Web of Conferences*, vol. 255, 06002 (EDP Sciences, 2019).
13. Zhang, J. et al. A new bearing fault diagnosis method based on modified convolutional neural networks. *Chin. J. Aeronaut.* **33**, 439–447 (2020).
14. Saeed, F. et al. A robust approach for industrial small-object detection using an improved faster regional convolutional neural network. *Sci. Rep.* **11**, 23390 (2021).
15. Ren, L., Sun, Y., Cui, J. & Zhang, L. Bearing remaining useful life prediction based on deep autoencoder and deep neural networks. *J. Manuf. Syst.* **48**, 71–77 (2018).
16. Huang, F. *et al.* A rolling bearing fault diagnosis method based on interactive generative feature space oversampling-based autoencoder under imbalanced data. *Struct. Heal. Monit.* 14759217241248209 (2024).
17. Cui, M., Wang, Y., Lin, X. & Zhong, M. Fault diagnosis of rolling bearings based on an improved stack autoencoder and support vector machine. *IEEE Sensors J.* **21**, 4927–4937 (2020).
18. Gao, Y., Liu, X. & Xiang, J. Fem simulation-based generative adversarial networks to detect bearing faults. *IEEE Trans. Industr. Inform.* **16**, 4961–4971 (2020).
19. Mao, W., Liu, Y., Ding, L. & Li, Y. Imbalanced fault diagnosis of rolling bearing based on generative adversarial network: A comparative study. *IEEE Access* **7**, 9515–9530 (2019).
20. Liu, H. et al. Unsupervised fault diagnosis of rolling bearings using a deep neural network based on generative adversarial networks. *Neurocomputing* **315**, 412–424 (2018).
21. Chen, X. *et al.* Deep transfer learning for bearing fault diagnosis: A systematic review since 2016. *IEEE Trans. Instrum. Meas.* (2023).
22. Wu, Z., Jiang, H., Zhao, K. & Li, X. An adaptive deep transfer learning method for bearing fault diagnosis. *Measurement* **151**, 107227 (2020).
23. Zhu, J., Chen, N. & Shen, C. A new deep transfer learning method for bearing fault diagnosis under different working conditions. *IEEE Sensors J.* **20**, 8394–8402 (2019).
24. He, J. et al. Deep transfer learning method based on 1d-cnn for bearing fault diagnosis. *Shock and Vibration* **2021**, 1–16 (2021).
25. Guo, L., Li, N., Jia, F., Lei, Y. & Lin, J. A recurrent neural network based health indicator for remaining useful life prediction of bearings. *Neurocomputing* **240**, 98–109 (2017).
26. Cui, Q., Li, Z., Yang, J. & Liang, B. Rolling bearing fault prognosis using recurrent neural network. In *2017 29th Chinese Control And Decision Conference (CCDC)*, 1196–1201 (IEEE, 2017).
27. Lee, K., Kim, J.-K., Kim, J., Hur, K. & Kim, H. Stacked convolutional bidirectional lstm recurrent neural network for bearing anomaly detection in rotating machinery diagnostics. In *2018 1st IEEE International Conference on Knowledge Innovation and Invention (ICKII)*, 98–101 (IEEE, 2018).
28. Chen, X., Zhang, B. & Gao, D. Bearing fault diagnosis base on multi-scale cnn and lstm model. *J. Intell. Manuf.* **32**, 971–987 (2021).
29. Yu, L., Qu, J., Gao, F., Tian, Y. *et al.* A novel hierarchical algorithm for bearing fault diagnosis based on stacked lstm. *Shock and Vibration* **2019** (2019).
30. Pan, H., He, X., Tang, S. & Meng, F. An improved bearing fault diagnosis method using one-dimensional cnn and lstm. *Journal of Mechanical Engineering/Strojniški Vestnik* **64** (2018).
31. Chen, X., Zhang, B. & Gao, D. Bearing fault diagnosis base on multi-scale cnn and lstm model. *J. Intell. Manuf.* **32**, 971–987 (2021).

32. Fang, H. et al. Lefe-net: A lightweight efficient feature extraction network with strong robustness for bearing fault diagnosis. *IEEE Trans. Instrum. Meas.* **70**, 1–11 (2021).
33. Zhang, W., Peng, G., Li, C., Chen, Y. & Zhang, Z. A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals. *Sensors* **17**, 425 (2017).
34. Li, S.-Y. & Gu, K.-R. Smart fault-detection machine for ball-bearing system with chaotic mapping strategy. *Sensors* **19**, 2178 (2019).
35. Case Western Reserve University Bearing Data Center. Case western reserve university bearing data center. https://csegroups.case.edu/bearingdatacenter/home. Accessed: Dec. 22, 2019.
36. Fang, H. et al. Clformer: A lightweight transformer based on convolutional embedding and linear self-attention with strong robustness for bearing fault diagnosis under limited sample conditions. *IEEE Trans. Instrum. Meas.* **71**, 1–8 (2021).
37. Griffin, D. & Lim, J. Signal estimation from modified short-time fourier transform. *IEEE Trans. Acoust.* **32**, 236–243 (1984).
38. Rehman, N. & Mandic, D. P. Multivariate empirical mode decomposition. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **466**, 1291–1302 (2010).
39. Rilling, G., Flandrin, P., Goncalves, P. *et al.* On empirical mode decomposition and its algorithms. In *IEEE-EURASIP workshop on nonlinear signal and image processing*, vol. 3, 8–11 (Grado: IEEE, 2003).
40. Zhao, B., Cui, Q., Song, R., Qiu, Y. & Liang, J. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 11953–11962 (2022).
41. Hu, E. J. et al. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021).
42. Neupane, D. & Seok, J. Bearing fault detection and diagnosis using case western reserve university dataset with deep learning approaches: A review. *IEEE Access* **8**, 93155–93178 (2020).
43. Islam, M. M. & Kim, J.-M. Automated bearing fault diagnosis scheme using 2d representation of wavelet packet transform and deep convolutional neural network. *Comput. Ind.* **106**, 142–153 (2019).
44. Wan, L., Chen, Y., Li, H. & Li, C. Rolling-element bearing fault diagnosis using improved lenet-5 network. *Sensors* **20**, 1693 (2020).
45. Shi, Y. et al. Enhanced lightweight multiscale convolutional neural network for rolling bearing fault diagnosis. *IEEE Access* **8**, 217723–217734 (2020).
46. You, K., Qiu, G. & Gu, Y. Rolling bearing fault diagnosis using hybrid neural network with principal component analysis. *Sensors* **22**, 8906 (2022).
47. Ji, M. et al. A neural network compression method based on knowledge-distillation and parameter quantization for the bearing fault diagnosis. *Appl. Soft Comput.* **127**, 109331 (2022).
48. Liao, J.-X. et al. Bearingpga-net: A lightweight and deployable bearing fault diagnosis network via decoupled knowledge distillation and fpga acceleration. *IEEE Trans. Instrum. Meas.* (2023).
49. Magar, R., Ghule, L., Li, J., Zhao, Y. & Farimani, A. B. Faultnet: a deep convolutional neural network for bearing fault classification. *IEEE access* **9**, 25189–25199 (2021).
50. Li, Z., Li, H. & Meng, L. Model compression for deep neural networks: A survey. *Computers* **12**, 60 (2023).
51. Liang, T., Glossner, J., Wang, L., Shi, S. & Zhang, X. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing* **461**, 370–403 (2021).
52. Nguyen, L. T., Kim, J. & Shim, B. Low-rank matrix completion: A contemporary survey. *IEEE Access* **7**, 94215–94237 (2019).
53. Gou, J., Yu, B., Maybank, S. J. & Tao, D. Knowledge distillation: A survey. *Int. J. Comput. Vis.* **129**, 1789–1819 (2021).
54. Choudhary, T., Mishra, V., Goswami, A. & Sarangapani, J. A comprehensive survey on model compression and acceleration. *Artif. Intell. Rev.* **53**, 5113–5155 (2020).
55. Zhu, X., Li, J., Liu, Y., Ma, C. & Wang, W. A survey on model compression for large language models. arXiv preprint arXiv:2308.07633 (2023).
56. Wang, J. et al. Review of large vision models and visual prompt engineering. *Meta-Radiology* 100047 (2023).
57. Li, Z. et al. When object detection meets knowledge distillation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).
58. Fan, Y., Jiang, F., Li, P. & Li, H. Grammargpt: Exploring open-source llms for native chinese grammatical error correction with supervised fine-tuning. In *CCF International Conference on Natural Language Processing and Chinese Computing*, 69–80 (Springer, 2023).
59. Lermen, S., Rogers-Smith, C. & Ladish, J. Lora fine-tuning efficiently undoes safety training in llama 2-chat 70b. arXiv preprint arXiv:2310.20624 (2023).
60. Thakur, A. & Vashisth, R. A unified module for accelerating stable-diffusion: Lcm-lora. arXiv preprint arXiv:2403.16024 (2024).
61. Rezazadeh, N., Perfetto, D., de Oliveira, M., De Luca, A. & Lamanna, G. A fine-tuning deep learning framework to palliate data distribution shift effects in rotary machine fault detection. *Structural Health Monitoring* 14759217241295951 (2024).
62. Chen, W., Miao, Z. & Qiu, Q. Parameter-efficient tuning of large convolutional models. arXiv preprint arXiv:2403.00269 (2024).
63. Sharma, M., Chatterjee, M., Peng, K.-C., Lohit, S. & Jones, M. Tensor factorization for leveraging cross-modal knowledge in data-constrained infrared object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 924–932 (2023).
64. De Boer, P.-T., Kroese, D. P., Mannor, S. & Rubinstein, R. Y. A tutorial on the cross-entropy method. *Ann. Oper. Res.* **134**, 19–67 (2005).

## Acknowledgements
The research was carried out within the financial support for the agreement on the provision of a grant from the federal budget in the form of a subsidy to the Federal State Budgetary Educational Institution of Higher Professional Education "Saint-Petersburg State University" dated April 16, 2025 No. 139-15-2025-007 id: 000000C313925P3O002.

## Author contributions
O. P. served as the project supervisor, responsible for project advancement and conceptual guidance. L. P., as the team leader, was responsible for the overall model design, proposed the DKDL-Net model, trained the model, organized experimental data, and wrote the manuscript. H. Y., as the algorithm researcher, was responsible for the research of the model's algorithm. L. J., as the project researcher, was responsible for the research of the problem. S. Z., as the algorithm researcher, was responsible for the research of model compression algorithms. F. G., as the algorithm researcher, was responsible for analyzing experimental results and conducting some experiments. M. L., as the algorithm researcher, was responsible for analyzing experimental results and conducting some experiments.

## Declarations

## Competing interests
The authors declare no competing interests.

## Accession codes
Our code is available at Github link: https://github.com/SPBU-LiPengyi/DKDL-Net.git.

## Additional information
**Correspondence** and requests for materials should be addressed to P.L.

**Reprints and permissions information** is available at www.nature.com/reprints.

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.