Intelligent Classification of Wear Particles Based on Deep Convolutional Neural Network

Fengguang Jia\textsuperscript{1,2\textdagger}, Fulin Yu\textsuperscript{1}, Lei Song\textsuperscript{1}, Shaojun Zhang\textsuperscript{1}, Hongyuan Sun\textsuperscript{1}

College of Naval Architecture and Marine Engineering, Shandong Jiaotong University, Weihai 264209, China
College of Merchant Marine, Shanghai Maritime University, Shanghai 201306, China

Email: dz_jfg@163.com.

Abstract. The intelligent classification of wear particles has remained a high priority research area for ferrography technology and industrial tribology. In this study, five deep convolutional neural network (DCNN) models are used for identification of seven kinds of wear particles. Instead of manually designing and selecting the features, the proposed DCNN realizes an end-to-end processing. Wear particles’ dataset is built by various kinds of tribology experiments. The experimental results show that the accuracy of DenseNet121 on the test set is 88.39\%. A conclusion can be drawn that DCNN is suited for wear particles’ classification and can be put into practical use in condition monitoring systems in the future.

1. Introduction
Wear particles in the lubricant used in machine equipment contain rich information that can be used to evaluate the wear condition and determine the wear mechanisms occurring.\cite{1} As a non-destructive testing technology, wear particle analysis can, on the one hand, prolong the oil change intervals and reduce unnecessary regular maintenance, on the other hand, it can predict potential failures from early stage to avoid catastrophic accidents.\cite{2, 3}

Different types of wear particles are generated through the wear process, including normal, cutting, spherical, severe sliding, fatigue laminar, chunky spalling and nonferrous metal particles. Ferrography is a methodology that uses a high-gradient magnetic field to extract wear particles from a fluid sample as it flows down a specially prepared microscope substrate.\cite{4} Ferrography has been successfully used to monitor the conditions of aircraft engines, marine engines and gearboxes.\cite{5} The problem with current off-line ferrography techniques is that the particle morphology assessment, particle classification and the machine status evaluation relies heavily on human expertise, which is time-consuming, costly and objective. These limitations have driven further development and need to automate the technique. The automation of wear particles’ classification is the cornerstone for on-line wear particles analysis which will certainly benefit the industry and reduce the need for manpower.\cite{6}

The intelligent classification of a ferrography image remains a challenge despite numerous studies in this area. Peng Z et al. used the cluster analysis in the gray system theory for the identification of wear particles.\cite{7, 8} Also, in the study by Stachowiak and Podsiadlo, attempt had been made to distinguish adhesive, fatigue and abrasive particles using area, perimeter and elongation parameters, these studies showed that these simple parameters were effective for certain types of wear particles such as abrasive particles and sphere like fatigue particles.\cite{9-11} Tonghai Wu team carried out extensive research on
the on-line monitoring of wear particles.[12] The team had independently developed a set of online visualized ferrography and proposed that the area of particle coverage and particle diameter in the visible area can be used as indicators for judging the status of the machine. This study provided a hardware design for on-line ferrography. However, because the surface texture of the particles was not obtained, the wear mode cannot be accurately reflected. To sum up, a lot of research had been done on the classification of wear particles. The recognition of cutting, spherical and oxide abrasive grains had basically been realized while classification of similar particles such as severe sliding particles, laminar particles, and fatigue chunky particles was not entirely successful yet.

Some recognition methods, including support vector machine (SVM) [13] and artificial neural network (ANN) [14], had been applied in the classification of wear particles.[15] However, shallow neural network models usually required input features, and thus the effectiveness of these methods depended on manual feature extraction to a certain extent. The linear flow of computer-aided ferrography image analysis made it difficult to tune the algorithms as a whole, which limited their application in automatic wear particle analysis. In 2006, Hinton proposed the theory of Deep Learning (DL).[16] Deep convolutional neural network (DCNN) is a popular research area in the field of DL, and has certain advantages such as the self-learning and transmission of features, as well as an end-to-end processing, which significantly improves the accuracy of the classification or recognition over traditional methods.[17-19] In 2012, Krizhevsky et al. used the DCNN to obtain the best classification accuracy at the ImageNet Large Scale Visual Recognition Challenge (LSVRC).[20] Since then, DCNN models have been dominated in all kinds of image recognition competitions.

In this paper, deep convolutional neural network (DCNN) for intelligent classification of wear particles is proposed. It can realize the identification of seven types of wear particles, namely, normal, cutting, spherical, severe sliding, fatigue laminar, chunky spalling and nonferrous metal particles. The DCNN model provides an end-to-end processing for automatic wear particle recognition, which is applicable to the wear condition analysis of the mechanical equipment.

2. Deep convolutional neural network

2.1. Architecture of DCNN

As shown in figure 1, the basic structure of DCNN is composed of input layers, convolutional layers (CONV layers), pooling layers, fully-connected layers and output layers. Input layer contains pixel information of input images. The convolution operation uses a fixed-size convolution kernel (filter) to slide on the input image to extract features and obtain the feature maps. Nowadays, a deep neural network can contains a thousand CONV layers, giving the model extremely representation power.[20] A pooling operation is a down-sampling process by aggregating the statistics of adjacent areas in the feature maps. This will reduce the computational cost and further improve the representation of the features. Using pooling, the purpose of describing high-level features with a small-sized feature map can be realized. Fully-connected layers serve the purpose of final classification in the end of structure. Non-linearity is referred to a non-linear activation function. Typically it is applied right after CONV or FC layer, the non-linearity is necessary for the model since the network can be simplify to a single linear function mathematically. The most popular non-linearity used is rectified linear unit (ReLU) because of its simplicity.[21] Normalization is a solution used for a gradient explosion, which adjusts the input distribution to increase the training speed and allows a higher learning rate in a DCNN.[22]

The convolution formula is as follows:

$$X^l_i = f\left(\sum_{i\in M_j} X^{l-1}_i \times W^l_{ij} + b^l_{ij}\right)$$ (1)

In the formula, \(X^l_i\) is the feature map outputed by convolution operation, \(X^{l-1}_i\) is the input feature map, \(M_j\) is the set of input feature map, \(W^l_{ij}\) is the convolution kernel, \(b^l_{ij}\) is the mapping bias of the output feature map, and \(f\) is the activation function.
The pooling formula is as follows:

\[ x_{i}^{l+1} = \text{down}(x_{i}^{l}) \]

(2)

In the formula, \( x_{i}^{l+1} \) represents the feature map outputted by pooling operation, and \( \text{down()} \) represents the maximum or mean operation of the input feature map with a fixed-size filter of \( n \times n \).

2.2. Typical DCNN models

Various DCNN models for image classification have been developed in the past decade. We have tested five popular DCNN models which have quite different architectures in terms of layer types, layer shapes and the way layers are connected. In this section, the detail of these models is introduced.

2.2.1. AlexNet.

AlexNet contains 8 layers with weights, the first 5 are CONV layers and the remaining 3 are FC layers.[17] The output of the last fully-connected layer is fed to a 1000-way softmax which produces a distribution over the 1000 class labels. An illustration of the architecture of AlexNet is shown in figure 2. The detail of AlexNet model is summarized in table 1 as well as the five models described below.

2.2.2. VGG16.

VGG16 is a DCNN model with 16 layers as its name suggested. The model, consist of 13 CONV layers and 3 FC layers, has a rather plain design.[18] The basic block of the network is CONV layer
followed by a Batch Normalization and ReLU as its activation function, this design is pretty standard in current DCNN structure.[23] The architecture of VGG16 is shown in figure 3.

![Figure 3. The architecture of VGG16.](image)

2.2.3. *InceptionV3*.

InceptionV3 goes deeper to 22 layers.[24] This model is best known for a unique design: the inception module. As shown in figure 4, it contains three different modules called inception A, B and C. In the inception module, filters with different receptive field and max-pooling are used in parallel, and their outputs are concatenated for the output of the entire module. The idea is to catch features of multiple scales. The 1×1 filters in the inception module are applied to reduce the number of channels and further lighten the network. The 5×5 CONV layer in the inception module was replaced by two consecutive 3×3 CONV layers. The receptive field is the same while the number of parameters goes from 25 to 18. These alternations can leads to faster training as well as better generalization. Table 1 gives detail information about the architecture of InceptionV3.

![Inception module](image)

2.2.4. *ResNet*.

ResNet goes even deeper to 34 layers or more.[20] In theory, deeper network has greater expressive ability. However, deeper neural network is also more difficult to train, especially with current optimization algorithm which is stochastic gradient descent with back propagation. To address this problem, ResNet proposes a ‘shortcut’ module which contains an identity connection. As demonstrated in figure 5(a), instead of learning the function for the weight layers, the shortcut module learns the residual function F(x). This reformulation can help to precondition the mapping. In extreme
cases, if the identity mapping was the optimal function, it’s easier for the solver to push the residual to zero than to learn the function as a new one. ResNet family have quite many branches, ResNet50 is adopted in this experiment in which each stage is composed of a number of residual blocks. Typical residual block for ResNet50 is shown in figure 5(b). The detail layer information is shown in table 1.

Figure 5. Residual block.

2.2.5. DenseNet.
DenseNet goes even deeper to 121 layers or more.[25] To further improve the information flow between layers, DenseNet proposes a different connectivity pattern, which directs connections from any layer to all subsequent layers. DenseNet family have quite many branches, DenseNet121 is adopted in this experiment, which is composed of four dense blocks. Each dense block has five layers, and each layer is directly connected with all the previous layers to realize the reuse of features. To reduce the parameters, each layer of the network is designed to be very ‘narrow’, that is, only a small number of feature graphs (4) are learned. The architecture of dense block is shown in figure 6. The detail layer information is shown in table 1.

Figure 6. Dense block.
### Table 1. Architecture of 5 DCNN models.

| Model            | Input Size | CONV phase 1 | CONV phase 2 | CONV phase 3 | CONV phase 4 | CONV phase 5 | Dense Net |
|------------------|------------|--------------|--------------|--------------|--------------|--------------|-----------|
| AlexNet (8 layers) | 227×227 | conv3x3,64, conv3x3,64, max-pool | conv3x3,128, conv3x3,128, max-pool | conv3x3,256, conv3x3,256, max-pool | conv3x3,512, conv3x3,512, max-pool | conv3x3,512, conv3x3,512, max-pool | fc-4096, fc-4096, fc-1000 |
| VGG16 (16 layers) | 224×224 | conv3x3,32, conv3x3,32, max-pool | conv3x3,128, conv3x3,32, max-pool | conv3x3,256, conv3x3,64, max-pool | conv3x3,512, conv3x3,128, max-pool | conv3x3,512, conv3x3,128, max-pool | fc-4096, fc-4096, fc-1000 |
| InceptionV3 (22 layers) | 299×299 | conv3x3,32, conv3x3,32, max-pool | conv3x3,64, conv3x3,80, max-pool | conv3x3,192, inceptionA1,1288, inceptionA2,2288 | conv3x3,512, inceptionB1,768, inceptionB2,2768, inceptionB3,768, inceptionB4,768, inceptionB5,768 | conv3x3,128, inceptionC1,1280, inceptionC2,1280 | avg-pool,2048, fc-1000, avg-pool,2048, fc-1000 |
| ResNet50 (50 layers) | 224×224 | conv3x1,128, conv3x1,128, conv3x1,128, max-pool | conv3x3,128, conv3x3,32, max-pool | conv3x3,192, inceptionA1,1288, inceptionA2,2288 | conv3x3,512, inceptionB1,768, inceptionB2,2768, inceptionB3,768, inceptionB4,768, inceptionB5,768 | conv3x3,128, inceptionC1,1280, inceptionC2,1280 | avg-pool,2048, fc-1000, avg-pool,2048, fc-1000 |
| DenseNet121 (121 layers) | 224×224 | conv3x1,128, conv3x1,128, conv3x1,128, max-pool | conv3x3,128, conv3x3,32, max-pool | conv3x3,192, inceptionA1,1288, inceptionA2,2288 | conv3x3,512, inceptionB1,768, inceptionB2,2768, inceptionB3,768, inceptionB4,768, inceptionB5,768 | conv3x3,128, inceptionC1,1280, inceptionC2,1280 | avg-pool,2048, fc-1000, avg-pool,2048, fc-1000 |

Note: The convolutional layer parameters are denoted as ‘Receptive field, Number of channels’. The ReLU activation function and BN is not shown for brevity.

### 3. Experiments

#### 3.1. Dataset

The Bruker’s UMT Universal Mechanical Tester has been used in experiment to generate different kinds of wear particles. The machine, showed in figure 7(a), has a universal base that can be equipped with a range of drive modules simulating rotational, linear, or oscillating motions as well as an upper carriage fitted with a force measuring sensor. Three kinds of tribology tests had been performed to simulating different wear mode: pin on disk test, pin on plate test and 4-ball test. Pin on disk test was mainly for generating adhesive wear particles. The tests were set this long so that both mild and severe wear particles can be generated.

Pin on disk consists of a stationary pin loaded against a rotating disk, which was shown in figure 7(b). The upper pin was made of standard 416 stainless steel with cylindrical in shape and polished flat ends of 8mm, the disk was made of alloy steel E52100 with surface roughness RA=0.02μm. The Mobile Gard 412 had been used as lubricant. The tests were conducted at 900r/min, which means the linear velocity reached 170mm/s. The wear particles were generated under condition of a 30kg (294N) load for 25h. Pin on disk test was mainly for generating adhesive wear particles. The tests were set this long so that both mild and severe wear particles can be generated.
Compared to pin on disk test, pin on plate test replaced the rotating module by a reciprocating module. The upper pin was made of cast iron HT250 and the plate was made of Gr15 steel with surface roughness RA=0.2μm. The stroke was set to 18mm and frequency was set to 4Hz. The wear particles were produced with a 48kg (470.4N) load for 12h. Unlike pin on disk test, pin on plate test was meant to simulating the wear condition of reciprocating friction pairs such as piston-ring cylinder or hydraulic cylinder. Some of the severe sliding wear particles were generated. The fatigue wear particles were generated by 4-ball testing machine, the 4-ball module was shown in figure 7(c). The material of the ball was GCr15 (hardness 63HRC). Maximum load and speed were set to 1500N and 300r/min. In order to generate fatigue wear particles the running time was set to 30h. Fatigue began with weakening of lubrication condition and continuous stresses that exceeded the endurance limit of the material, causing cracks beneath the surface. This created fatigue laminar and eventually chunky spalling wear particles.

![Experiment equipment](image)

(a) Bruker’s UMT  (b) Pin on disk module  (c) 4-ball module  
(d) Ferrography analyzer T2FM  (e) Optical microscope OlympusBX51

**Figure 7.** Experiment equipment.

Other wear particles, such as normal, cutting, spherical and nonferrous metal particles, were generated from marine lubricating oil delivered for testing. The wear particles in lubricating oil were deposited under a high-gradient magnetic field of ferrography analyzer T2FM, which was shown in figure 7(d). Images were photographed by optical microscope OlympusBX51, which was shown in figure 7(e). Images of different kinds of particles were carefully selected and then cut off from original picture, after attaching them with label, the dataset was ready. In total, 1120 images, 160 for every kind of particles, had been chosen from thousands of pictures. Examples of wear particles generated in tests were shown in figure 8.

### 3.2. Experiment settings

In order to evaluate the effectiveness of the model, the dataset is split into training set, validation set and test set. The training set is used to train models. The validation set is used for setting the hyper-parameters of the model such as learning rate from the back-propagation algorithm. Different hyper-parameter values can be tested, the values that lead to the best performance on the validation set is kept. The test set is used for the final evaluation of a model. To prevent over-fitting, the rule of thumb is that test set shall never be touched until a single time, at the very end. In our image dataset, wear
particles are divided into seven classes: normal, cutting, spherical, severe sliding, fatigue laminar, chunky spalling and nonferrous metal particles. There are a total of 160 images for every kind of particles, among which 100 are labeled images in the training set, 30 are in the validation set, and 30 are in the test set.

![Figure 8. Examples of wear particles generated in tests.](image)

The training of a DCNN model normally requires numerous labeled images to achieve a better performance. For this reason, transfer learning (TL), particularly fine-tuning from a pre-trained classification network, is a common approach in a classification network.\[^{26}\] It has also been proved that transferring features even from distant tasks can be better than using a random initialization.\[^{27}\] Therefore, in our experiments, the DCNN model is based on a pre-trained network by ImageNet (contains 1000 categories and 1.2 million images) and is fine-tuned on the image dataset for different CONV phases using TL. In this study, the dense net was redesigned for wear classification tasks, the specific operation steps are as follows: Add a fully connected network to a pre-trained network (base network); Freeze the base network; Training the added part; Unfreeze CONV phases of base network according to the order from the bottom to the top; Fine-tune the unfroze CONV phases until the whole network.

To increase the diversity of the dataset, each image fed into the deep learning model is randomly cropped from the original image or its horizontal-flipped version, which is called data augmentation. The idea is to guarantee the network can generalize well in different situations and robust to small image transformations. Dropout is to discard some output features of this layer randomly (the output is set to 0) during training. Dropout is an effective technique to prevent over-fitting, which is mainly used in the full connection layer.

In these experiments, hyper parameters were set as follows: For input data, the pixel size of each image was converted to the input size for different DCNN models shown in table 1. As mentioned above, input image was random cropped from reshaped image or its horizontal flip. For training, Softmax was used as a classifier in the output layer and RMSprop was used as an optimizer. Dropout rate was set to 0.5. The cross-entropy loss function is selected to measure the difference between predicted value and expected value of DCNN models. The BN parameter was fixed, and the batch size was set to 10. The learning rate was set to 10-5, and an exponential decay is applied to the learning rate. The proposed DCNN models were implemented based on the tensorflow framework using a computer platform with 2 Intel Xeon 6130 CPU, 64 GB of RAM, and 2 RTX 2080Ti GPU.\[^{28}\]

4. Results and discussion

During the experiments, we mainly adopt three evaluation metrics, namely, accuracy, loss and time cost.

The accuracy is traditionally described as follows:
In the formula, \( N_c \) is the number of correctly predicted images and \( N_{\text{all}} \) is the number of all images. While the cross-entropy loss is calculated as:

\[
\text{Loss} = -\log \left( \frac{\exp(S_{y_i})}{\sum_j \exp(S_j)} \right)
\]  

(4)

In the formula, the label \( y_i \) specifies the index of the correct class if \( i \)-th input. The final score vector is \( S \) and \( S_j \) is the score for the \( j \)-th class. The time cost is the processing time of each step.

Five DCNN models were trained based on the images of wear particles generated in tests. The batch size was set to 10, step per epoch was set to 70 and epoch was set to 50. The number of epochs indicates how many times the dataset have been iterated. Therefore, each model was iterated 3500 times. After each epoch, the model was tested using the data on validation set, hence the validation loss and accuracy were recorded by epoch. The validation-loss evolution of the five models is demonstrated in figure 9 and the validation-accuracy evolution is demonstrated in figure 10. From the curve, one can see that the loss rate is decreasing, and the accuracy rate is increasing, suggesting the learning process is indeed on-going.

From the curve, one can see that the convergence happens fast, the loss and accuracy of the model trend to be stable after 30 epochs of training. The performance of AlexNet is much worse than the other four models, its loss rate stabilizes at around 0.6, and its accuracy rate stabilizes at around 70% which is far better than random guess. The best performing model is obviously DenseNet121, its loss rate is reduced to about 0.3, and its accuracy rate stabilizes at around 90% finally. One can see that the loss rate of ResNet50 is the largest at the beginning of training, and the reason may be that the model learns a bad pattern, which leads to the increasing of calculation cost.

![Figure 9. The validation-loss evolution curve.](image)

More information about the accuracy rate and the time cost of different DCNN models are shown in table 2. With the exception of AlexNet, the other four DCNN models are fine-tuned for different CONV phases using TL. The reason is the number of layers in AlexNet is very small and the calculation cost is not high. The dense net of all models is redesigned for wear classification tasks. All dense networks are designed as two fully-connected layers, and the design of more layers leads to the increase of computational complexity and the decrease of accuracy. The number of units in the last
fully-connected layer is 7 which is the category of wear particles, and the number of units in the first fully-connected layer is optimized through experiments.

![Figure 10. The validation-accuracy evolution curve.](image)

**Table 2.** Accuracy rate and time cost of different DCNN models.

| DCNN models | Fine-tune layers          | Time cost per step (ms) | Validation accuracy (%) | Test accuracy (%) |
|-------------|---------------------------|-------------------------|-------------------------|------------------|
| AlexNet     | the whole network (fc-512+fc-7) | 213ms                   | 76.00%                  | 74.60%           |
|             | dense net (fc-256+fc-7)    | 217ms                   | 75.00%                  |                  |
|             | phase 5                   | 217ms                   | 81.00%                  |                  |
|             | phase 4                   | 219ms                   | 83.00%                  |                  |
|             | phase 3                   | 217ms                   | 85.00%                  |                  |
|             | phase 2                   | 218ms                   | 83.00%                  |                  |
| VGG16       | the whole network         | 217ms                   | 84.00%                  | 83.99%           |
|             | dense net (fc-512+fc-7)    | 220ms                   | 70.00%                  |                  |
|             | phase 5                   | 224ms                   | 85.00%                  |                  |
|             | phase 4                   | 226ms                   | 88.00%                  |                  |
|             | phase 3                   | 227ms                   | 89.00%                  |                  |
| InceptionV3 | the whole network         | 228ms                   | 88.00%                  | 82.19%           |
|             | dense net (fc-512+fc-7)    | 117ms                   | 44.00%                  |                  |
|             | phase 5                   | 120ms                   | 89.00%                  |                  |
|             | phase 4                   | 123ms                   | 89.00%                  |                  |
|             | phase 3                   | 125ms                   | 91.00%                  |                  |
| ResNet50    | the whole network         | 126ms                   | 92.00%                  | 87.99%           |
|             | dense net (fc-512+fc-7)    | 118ms                   | 72.00%                  |                  |
|             | phase 5                   | 125ms                   | 86.00%                  |                  |
|             | phase 4                   | 129ms                   | 89.00%                  |                  |
|             | phase 3                   | 139ms                   | 93.00%                  |                  |
| DenseNet121 | the whole network         | 153ms                   | 92.00%                  | 88.39%           |
From table 2, it is clear that the accuracy of DenseNet121 model is 88.39% which is the largest accuracy rate on the test set. ResNet50 has the same level classification accuracy as DenseNet121, slightly worse in fact. In addition to AlexNet, the accuracy of other models is more than 80%, because the learning effect will be better as the number of layers increases. From the change of fine tune layers, the accuracy of the unfroze dense net is relatively low on the validation set. With the increase of unfroze CONV phases, the accuracy on the validation set gradually increases and tends to be stable. In the model, the layer closer to the top extracts local and highly general features, such as edge, colour and texture, while the layer closer to the bottom extracts more abstract features. The base network pretrained by ImageNet cannot be directly used for classification of wear particles, but using it to transfer learning can greatly improve learning efficiency.

The time cost per step in training of each DCNN models also represents in table 2. We can see that ResNet50 and DenseNet121 are more efficient than VGG-16 and InceptionV3. The reason behind its efficiency is that the residual connections serve as a gradient highway. But with the development of hardware, the time cost of training small dataset maybe neglected by designer. For instance, fine-tune the whole VGG16 cost about 50 minutes which is actually acceptable even if it spend two times longer than DenseNet121.

In summary, DenseNet121 is the best DCNN model evaluated in all three metrics to classify wear particles in ferrography images, making it a promising architecture for real time training.

5. Conclusion
The intelligent classification of wear particles provides important clues for the identification of wear condition and wear mechanism. Through a series of experiment, not only a deep convolutional neural network structure suitable for real-time detection of wear particles can be established, what’s more important is that the training and testing procedure proposed, also a dataset including seven kinds of wear particles is built which can be valuable for future related research.

A conclusion can be drawn that deep convolution neural network is suited for wear particles’ intelligent classification. Using transfer learning and proper data augmentation methods, AlexNet, VGG16, InceptionV3, ResNet50 and DenseNet121 all achieve high accuracy. In particular, DenseNet121 is proved to be the most efficient model. The experimental results show that the accuracy of DenseNet121 on the test set is 88.39%.

Instead of manually designing and selecting the features, the proposed DCNN can automatically learn the features through a layer-wise representation and realize a classification of the seven types of wear particles in the ferrography images. The proposed DCNN realizes an end-to-end processing, that is, from the original image to the classification results of different types of wear particles, avoids the accumulation and transmission of errors caused by numerous steps applied in a traditional linear process, and improves the efficiency and accuracy of wear particle analysis.

In further studies, more images of various wear particles should be collected to make a bigger dataset and instance segmentation of wear particles shall draw more attention.

Acknowledgments
This study was financially supported by the funds of the Shandong Provincial Key Research and Development Program, China (Grant no. 2018GGX105002).

References
[1] Kumar, Manoj, Parboti Shankar Mukherjee, and Nirenda Mohan Misra. "Advancement and current status of wear debris analysis for machine condition monitoring: a review." Industrial Lubrication and Tribology 65.1 (2013): 3-11.
[2] Upadhyay, R. K. "Microscopic technique to determine various wear modes of used engine oil." Journal of microscopy and ultrastructure 1.3 (2013): 111-114.
[3] Ebersbach, S., Z. Peng, and N. J. Kessissoglou. "The investigation of the condition and faults of a spur gearbox using vibration and wear debris analysis techniques." Wear 260.1-2 (2006):
16-24.

[4] Roylance, B. J. "Ferrography—then and now." Tribology International 38.10 (2005): 857-862.

[5] Eliaz, Noam, and Ronald M. Latanision. "Preventative maintenance and failure analysis of aircraft components." Corrosion Reviews 25.1-2 (2007): 107-144.

[6] Sperring, T. P., and T. J. Nowell. "SYCLOPS—a qualitative debris classification system developed for RAF early failure detection centres." Tribology international 38.10 (2005): 898-903.

[7] Peng, Zhongxiao. "An integrated intelligence system for wear debris analysis." Wear 252.9-10 (2002): 730-743.

[8] Peng, Z., and T. B. Kirk. "Automatic wear particle classification using neural networks." Tribology Letters 5.4 (1998): 249-257.

[9] Stachowiak, Gwido P., Gwido W. Stachowiak, and Pawel Podsiadlo. "Automated classification of wear particles based on their surface texture and shape features." Tribology International 41.1 (2008): 34-43.

[10] Stachowiak, G. W., and P. Podsiadlo. "Towards the development of an automated wear particle classification system." Tribology International 39.12 (2006): 1615-1623.

[11] Stachowiak, G. P., Pawel Podsiadlo, and G. W. Stachowiak. "A comparison of texture feature extraction methods for machine condition monitoring and failure analysis." Tribology Letters 20.2 (2005): 133-147.

[12] Wu, Tonghai, et al. "Imaged wear debris separation for on-line monitoring using gray level and integrated morphological features." Wear 316.1-2 (2014): 19-29.

[13] Gu, Daqiang, Lixia Zhou, and Jing Wang. "Ferrography Wear Particle Pattern Recognition Based on Support Vector Machine." Zhongguo Jixie Gongcheng(China Mechanical Engineering) 17.13 (2006): 1391-1394.

[14] Gonçalves, Valdeci Donizete, Luis Fernando de Almeida, and Mauro Hugo Mathias. "Wear Particle Classifier System Based on an Artificial Neural Network." Strojniški Vestniki/Journal of Mechanical Engineering 56.4 (2010).

[15] Li, Qiong, et al. "Ferrography wear particles image recognition based on extreme learning machine." Journal of Electrical and Computer Engineering 2017 (2017).

[16] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436-444.

[17] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[18] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[19] Peng, Peng, and Jiugen Wang. "Wear particle classification considering particle overlapping." Wear 422 (2019): 119-127.

[20] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[21] He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." Proceedings of the IEEE international conference on computer vision. 2015.

[22] Sze, Vivienne, et al. "Efficient processing of deep neural networks: A tutorial and survey." Proceedings of the IEEE 105.12 (2017): 2295-2329.

[23] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

[24] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[25] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[26] Ahmed, Amr, et al. "Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks." European Conference on Computer Vision. Springer, Berlin, Heidelberg, 2008.

[27] Yosinski, Jason, et al. "How transferable are features in deep neural networks?." Advances in neural information processing systems. 2014.

[28] Abadi, Martin, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016).