A Comparative Study of Convolutional Neural Network Architectures for Ferrograph Image Classification

Vedmani Vaidya
Department of Mechanical Engineering
Vishwakarma Institute of Technology
Pune, India
vedmani.vaidya19@vit.edu

Dr. Sangita Jaybhaye
Department of Computer Engineering
Vishwakarma Institute of Technology
Pune, India
sangita.jaybhaye@vit.edu

***Abstract*— Classification of ferrography images, or wear particle images, is an important task in the field of predictive maintenance of mechanical systems (Tribosystems). In this study, we compared the performance of several popular convolutional neural network (CNN) architectures for wear particle image classification. The architectures we considered included VGG, ResNet, InceptionV3, DenseNet, Xception, MobileNet, MobileNetV2, EfficinetNet and ConvNext- T. Small dataset consisting of 5 classes of wear particles was used for the purpose of fine tuning and performance evaluation of CNN architectures. Performance of each convolutional neural network architecture was evaluated using validation accuracy, F-score, computational efficiency. Results indicated that certain CNN architectures are better suited for ferrography image classification than others. In particular, we found that the EfficientNet and DenseNet architectures outperformed the other architectures considered in this study with validation accuracy of 99.43% and 98.86% respectively. In conclusion, our study provides valuable insights into the suitability of different CNN architectures for wear particle classification and highlights the potential of the EfficientNet and DenseNet architectures in this domain.**

Keywords*— *Convolutional Neural Network, Transfer Learning, Ferrography, Wear Particles, Image Classification.

I. INTRODUCTION

Machine and component failures are frequently caused by wear and tear. Wear particle analysis is a powerful tool for understanding the underlying causes of wear and tear on mechanical components. Wear particles produced by friction pairs contain useful information about wear rate, severity, and mechanism. As a result, wear debris analysis is acknowledged as an efficient method for monitoring machine wear condition and fault detection (B. Fan et al., 2017; Cao et al., 2018).Ferrograph image analysis is a highly effective method for providing particle characteristics information. Machine condition monitoring can be performed using ferrography, which provides a wealth of information about wear particles, including particle concentration and wear type.

Ferrographic analysis, which involves collecting and analyzing oil samples to monitor the condition of lubricated systems, is a well-established technique. By examining the properties, chemical composition, and presence of wear particles and contamination in the oil, it is possible to determine the mode, source, and severity of wear in a system. Ferrographic images can be particularly useful in this process, as they allow for detailed analysis of wear particles. (Fan et al., 2018; Wu et al., 2014; Henneberg et al., 2015; Cao et al., 2014). Many approaches are reported based on particle feature extraction to develop relationships between particle types and wear mechanisms.

Historically, the analysis of ferrographic images has involved manually examining them through visual inspection, a process that can be time-consuming and requires specialized expertise. Particle feature extraction is the foundation of classification of wear particles and is used as input for wear debris classifiers The process of manually extracting features from ferrograph images typically involves four steps: collecting a sample of particles, extracting features from the images, establishing a database, and processing the feature information. This process can be complex and time-consuming. Convolutional neural networks (CNNs) have emerged as a promising approach for automating the analysis of wear particles, allowing for faster and more accurate analysis.

Initiatives to automate the study of wear debris has been undertaken over the years. Thomas et al. (1991) attempted to automate the classification of wear particles by calculating various features such as area, aspect ratio, roundness, skewness, and dispersion. Kirk et al. (1995) also developed numerical parameters, including area, perimeter, convex area, convexity, and fiber length, to describe the morphology of wear particles.

Grey system theory was applied by (Z. Peng & Kirk, 1999) to classify six types of wear particles using three dimensional images, showing that this approach combined with fuzzy logic can be effective for wear particle classification. (Myshkin et al., 1997) introduced the possibility of application of neural network with Fourier descriptor for classification of wear debris.

(Wang et al., 2019) proposed a two-level strategy, incorporating a BP neural network and a 6-layer CNN model, for classifying five types of wear particles. Other traditional methods of identifying wear debris may struggle to classify particles that are stuck together. (P. Peng & Wang, 2019b) addressed this issue by developing a CNN model with InceptionV3 and fully connected layers that can effectively classify overlapping particles.

An automatic process for detecting and classifying wear was developed by (Y. Peng et al., 2020) using cascade of two CNNs and a support vector machine classifier decreasing the compute and increasing the accuracy when compared to R-CNN, Faster R-CNN. Hybrid convolutional neural network with Transfer learning (TL) was used by (Y. Peng et al., 2019) to reduce the reliance on the number of samples, along with

support vector machine (SVM) to enhance the accuracy of automatic wear particle classification.

A convolutional neural network (CNN) model called FECNN was proposed by (P. Peng & Wang, 2019a) to identify wear particles. The CNN model proposed included dropout operations to prevent overfitting and 1-D convolutional operations to reduce the number of parameters in the model.

A small dataset and different wear particle sizes make it difficult to classify ferrograph images. To cope with the problem of insufficient samples a data augmentation algorithm based on the permutation of image patches was developed by (P. Peng & Wang, 2020) along with custom feature extraction loss function to extract more abundant features and to reduce redundant representations.

(H. Fan et al., 2020) proposed a recognition method for small sample ferrographic images that leverages CNN and transfer learning, using a virtual ferrographic image dataset. The authors used the AlexNet frame and studied the impact of various CNN internal factors, such as network structure, convolution parameters, activation function, optimization mode, learning rate, and L2 regularization, on the performance of the model. (Xie & Wei, 2022) focused on problems like the wear particle image's fuzzy edges and complex surface texture, which have an impact on the computer model's accuracy and feature extraction and developed a multichannel image encoder. This encoder enhanced the visible edge and surface characteristics of the image, and the resulting model (called MCECNN) was created by linking the encoder with a convolutional neural network.

Ferrograph images are often blurry due to lubricant contamination and limited optical field depth. To address this issue, (H. Wu et al., 2019) developed a restoration method to mitigate the impact of out-of-focus blurring. The degradation model was created using larger convolutional kernels to accurately capture the blurring process, and a pixel-by-pixel regression procedure was used to perform the final restoration using a trained network. This proposed method is more computationally efficient than traditional deblurring techniques, making it suitable for use in online applications.

(Wang et al., 2021) utilized 3D topographical information of wear particles to create a knowledge-embedded double CNN model for classifying fatigue and severe sliding wear particles. The first CNN was constructed using a 2D height map of 3D wear particles, and to further improve performance, prior knowledge of particle properties was used to identify four effective kernels based on the physical wear mechanism of the two types of particles. The second CNN network was then built using the feature maps of these four kernels and six parallel convolution layers to objectively identify particles.

(More & Jaybhaye, 2022) utilized a web-based tool called a teachable machine to create a convolutional neural network (CNN) model to evaluate the health of a lubricated system as part of condition monitoring. The developed CNN was able to recognize five types of wear particles with an accuracy of 95.4%. (Jia et al., 2020) conducted a study to evaluate the potential of deep convolutional neural network architectures for the identification of wear particles in condition monitoring systems. In this study, the authors employed five different CNN models: AlexNet, VGG16, InceptionV3, ResNet50, and DenseNet121, and evaluated their performance on a dataset containing seven types of wear particles. The results showed that DenseNet121 achieved the highest accuracy of 88.39%, indicating that DCNNs can be effectively used for wear particle classification.

In our study, we investigate the compatibility of several convolutional neural network (CNN) architectures for the identification of wear particles. We focused on deep popular convolutional neural networks and employed a transfer learning with fine-tuning approach, using pre-trained CNN models that were trained on the ImageNet dataset (Deng et al., 2009), one of the most extensive and commonly used datasets for image classification. In particular, we focused on popular DCNN models, such as VGG, ResNet, InceptionV3, MobileNet, MobileNetV2, Xception, DenseNet, EfficientNet, and ConvNext. In order to tailor these architectures to the task of wear particle classification, we fine-tuned them on a specific dataset of wear particles. Dataset used for tuning is the same dataset described in (More & Jaybhaye, 2022). Our findings indicate that the pre-trained CNN models, can effectively be used for the purpose of wear particle classification, providing further evidence for the potential of CNNs in condition monitoring applications. Overall, our study builds upon the work of (Jia et al., 2020) and (More & Jaybhaye, 2022), and provides additional support for the use of CNN architectures for wear particle classification.

We provide a detailed description of our experimental setup, including the dataset and evaluation metrics used, as well as the results of our experiments and a discussion of their implications, in following sections of the paper.

II. Dataset

Dataset consists of 979 images of five types of wear particle images obtained through ferrography (More & Jaybhaye, 2022), including cutting wear, normal rubbing wear, a combination of normal rubbing wear and cutting wear, red oxide, and severe sliding wear. The dataset is divided into three parts including a training data, a validation data, and a test data for model training and evaluation. Out of the total 979 images, 100 are designated as test data, 175 images (20% of the total) are randomly chosen for the validation set, and the remaining 704 images are used to train or fine-tune the CNN model. The distribution of images in the dataset can be found in the table 1.

Table 1 Class Representation in the Dataset"

Type of wear particle	Train data	Validation data	Test data
Cutting wear	81	27	12
Normal rubbing wear	68	12	10
Normal rubbing and cutting wear	230	54	33
Red oxide	46	14	9
Severe sliding wear	279	68	36

III. Transfer learning and Fine tuning

Transfer learning involves adapting a model that has already been trained on one task to a new, related task. In this study, we used transfer learning with fine-tuning to improve the performance of our models for the classification of wear particles in condition monitoring systems. Transfer learning is particularly useful when the dataset for the new task is small or limited, as is often the case in these types of systems. By using a pre-trained model as the starting point for training a new model on a different dataset, transfer learning reduces the amount of data and computation required and can improve the performance of the new model. The pre-trained model, typically a deep neural network, has already learned many

useful features that can be transferred to the new task, and using a pre-trained model that has been trained on a very large dataset may be more effective than training a new model from scratch. In this study, all of our models were pre-trained on the ImageNet dataset (Deng et al., 2009).

Fine-tuning approach allows a pre-trained model to be used for use on a different, but related, task. In the case of wear particle classification, which typically has a small dataset, transfer learning with fine-tuning can be used to adapt a pre-trained convolutional neural network (CNN) model for wear particle classification. After using a pre-trained model as the starting point for training a new model on the target dataset, the weights of the pre-trained model can be adjusted through fine-tuning to better fit the new data. This process, known as transfer learning with fine-tuning, can improve the generalizability of machine learning models. By using pre-trained models that have been trained on large, diverse datasets, transfer learning can help to ensure that the new model is able to handle a wide range of data and is not overly specialized for a specific dataset. This can help to improve the model's performance and make it more applicable to real-world situations.

The output layers of all architectures are modified to match the number of classes in the dataset in order to adapt the pre-trained model to the task of wear particle classification, and the weights of this layer are randomly initialized. Output layer of all the architectures including fully connected layers (if present) is replaced with dense layer with 5 units matching the number of classes in our dataset. The rest of the model is then frozen, just the output layer is trained on the train dataset, by re-training the model on the training data, using a lower learning rate to prevent the pre-trained model from forgetting the features it has already learned.

IV. Hardware

For our experiments, we used a NVIDIA Tesla T4 GPU with 15109 MiB of memory, and a high-performance Intel(R) Xeon(R) CPU @ 2.00GHz with 2 cores and 2 threads per core.

V. Convolutional neural network architectures

This section will concentrate on the various deep learning network architectures that have been used in this research and are significant in the field of computer vision, including VGG, ResNet, InceptionV3, MobileNet, MobileNetV2, Xception, DenseNet, EfficientNet, and ConvNeXt. We will delve into the details of each architecture, including its underlying principles and key features.

A. VGG

The VGG architecture developed by (Simonyan & Zisserman, 2014) achieved state-of-the-art accuracy on ILSVRC localization and classification tasks. VGG architecture is a very simple and uniform CNN architecture consisting of multiple convolutional and pooling layers followed by a few fully connected layers. When processing an input image of size 224 x 224 in RGB format, the VGG architecture subtracts the mean RGB value, which is calculated from the training set, from each pixel. The image is then passed through a series of convolutional layers that utilize 3 x 3 filters for feature extraction. these filters have a small receptive field, the network can extract detailed features from the input image. Spatial pooling is performed using five max pooling layers, and spatial padding is applied after each convolution operation to preserve spatial resolution. Throughout the network, the VGG architecture employs the same convolutional layer configuration, including the same number and size of filters. There are multiple versions of the VGG architecture, we focused on the VGG16 and VGG19 architectures, which have 16 and 19 weight layers, respectively.

B. ResNet

The ResNet convolutional neural network architecture, proposed by (He et al., 2016) proposed ResNet convolutional neural network (CNN) architecture winning 1st place on the ILSVRC 2015 classification task. When deep neural networks begin to converge, a problem is observed: as network depth increases, accuracy becomes saturated and degrades rapidly, and the addition of more layers leads to higher training error. ResNet architecture addresses this issue by introducing a deep residual learning framework. The bottleneck block (figure 1) introduced in ResNet is a type of residual block that lowers the number of filters in the convolutional layers, thereby reducing the network's computational complexity without sacrificing accuracy. This is achieved by using a 1x1 convolutional layer to reduce the number of filters, followed by a 3x3 convolutional layer, and then another 1x1 convolutional layer to restore the original number of filters. One or more convolutional layers, batch normalisation, and a rectified linear unit (ReLU) activation function are all present in each residual block. The output of the block is then combined with its input using a shortcut or skip connection. Instead of attempting to model the entire mapping from input to output, this enables the network to learn residual functions, or the differences between the input and the desired output.

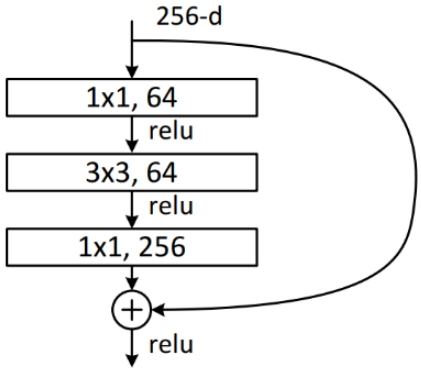

Figure 1 “Bottleneck” building block for ResNet (He et al., 2016)

The ResNet architecture is available in a number of variants, including ResNet50, ResNet101, ResNet152, and ResNet169. We took into account ResNet50, ResNet101, and ResNet152 for this study. The network's weight layers are indicated by the number in the name. ResNet50, for instance, has 50 layers of weight, whereas ResNet152 has 152 layers.

C. InceptionV3

InceptionV3 architecture was proposed by (Szegedy et al., 2016) as an upgrade to original Inception (GoogleNet) architecture. InceptionV3 incorporates several improvements over previous version of the Inception architecture including improvements proposed in InceptionV2. InceptionV3 replaces 5 x 5 convolutions used in Inception architecture with two 3 x 3 convolutions (figure 2 (a)). Although there are fewer parameters, the receptive field stays the same.

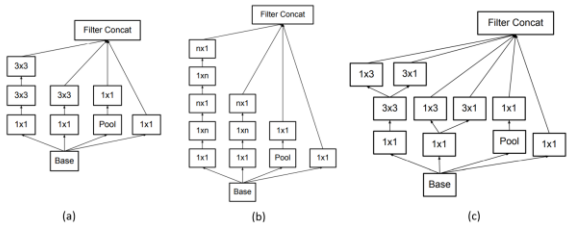

Figure 2 InceptionV3 building blocks (Szegedy et al., 2016)

The InceptionV3 architecture uses a combines 1 x n and n x 1 convolutions (figure 2(b)) to factorize larger convolutions and reduce the model's computational complexity. Additionally, to avoid the "representational bottleneck" that can happen when using deep neural networks, the filter banks in the InceptionV3 architecture have been expanded (made wider rather than deeper) (see figure 2(c)). Inceptionv3 incorporates all the discussed improvements along with use of RMSProp optimizer, factorized Batch Normalization, 7 × 7 convolutions, and Label Smoothing.

D. Xception

Xception is a convolutional neural network (CNN) developed by (Chollet, 2017) based entirely on layers of depth-separable convolution. It is a variant of Inception architecture and is designed to improve upon Inception's performance. Figure 3 illustrates how the Xception architecture uses 1 × 1 first map the cross-channel correlations of each output channel before mapping each output channel's spatial correlations separately. In the Xception architecture, the network's feature extraction base is made up of 36 convolutional layers. All 14 of the modules that make up the 36 convolutional layers, with the exception of the first and last, have linear residual connections surrounding them. The Xception architecture can be described as a series of depthwise convolution layers with residual connections arranged in a linear stack.

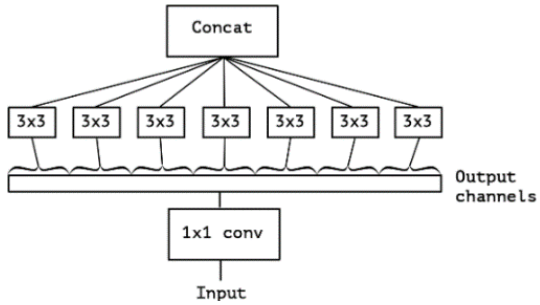

Figure 3 Xception module(Chollet, 2017)

E. MobileNet

MobileNet is a convolutional neural network architecture developed by (Howard et al., 2017) designed for efficient use on mobile and embedded devices. MobileNet architecture is based on use of depthwise separable convolutions. With MobileNets, each input channel is subjected to a single filter as part of depthwise convolution, which is followed by pointwise convolution, which combines the results of depthwise convolution using a 1x1 convolution, drastically reducing model size and computation. MobileNet consists of a series of 3x3 depthwise separable convolutional layers, followed by batch normalization and a ReLU nonlinearity, followed by 1x1 convolution (pointwise convolution) with batch normalization and ReLU. Two new global hyperparameters introduced by MobileNet that let the model builder trade off between accuracy and latency. This makes it possible to adapt the model to the particular constraints of the current problem. If depthwise and pointwise convolutions are counted as separate layers, MobileNet has 28 layers. Two hyperparameters, Width Multiplier and Resolution Multiplier, are left at their default values of 1.0 and 1.0 for this study.

F. DenseNet

One of the main concepts behind DenseNet is the incorporation of dense connections between layers. In a traditional convolutional neural network, each layer only receives input from the preceding layer. However, in DenseNet, each layer is connected to every other layer in a sequential manner. This means that each layer utilizes the feature maps of all previous layers as inputs, and its own feature maps are used as inputs for all subsequent layers. This allows the network to learn features at various scales, which can enhance the model's performance. (Huang et al., 2017). DenseNet block implements a non-linear transformation using Batch Normalization (BN), rectified linear units (ReLU), Pooling, or Convolution (Conv). Layers between two dense blocks are called transition layers, which perform batch normalization, 1×1 convolutional operation and 2×2 average pooling operation. DenseNet introduces hyperparameter called growth rate (k). Growth rate refers to the number of filters added to the output of a convolutional layer each time a new convolutional layer is added to the network. For this study we have used DenseNet121, DensNet169, DenseNet201 with growth rate of 32. The number of layers and the number of filters per layer vary between these versions.

G. MobileNetV2

MobileNetV2 (Sandler et al., 2018) expands on the concepts proposed in MobileNetV1 architecture. Similar to MobileNet MobileNetV2 uses depthwise separable convolution as efficient building blocks. Two new features are introduced to the architecture by MobilnetV2: (1) linear bottlenecks between the layers, and (2) shortcut connections between the bottlenecks. The foundation of MobileNetV2 is an inverted residual structure, where the connections between the bottleneck layers are the residual connections. The residual connections in MobileNetV2's inverted residual structure are those between the bottleneck layers. The MobileNetV2 architecture consists of a 32-filters in initial fully convolution layer, followed by 19 additional bottleneck layers. Kernels of 3 × 3 size are used, as is the standard for CNN. MobileNetV2 significantly improves on MobileNetV1 and advances the state-of-the-art in mobile visual recognition, including classification, object detection, and semantic segmentation.

H. EfficientNet

The key idea behind EfficientNet is the concept of model scaling, which involves scaling up the size of a neural network in a systematic and principled way. This is in contrast to traditional approaches, which often involve simply increasing the number of layers or the number of filters in each layer without considering the overall architecture of the network.(Tan & Le, 2019). One of the ways that EfficientNet achieves better performance is by using a compound scaling method, which involves simultaneously scaling up the dimensions of the network (such as the width and depth), as well as the resolution of the input images. This allows the network to maintain a similar level of complexity while handling higher-resolution images, which can improve

performance on tasks such as image classification. If 2^N times more computational power available, then network depth is increased by α^N, width by β^N and image size by γ^N, where α, β, γ are determined by small grid search on original model. Another key aspect of EfficientNet is its use of a novel building block called a MBConv block, which stands for mobile inverted bottleneck convolution. It uses depthwise separable convolutions like MobileNet, which can reduce the number of parameters and computations required by the network, and it also includes an inverted bottleneck structure similar to MobileNetV2, which can further improve the efficiency of the network. Squeeze-and-excitation optimization with MBConv block is used to create baseline model EfficientNetB0 which is further scaled according to requirement using compound scaling method.(Tan & Le, 2019)

I. ConvNeXt

ConvNeXt, is a family of convolutional neural network architectures that are designed to compete with vision Transformers (ViTs) in terms of accuracy and scalability (Liu et al., 2022). Facebook AI Researchers and UC Berkeley researchers Liu, Mao, Wu, Feichtenhofe, Darrell, and Xie proposed the ConvNext convolutional network architecture. Training Techniques, Macro Design, ResNeXt-ify, Inverted Bottleneck, Large Kernel, and Micro Design are just a few of the crucial components that the authors found to be essential in "modernising" a standard ResNet toward the Vision Transformer architecture and explaining the performance difference. For model training, authors used the AdamW optimizer, regularisation techniques like Stochastic Depth and Label Smoothing, and data augmentation methods like Mixup, Cutmix, RandAugment, and Random Erasing. In Macro Design of ConvNext architecture, following the Swin Transformer design, the number of blocks in ResNet50 at each stage was modified to (3, 3, 9, 3). In the modified ResNet model, the stem cell has been replaced with a "patchify" layer, which is implemented using a 4×4 convolutional layer with stride 4. The width of the network has also been increased to match the number of channels in Swin-Transformers (96), following the strategy proposed in ResNeXt. The original stem cell consisted of a 7×7 convolution layer with stride 2 followed by a max pool. ConvNext, like the MobileNetV2 architecture, utilizes inverted bottlenecks. ConvNext makes use of 7 × 7 convolutions. ConvNext employs GELU (Gaussian Error Linear Unit) activation, which is used in advanced transformers like as BERT, GPT-2, and ViT's. Other differences from ordinary convnets include the use of layer normalisation, independent downsampling layers, etc. ConvNext- T version trained on ImageNet- 1K dataset is used for this study.

VI. MODEL CONFIGURATION AND TRAINING

In this section we will discuss about the overall network architecture used to fine tune the CNN architectures discussed for wear particle classification along with training configuration.

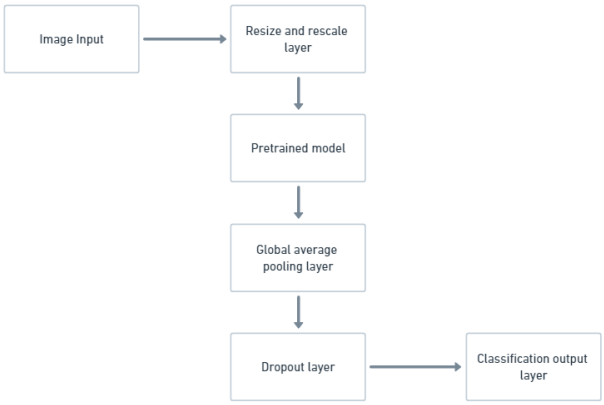

Figure 4 Process flow of overall model architecture for fine tuning

The image is first passed through a resize and rescale layer, where it is reshaped to a 224x224x3 shape and normalized by dividing every pixel value by 255. As a result, the pixels' values are changed from having a range of 0 to 255 to 0 to 1. The normalized image is then passed to the base model architecture with the top classification layer removed. A global average pooling layer receives the output of the base model instead of fully connected layers. A dense classification layer (output layer), which assigns the image to one of the dataset's classes, receives the output from the average pooling layer after passing it through a dropout layer.

Multiple deep learning architectures are used as pretrained models, including VGG16, VGG19, ResNet50, ResNet101, ResNet152, InceptionV3, DenseNet121, DenseNet169, DenseNet201, Xception, MobileNet, MobileNetV2, EfficientNetB0, and ConvNext-T and fine-tuned by the process shown in figure 4.

Global average pooling is a technique used in convolutional neural networks (CNNs) to reduce the dimensionality of the data (Lin et al., 2013). This method uses feature maps produced by convolutional layers instead of fully connected layers on top of these feature maps, allowing the model to retain important features learned by the convolutional layers while also reducing the number of required parameters and computational resources. This replaces the need for fully connected layers, allowing for a more efficient and effective model. Global average pooling has several advantages over fully connected layers, including improved interpretability of the feature maps and reduced risk of overfitting. Output from global average pooling layer is fed directly into the dropout layer.

Dropout is a method of regularization in neural networks that helps to prevent overfitting (Srivastava et al., 2014). It randomly drops units (and their connections) from the network during training, which prevents the units from becoming too co-adapted. The dropout rate is the fraction of units that are randomly dropped during training. In our experiment, we used a dropout rate of 0.2. The dropout layer helps to improve the generalizability of the model by preventing the units from becoming too specialized to the training data.

Following the dropout layer, the output is passed into a dense layer with 5 neurons with a softmax activation. The dense layer uses the output from the dropout layer to classify the image into one of the 5 classes of the dataset. The softmax activation is commonly used in the output layer of a classification model because it ensures that the predicted probabilities for each class sum to 1, allowing the model to make a probabilistic prediction for each class. In our experiment, we used a softmax activation in the final dense

layer to enable the model to make probabilistic predictions for each of the 5 classes in the dataset.

To track the progress of our experiments and model versions during training, we utilized the Weights & Biases (wandb) library (Biewald, 2020). This allowed us to easily log and visualize the training process, as well as keep track of the different versions of the models. Fine-tuned models are compared based on various performance metrics including training accuracy, validation accuracy, F1 score, precision, recall, number of parameters, and floating-point operations (FLOPs).

VII. Results

We demonstrate the effectiveness of the previously discussed CNN architectures for the wear particle classification task in this section. We compared the performance of VGG16, VGG19, ResNet50, ResNet101, ResNet152, InceptionV3, DensNet121, DenseNet169, DenseNet201, Xception, MobileNet, MobileNetV2, EfficientNetB0 and ConvNeXt- T based on different parameters like training accuracy, validation accuracy, number of parameters, and FLOPs.

EfficientNetB0 architecture showed best performance with validation accuracy of 99.43% followed by DenseNet201 architecture with 99.32% validation accuracy. ResNet architectures ResNet152, ResNet50 and ResNet101 achieved least validation accuracy of 56.57%, 57.71% and 64.57% respectively.

Table 2 Training and validation accuracy

Name	Training accuracy	Validation accuracy
EfficientNetB0	0.995739	0.994286
DenseNet201	1	0.993243
DenseNet121	0.997159	0.988571
DenseNet169	1	0.986486
MobileNet	1	0.982857
MobileNetV2	1	0.977143
InceptionV3	1	0.971429
ConvNeXt- T	0.992898	0.971429
Xception	0.99858	0.971429
VGG19	0.923295	0.902857
VGG16	0.9375	0.897143
ResNet101	0.646307	0.645714
ResNet50	0.643466	0.577143
ResNet152	0.639205	0.565714

Validation accuracy and training accuracy for discussed convolutional neural network (CNN) architecture are listed in table 2. The validation accuracy reflects the model's performance on a holdout dataset that was not used for training, while the training accuracy reflects the model's performance on the training data. Variation of training accuracy over 40 epochs can be observed in Fig. 5.

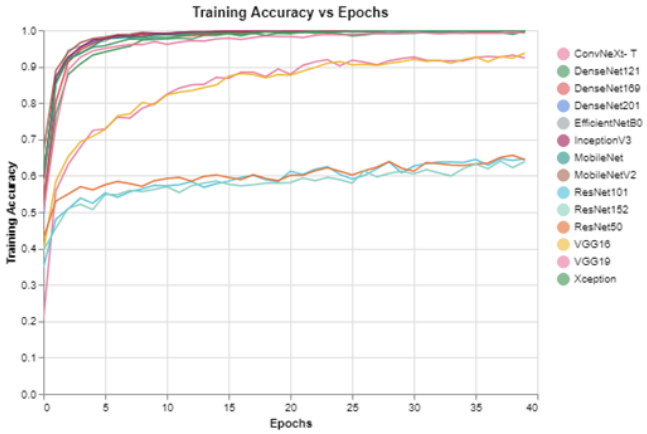

Figure 5 Training accuracy vs Number of epochs

EfficientNetB0, DenseNet201, DensNet121, DensNet169, MobileNet, MobileNetV2, InceptionV3, and Xception models have low training loss and validation loss, indicating that they are making accurate predictions on both the training data and the validation data. The VGG16, VGG19, ResNet50, ResNet101, and ResNet152 models, on the other hand, have higher training loss and validation loss, indicating that they may be struggling to make accurate predictions.

Table 3 lists the training loss and validation loss for convolutional neural network (CNN) architectures. The validation loss represents the model's error on a holdout dataset that wasn't used for training, whereas the training loss represents the model's error on the training data.

Table 3 Cross-entropy training and validation loss

Name	Training loss	Validation loss
EfficientNetB0	0.027881	0.05973
DenseNet201	0.015449	0.061714
DenseNet121	0.039728	0.058115
DenseNet169	0.0165	0.064547
MobileNet	0.013799	0.049526
MobileNetV2	0.00648	0.057727
InceptionV3	0.007701	0.06866
ConvNeXt- T	0.060722	0.105909
Xception	0.025288	0.083657
VGG19	0.292491	0.292816
VGG16	0.281643	0.342128
ResNet101	0.963095	0.974254
ResNet50	0.907582	0.952705
ResNet152	0.987763	1.030513

Variation of training loss over 40 epochs of training can be observed in figure 6.

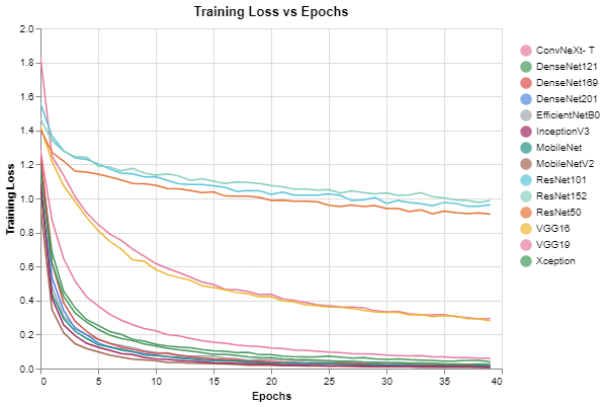

Figure 6 Training loss vs Number of epochs

FLOPs is a measure of the computational complexity of a model and is often used to compare the performance of different models on the same hardware. A model with a higher number of GFLOPs will generally be more computationally expensive to train and may require more powerful hardware to run efficiently. Table 4 lists number of FLOPs (in billion) for all the architectures.

The EfficientNetB0 model has lowest number of FLOPs, at approximately 0.06B. The MobileNetV2 and MobileNet models also have relatively low numbers of FLOPs, at approximately 0.3B and 0.57B, respectively. The DenseNet121 model has a slightly higher number of GFLOPs, at around 2.8. The InceptionV3, DenseNet169, and ResNet50 models have higher numbers of GFLOPs, with approximately 2.8, 3.4, and 3.9, respectively. The DenseNet201, ConvNeXt-T, and Xception models have even more FLOPs, with around 4.3B each. The ResNet101 and ResNet152 models require approximately 7.6B and 11.3B FLOPs. The VGG16 and VGG19 models also have a highest number of FLOPs, with approximately 15.4 and 19.5, respectively.

Table 3 FLOPs (in billion)

Name	GFLOPs
EfficientNetB0	0.06387
MobileNetV2	0.364
MobileNet	0.5678
DenseNet121	2.834
InceptionV3	2.842
DenseNet169	3.359
ResNet50	3.862257679
DenseNet201	4.290869839
ConvNeXt- T	4.487
Xception	4.554
ResNet101	7.577037839
ResNet152	11.29242011
VGG16	15.357
VGG19	19.519

Table 4 lists the number of parameters for convolutional neural network (CNN) architectures. The number of parameters reflects the amount of memory required to store the model's weights and biases.

Table 4 Parameters

Name	Parameters
MobileNetV2	2264389
MobileNet	3233989
EfficientNetB0	4055976
DenseNet121	7042629
DenseNet169	12651205
VGG16	14717253
DenseNet201	18331589
VGG19	20026949
Xception	20871725
InceptionV3	21813029
ResNet50	23597957
ConvNeXt- T	27823973
ResNet101	42668421
ResNet152	58381189

The MobileNetV2 model has lowest number of parameters, at approximately 2.2 million. The MobileNetV2 and EfficientNetB0 models also have relatively low numbers of parameters, at approximately 3.2 million and 4.1 million, respectively. The VGG16 and VGG19 models have higher number of parameters, with approximately 42.7 million and 58.4 million, respectively. ResNet152 model has highest number of parameters with 58.4 million parameters.

Table 6 presents the results of evaluation of CNN architectures on a test dataset of 100 images. The columns of the table provide various metrics of model performance, including the loss (crossentropy loss), accuracy (the percentage of correct predictions), precision (the percentage of true positive predictions), recall (the percentage of true positive cases that the model correctly identified), and F-score (a measure of the balance between precision and recall).

Table 6 Evaluation of architectures on Test data

Name	Loss	Accuracy	Precision	Recall	F- score (macro)
EfficientNet	0.0739	0.99	0.9946	0.9939	0.9942
DenseNet121	0.1085	0.99	0.9946	0.9939	0.9942
MobileNet	0.0886	0.99	0.9946	0.9939	0.9942
ConvNeXt-T	0.1023	0.99	0.9946	0.9939	0.9942
IncpetionV3	0.0659	0.98	0.9791	0.9883	0.9834
Xcpetion	0.1091	0.98	0.9895	0.9773	0.9828
MobileNetV2	0.0804	0.97	0.9834	0.955	0.9681
DenseNet169	0.1469	0.95	0.9735	0.9429	0.9565
VGG19	0.2666	0.95	0.9576	0.93	0.9424
DenseNet201	0.1683	0.94	0.9065	0.9636	0.9267

VGG16	0.3068	0.94	0.9541	0.8962	0.9212
ResNet101	1.0494	0.61	0.4147	0.4409	0.4183
ResNet152	1.1318	0.61	0.4424	0.4303	0.4167
ResNet50	0.9738	0.59	0.44	0.3646	0.3319

Architectures with highest F- score are EfficientNetB0, DenseNet121, MobileNet and ConvNeXt- T with F- score of 0.9942. These architectures performed particularly well on test data with with high accuracy, precision, and recall scores. Conversely, the models with the lowest F-scores were ResNet50, ResNet101, and ResNet152. These models had higher loss values and lower accuracy, precision, and recall, indicating that they made more errors and were less accurate in their predictions.

Upon comparing the validation accuracy and test accuracy of different CNN architectures, it was observed that the DenseNet201 and DenseNet169 models appeared to be overfitting on the validation data, as their performance on the test data was relatively poor compared to their validation accuracy. On the other hand, the EfficientNetB0 and DenseNet121 models demonstrated a good ability to generalize to unseen data, as it achieved a relatively high level of accuracy on the test set as well.

VIII. Conclusion

In This study aimed to identify the most effective convolutional neural network (CNN) architectures for wear particle classification and evaluate the effectiveness of transfer learning with fine-tuning for this task. A total of fourteen CNN architectures, namely VGG16, VGG19, ResNet50, ResNet101, ResNet152, InceptionV3, MobileNet, MobileNetV2, Xception, DenseNet121, DenseNet169, DensNet201, EfficientNetB0, and ConvNeXt, were fine-tuned on a small dataset of wear particle images.

The results showed that the EfficientNetB0, DenseNet121, and MobileNet architectures achieved the best performance in terms of accuracy, precision, and recall, while the ResNet architectures performed poorly. Based on these findings, it can be concluded that the EfficientNet, DenseNet121, and MobileNet architectures are the most effective CNN architectures for wear particle classification, while the ResNet architectures are not suitable for transfer learning with fine tuning approach for wear particle classification. Whereas the fine-tuning methodology used in this study is not suitable for ResNet architectures for wear particle classification.

After evaluating various factors, including accuracy, precision, recall, and computational efficiency, it was found that the EfficientNetB0 architecture demonstrated the best overall performance for wear particle classification. This architecture achieved a validation accuracy of 99.42% and an F-score of 0.99 on a small test dataset and was also computationally efficient with least number of FLOPs among the CNN architectures studied. The results suggest that these architectures could be used for efficient and accurate wear particle classification. Overall, this study has contributed to the development of more effective and efficient methods for wear particle classification, which could have significant implications for various fields such as predictive maintenance and online machine condition monitoring.

References

[1] Biewald, L. (2020). *Experiment Tracking with Weights and Biases*. https://www.wandb.com/

[2] Cao, W., Chen, W., Dong, G., Wu, J., & Xie, Y. (2014). Wear condition monitoring and working pattern recognition of piston rings and cylinder liners using on-line visual ferrograph. *Tribology Transactions*, *57*(4), 690–699.

[3] Cao, W., Dong, G., Xie, Y.-B., & Peng, Z. (2018). Prediction of wear trend of engines via on-line wear debris monitoring. *Tribology International*, *120*, 510–519.

[4] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1251–1258.

[5] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

[6] Fan, B., Feng, S., Che, Y., Mao, J., & Xie, Y. (2018). An oil monitoring method of wear evaluation for engine hot tests. *The International Journal of Advanced Manufacturing Technology*, *94*(9), 3199–3207.

[7] Fan, B., Li, B., Feng, S., Mao, J., & Xie, Y.-B. (2017). Modeling and experimental investigations on the relationship between wear debris concentration and wear rate in lubrication systems. *Tribology International*, *109*, 114–123.

[8] Fan, H., Gao, S., Zhang, X., Cao, X., Ma, H., & Liu, Q. (2020). Intelligent recognition of ferrographic images combining optimal CNN with transfer learning introducing virtual images. *IEEE Access*, *8*, 137074–137093.

[9] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

[10] Henneberg, M., Eriksen, R. L., Jørgensen, B., & Fich, J. (2015). A quasi-stationary approach to particle concentration and distribution in gear oil for wear mode estimation. *Wear*, *324*, 140–146.

[11] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv Preprint ArXiv:1704.04861*.

[12] Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708.

[13] Jia, F., Yu, F., Song, L., Zhang, S., & Sun, H. (2020). Intelligent classification of wear particles based on deep convolutional neural network. *Journal of Physics: Conference Series*, *1519*(1), 012012.

[14] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980*.

[15] Kirk, T. B., Panzera, D., Anamalay, R. V. and, & Xu, Z. L. (1995). Computer image analysis of wear debris for machine condition monitoring and fault diagnosis. *Wear*, *181*, 717–722.

[16] Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *ArXiv Preprint ArXiv:1312.4400*.

[17] Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11976–11986.

[18] Masters, D., & Luschi, C. (2018). Revisiting small batch training for deep neural networks. *ArXiv Preprint ArXiv:1804.07612*.

[19] More, P. P., & Jaybhaye, M. D. (2022). Wear particles recognition through teachable machine. *Industrial Lubrication and Tribology*.

[20] Myshkin, N. K., Kwon, O. K., Grigoriev, A. Y., Ahn, H.-S., & Kong, H. (1997). Classification of wear debris using a neural network. *Wear*, *203*, 658–662.

[21] Peng, P., & Wang, J. (2019a). FECNN: A promising model for wear particle recognition. *Wear*, *432*, 202968.

[22] Peng, P., & Wang, J. (2019b). Wear particle classification considering particle overlapping. *Wear*, *422*, 119–127.

[23] Peng, P., & Wang, J. (2020). Ferrograph image classification. *ArXiv Preprint ArXiv:2010.06777*.

[24] Peng, Y., Cai, J., Wu, T., Cao, G., Kwok, N., & Peng, Z. (2020). WP-DRnet: A novel wear particle detection and recognition network for automatic ferrograph image analysis. *Tribology International*, *151*, 106379.

[25] Peng, Y., Cai, J., Wu, T., Cao, G., Kwok, N., Zhou, S., & Peng, Z. (2019). A hybrid convolutional neural network for intelligent wear particle classification. *Tribology International*, *138*, 166–173.

[26] Peng, Z., & Kirk, T. B. (1999). Wear particle classification in a fuzzy grey system. *Wear*, *225*, 1238–1247.

[27] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510–4520.

[28] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *ArXiv Preprint ArXiv:1409.1556*.

[29] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929–1958.

[30] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826.

[31] Tan, M., & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning*, 6105–6114.

[32] Thomas, A. D. H., Davies, T., & Luxmoore, A. R. (1991). Computer image analysis for identification of wear particles. *Wear*, *142*(2), 213–226.

[33] Wang, S., Wu, T. H., Shao, T., & Peng, Z. X. (2019). Integrated model of BP neural network and CNN algorithm for automatic wear debris classification. *Wear*, *426*, 1761–1770.

[34] Wang, S., Wu, T., & Wang, K. (2021). Automated 3D ferrograph image analysis for similar particle identification with the knowledge-embedded double-CNN model. *Wear*, *476*, 203696.

[35] Wu, H., Kwok, N. M., Liu, S., Li, R., Wu, T., & Peng, Z. (2019). Restoration of defocused ferrograph images using a large kernel convolutional neural network. *Wear*, *426*, 1740–1747.

[36] Wu, T., Peng, Y., Wu, H., Zhang, X., & Wang, J. (2014). Full-life dynamic identification of wear state based on on-line wear debris image features. *Mechanical Systems and Signal Processing*, *42*(1–2), 404–414.

[37] Xie, F., & Wei, H. (2022). Research on controllable deep learning of multi-channel image coding technology in Ferrographic Image fault classification. *Tribology International*, *173*, 107656.