Quantized Deep Transfer Learning - Gearbox Fault Diagnosis on Edge Devices

Hetarth Chopra[@ a], Subrata Mukherjee[@ a,b,*], Jino Rohit[@ c], Vikash Kumar[b], Nishtha Hooda[d], Prashant Singh Rana[a], Somnath Sarangi[b]

[a]*Department of Mechanical Engineering, Thapar Institute of Engineering and Technology Patiala, Punjab 147004, India*
[b]*Department of Mechanical Engineering, Indian Institute of Technology Patna, Bihar 801103, India*
[c]*Department of Computer Engineering, Kumaraguru College of Technology, Coimbatore, Tamil Nadu 641049, India*
[d]*Department of Computing, Indian Institute of Information Technology Una, Himachal Pradesh 177209, India*

Abstract

This study has designed and implemented a deep transfer learning (DTL) model-based framework that takes an input time series of gearbox vibration patterns, which are accelerometer readings. It classifies the gear's damage type from a predefined catalog. Industrial gearboxes are often operated even after damage because damage detection is formidable. It causes a lot of wear and tear, which leads to more repair costs. With this proposed DTL model-based framework, at an early stage, gearbox damage can be detected so that gears can be replaced immediately with less repair cost. The proposed methodology involves training a convolutional neural network (CNN) model using a transfer learning technique on a predefined dataset of eight types of gearbox conditions. Then, using quantization, the size of the CNN model is reduced, leading to easy inference on edge and embedded devices. An accuracy of 99.49 % using transfer learning of the VGG16 model is achieved, pre-trained on the Imagenet dataset. Other models and architectures were also tested, but VGG16 emerged as the winner. The methodology also addresses the problem of deployment on edge/embedded devices, as in most cases, accurate models are too heavy to be used in the industry due to memory and computation power constraints in embedded devices. This is done with the help of quantization, enabling the proposed model to be deployed on devices like the Raspberry Pi, leading to inference on the go without the need for the internet and cloud computing. Consequently, the current methodology achieved a 4x reduction in model size with the help of INT8 Quantization.

Keywords: Fault Diagnosis, Spectrogram, Neural Network, Transfer Learning, Quantization, Edge Devices.

1. Introduction

Gearboxes are one of the most important key elements in the transmission industry due to their flexibility and compatibility with smooth power transmission [1, 2, 3, 4, 5, 6]. The gearbox failed unexpectedly due to high loading and harsh environmental conditions [1, 2, 3, 4, 5, 6]. Failures of gearboxes can result in significant financial losses, loss of human life, disruption of operations, and so on [1, 2, 3, 4, 5, 6]. To overcome the aforementioned limitations, it is critical to diagnose gearbox faults at an early stage.

This problem is usually solved with the help of statistical analysis, signal processing, and/or deep learning methods [7]. In the former two methodologies, techniques such as using spectrograms, wavelet filters, scaleograms, PCA, and time series algorithms such ARIMA, SARIMAX, etc. [8] [9] [10]. However, due to noise present in most systems due to unrelated components, rigorous signal pre-processing is needed to filter these signals out, which requires intensive domain knowledge [11]. Hence, due to this constraint not being present in DTL-based systems, they have gained

[@]These authors contributed equally to this work.
[*]Corresponding author
Email addresses: `choprahetarth@gmail.com` (Hetarth Chopra), `subrata.re.me@gmail.com` (Subrata Mukherjee), `find.jinorohit@gmail.com` (Jino Rohit), `vikashkumar7r@gmail.com` (Vikash Kumar), `27nishtha@gmail.com` (Nishtha Hooda), `psrana@gmail.com` (Prashant Singh Rana), `somsara@iitp.ac.in` (Somnath Sarangi)

popularity in solving problems related to this domain. According to researchers, DTL has made significant contributions to fault diagnosis in recent years [12, 13, 14, 15, 16, 17]. DTL has achieved outstanding fault diagnosis results with significant advancements in deep learning frameworks [12, 13, 14, 15, 16, 17]. Many researchers applied methods to fault diagnostics practices and achieved positive results, providing a novel approach to solving the fault diagnosis problem [12, 13, 14, 15, 16, 17]. Also, some research suggests that DTL can be very effective for IoT-based fault diagnosis [18]. In this context, edge intelligence (EI) [19, 20, 21] has recently made a significant contribution to the various IoT-based applications due to its numerous advantages for studying large industrial datasets [22, 23, 24, 25]. Edge intelligence has also shown significant advances in vibration edge computing for developing IoT-based machinery fault diagnosis models [26, 27, 28, 29, 30, 31]. By taking the advantages of deep learning (DL), some research suggests that the deployment of DL with EI [32, 33, 34, 35, 36, 37, 38, 39] has recently been a critical development in a variety of IoT-based applications. Deep learning, in conjunction with edge intelligence, has also aided in the development of some promising frameworks for real-time fault diagnosis of rotary machinery [21, 28, 29, 28, 30, 31]. Despite the above, only a few authors have proposed using DTL with edge intelligence [21, 28, 29, 28, 30, 31] in various applications. Sufian et al. [40] proposed a home health monitoring edge computing method based on transfer learning. For example, a pre-trained convolutional neural network (CNN) model could fine-tune the model right on edge devices with a small amount of ground-labeled data, with minimal computational cost. This allows the model to learn continuously and discreetly since the labeled data does not leave the edge device, addressing security concerns. Further, the user can consent to data sharing with the cloud, where the inference can be cached for re-use. Daga et al. [41] proposed a system for collaborative learning in edge clouds that creates a model-sharing environment in which tailored models at each edge can adapt to changes quickly and are as robust and accurate as centralized models. It works better than usual systems that transfer data from different edge devices to a centralized server, requiring heavy data movement. It also can result in inaccuracies of model learning because of local data shifts and reduced data availability. The method addresses these problems by applying three techniques- drift detection (to detect data drift), logical neighbors' detection (to find similar nodes), and knowledge transfer (to combine knowledge from different nodes). Liu et al. [42] proposed EdgeVegfru, a novel edge computing system for classifying vegetable and fruit images. The model takes advantage of the work done on the Vegfru dataset by deploying it on an Android device with the help of quantization. The strength of their system is that the inference speed is comparable to machine learning (ML) models, which is an accomplished thing to achieve for a deep learning model. Sufian et al.[43] presented a detailed survey on deep transfer education to edge computing for COVID-19 pandemic mitigation.

1.1. Motivations

The previous fault diagnostics applications mainly focused on traditional deep learning with edge computing to develop various fault diagnosis frameworks. A lot of studies have been done, and a lot of models have been made in this particular area. However, huge models are useless if not deployed and used in the industry. For example, the works of Li et al. [30] and Qizhao et al. [31] focus primarily on training DL models specific to edge computing but do not include transfer learning, which leads to a large requirement of data as opposed to the proposed method. In addition, works similar to Zhao et al. [28] do not use techniques to reduce the size of neural networks running in inference on edge devices, such as quantization, distilling, and so on, resulting in huge models, which are a problem because edge devices for industries are embedded systems with low memory. Similarly, works like [29] Tallón-Ballesteros consider quantization to reduce model size on edge devices but do not consider standardized CNN architectures like Resnet50/VGG16 to perform transfer learning. Another noteworthy work that has been done in this regard [21], where Qian et al. have discussed a novel edge computing framework for gearbox fault diagnosis using deep learning. However, their work required the extraction of features manually using Fourier transforms. Even though it makes the model development more explainable, at the same time, manual feature selection requires heavy domain knowledge, thus limiting the repeatability of the research. The work employed no model size reduction technique, indicating scope for improvement in that area. In conclusion, transfer learning is important not only because it reduces the amount of data required to train the model but also because it reduces the time and computational power required to train the model and, if scaled large enough, can even reduce the carbon footprint of performing the same job. These factors suggest that a quantized deep learning technique that is deployment-friendly on edge devices that can be used in the industry is required.

1.2. Contributions

Therefore, this paper presents a DTL model-based framework that uses an input spectrogram generated from gearbox vibration patterns and classifies the gear's damage type from a predefined catalog. The model is further quantized using post-training INT8 Quantization, reducing its size and making it easier to deploy on edge/embedded devices in the industry. With this proposed DTL model-based framework, gearbox damage can be detected early, and gears can be replaced with less repair cost. Firstly, this paper used to convert the vibration signals, which are in the form of time series data, to a 2D vibration spectrogram. Then, the spectrograms are provided as the data to fine-tune pre-trained CNN models using deep transfer learning. Finally, the model is made compatible for deployment on embedded/edge devices, leading to inference on the fly, enabling cost reduction of deploying the model on cloud services. The main novel contributions of the paper are:

- Designed a DTL model-based framework that trains CNN using Transfer Learning on a predefined gearbox dataset.
- A baseline model is developed for fine-tuning the ResNet-50 architecture on the produced spectrogram images using deep transfer learning.
- Multiple CNN architectures are fine-tuned with the same framework, and a comparative analysis is sought based on accuracy. The VGG-16 model is selected since it performs the best among the others selected.
- The best-performing model is optimized for inference on Edge and Embedded devices using INT-8 Quantization, increasing the inference speed and decreasing the cost of deployment.

The research is organized as follows: Section 2 discusses CNN's theoretical foundations. Section 3 discusses the principle of transfer learning. Section 4 discusses quantization. Section 5 demonstrates the experimental analysis and proposed methodology. Section 6 presents the findings and discussions, Section 7 presents a detailed comparative analysis of the proposed method, and Section 8 concludes the study.

2. Convolution Neural Networks

In recent years, Deep learning has come to attention, and it has made outstanding achievements and progress, especially in computer vision and natural language processing. In computer vision, deep learning-based techniques have assisted image processing-related tasks manifold. CNNs are specifically designed deep-neural networks to handle image-related tasks, as they have significant advantages over ANNs (Artificial Neural Networks), namely parameter sharing, pooling layers, etc. It helps them to outperform ANNs (in both accuracy and computation power) by taking advantage of semantic information, dimensionality reduction, etc. [44]. Stacking together convolutional layers, pooling layers, and fully connected layers in its architecture, CNNs learn how to extract and recognize patterns relating the input image with the targets.

2.1. Convolution Layer

The initial layers within any CNN are usually called Convolution Layers, also called "Feature Extractors," as they help produce feature maps essential for learning features. The feature maps are produced by applying an M x N convolution operation (denoted as *) between the source image and a convolutional filter. Generally, repeated convolutional operations occur, increasing the number of filters while decreasing the size of the image, leading to a large number of features being recognized. The mathematical operations required to produce the feature maps are:

$$a_j^{l+1} = \sigma(\sum_i x_i^l * w_{ij}^l + b_j^{l+1}) \tag{1}$$

where a_j^{l+1} represents $(l+1)$ layer's j^{th} channel of the feature map, produced by convolving x_i^l, which is the i^{th} channel of the feature map of the l^{th} layer with w_{ij}^l, which is the filter of the j^{th} channel using the convolution operation (*). It is finally added with a b_j^{l+1} bias vector, and a nonlinearity function called an activation function denoted by σ, is applied. These resulting feature maps dynamically change their values during the training of the CNN, as w and b are learnable parameters affected by backpropagation.

2.2. *Activation functions*

For the correct functioning of neural networks, activation functions such as ReLU [45], tanh [46], softmax [**?**], etc., are introduced to randomly scale the intensity of individual neurons during the neural networks (NN) training. NN optimizes a function concerning its output while learning. The optimization usually requires non-linear transformations achieved by activation functions. It enables the neural network to learn complex features not captured by linear transformations. The most commonly used activation function, ReLU, can be mathematically expressed by the following function:

$$a = max(0, z) \tag{2}$$

where z is the input from the convolution operation, in simple terms, it helps the CNN to output a linear value when $z \geq 0$ while it reduces the output to 0 when $z < 0$.

2.3. *Pooling layer*

A pooling layer is an essential component of a CNN since it helps to sub-sample an image, decreasing the overall feature map's size. It reduces the overall computation cost while retaining the most important features for the neural network to focus on, helping the CNN achieve spatial invariance. Max Pooling is the most common type of pooling, where a window slides over the input image, and the maximum pixel value is retained to make the next feature map. Mathematically, it can be expressed as:

$$f_{max}(a) = max_i(a_i) \tag{3}$$

where a_i represents a local pooling region's activations (feature map values) usually defined by a pxq vector. In this case, the vector slides over the activation maps as a window function, picking the maximum pixel value in its subregion and forming the next layer on which convolution operation can occur.

2.4. *Loss Function*

The objective function that CNN attempts to optimize is the loss function. The function is optimized with the back-propagation of the error between the actual and predicted labels by tweaking the CNN's parameters according to the error. For classification problems, the most commonly used loss function is known as the cross-entropy loss, which can be mathematically expressed by:

$$L_k = -\sum_{i=1}^{n} y_i log(\hat{y}_i) \tag{4}$$

where L_k is the loss of the k^{th} iteration, y_i and $\hat{y}_i$ are the true and predicted labels of the i^{th} sample respectively, and n is the total size of the sample space.

Other concepts essential for the working of Neural Networks, such as forward and backward propagation, gradient descent, momentum, etc., also form the basis of Convolutional Neural Networks as well [44].

3. Transfer Learning

Most neural networks perform well when the number of layers and/or the data increases. The dependence on data is very well established in works such as [47], where it has been theorized that there is an almost linear relationship between the scale of the model and the required amount of data. Moreover, having a large and annotated dataset of evenly distributed images for certain tasks, such as cancer detection and fault detection in mechanical and electrical systems, is impossible. Hence, such tasks need to look out for transfer learning, enabling the sharing of training parameters within the CNN's layers, even when the domains are different [48]. Not only does this technique help achieve more accurate models with fewer data, but it also reduces the training time drastically, as the weight matrix is provided with a good estimate of parameters, as opposed to standard He/Xavier initialization techniques [49] [50]. In Figure 1. It is seen that the pre-trained backbone/architecture (the CNN layers except for the last one) is directly carried forward for transfer learning. The change occurs at the training head (the last layer of the CNN), where the number of output nodes is changed according to the number of targets the custom dataset has. This process makes the task of transfer learning not only easy but also flexible at the same time since the number of targets does not limit knowledge transfer.

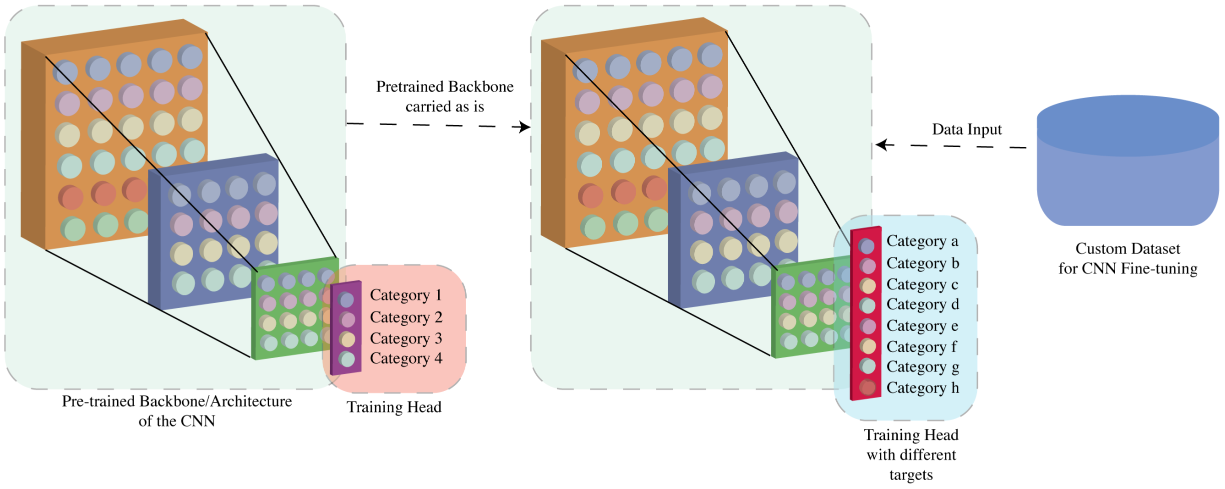

Pretrained Backbone carried as is
Category 1
Category 2
Category 3
Category 4
Pre-trained Backbone/Architecture of the CNN
Training Head
Data Input
Category a
Category b
Category c
Category d
Category e
Category f
Category g
Category h
Custom Dataset for CNN Fine-tuning
Training Head with different targets

Figure 1: Mechanism of Transfer Learning in CNN's

4. Quantization

After the CNN model is successfully trained on a dataset with the help of transfer learning, it is usually deployed as software for prediction. However, as the depth of the neural network increases, so do the number of parameters and the intrinsic calculations involved while producing an inference. This increases the overall size of the model and the inference time due to the computation cost, which is unsuitable when the model has to be deployed on an edge device such as a smartwatch or a microcontroller. Decreasing the number of layers is an option to decrease the model's size and the inference time; however, it can be drastic for the model's accuracy. Therefore, quantizing the model's weights post-training is important, a popular way of achieving light and fast models without compromising the quality of predictions. The quantization process usually involves approximating a neural network's weights to a certain discrete level from the original floating-point representation as studied in many texts such as [51] [52]. In the discussed case, the deployment of the trained CNN would be done on an edge device, such as a microcontroller, that would monitor the signals in real-time; the model must be quantized. For the proof of concept, the paper focuses on post-training quantization, also known as static quantization, as discussed in [53]. Quantization is most useful when the ML model has to be deployed on an edge device since it decreases the total number of computations and the amount of computation required to generate the inference, removing the dependency on the internet and cloud services.

5. Experimental Analysis and Proposed Method

To evaluate the proposed method, 2009 PHM gearbox fault data [1, 2, 3] were used. The challenge data is typical of a generic industrial gearbox with three shafts, four gears, and six bearings. Two kinds of gears were tested: spur gears and helical gears. The data set comprises two channels of accelerometer signals and one channel of tachometer signals collected by the corresponding sensors. Only the first channel of the vibration signals from the spur gearbox's labeled data set was used to train and test the proposed method in this study. The experimental data set contains eight different health conditions: level 1 (good gear), level 2 (gear chipped and eccentric), level 3 (gear eccentric), level 4 (gear eccentric, broken and bearing ball fault), level 5 (gear broken, bearing inner, ball, outer fault and shaft imbalance), level 6 (gear broken, bearing inner fault and shaft keyway sheared), level 7 (bearing inner fault, shaft keyway sheared), level 8 (bearing ball and outer fault). Signals were collected for each health condition at 66.67 kHz sampling frequency and 4 s acquisition time. The schematic and the overview of the gearbox used for data collection and a detailed description of the data set and pattern labels can be found in [1, 2, 3]. An example of vibration signals from all working conditions is shown in Figure 2 and Figure 3. Also, the entire development process is divided into a three-stage process covered in the diagram in Figure 4.

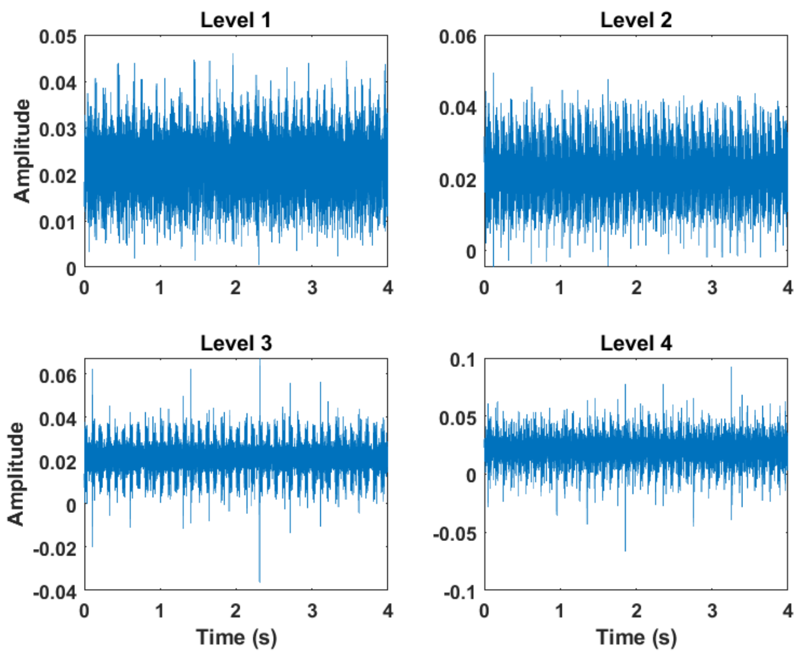


Figure 2: Vibration Signal of Level 1 to Level 4

5.1. *Data Preprocessing*

Before moving on to the machine learning model, the dataset was preprocessed, transforming the one-dimensional time series data into a 2D spectrogram for a CNN to process. The entire process can be described in the following steps -

(*a*) Data Division - Each Spur category consists of 20 files of 266656 data points of acceleration values; however, the total dataset was primarily divided into a window of 8333 samples to produce more image samples. Two prime factors of 266656, 13, and 641 were multiplied to generate a window size that helps utilize every data point without overlap. The other prime factors were not considered since the window size was close to 10,000, which was discussed in [54], producing an adequate dataset size. The division results in a combined dataset of shape (5114, 2), out of which (512, 2) samples were isolated solely for testing.

(*b*) Spectrogram Generation - The divided signals were converted into their respective spectrograms with the help of the matplotlib library in Python [55] to visualize the signal in both the time and frequency domains. Steps A and B are also visually represented in Figure 5.

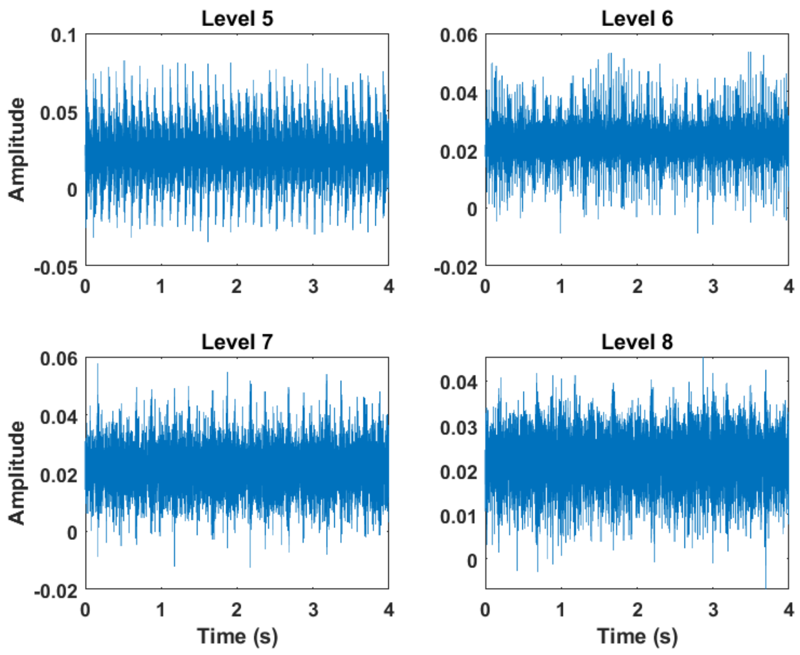


Figure 3: Vibration Signal of Level 5 to Level 8

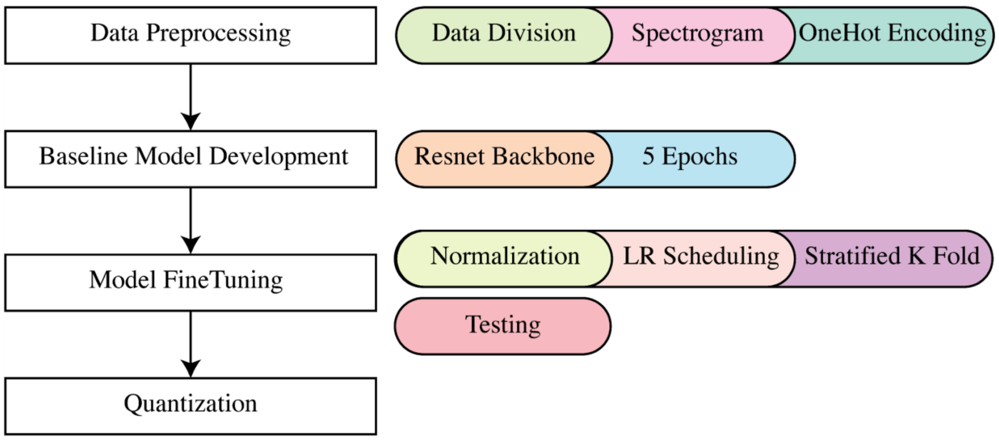


Figure 4: Development Methodology of the Proposed Model. Each module is divided into different steps, described individually in their sections.

(c) One Hot Encoding - Since the problem statement deals with multiclass classification, the CNN needs to be fed with a spare representation of the labels. Hence, the labels were one hot encoded before the model training, thus enabling the model to learn without having a label bias.

5.2. *Baseline Model Development*

The initial development step of this research work was to build a baseline model to monitor its performance on the dataset. The following steps elucidate the process clearly -

(a) Resnet Backbone - A pre-trained backbone network had to be defined for a baseline model development. In this particular case, Resnet50 [56] was the most suitable choice because of its extensive use in image processing

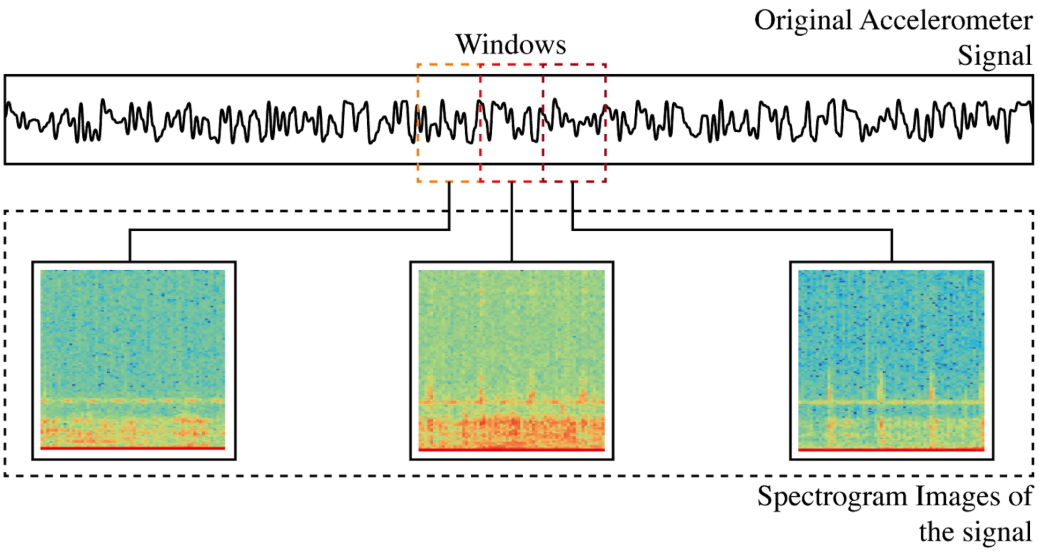
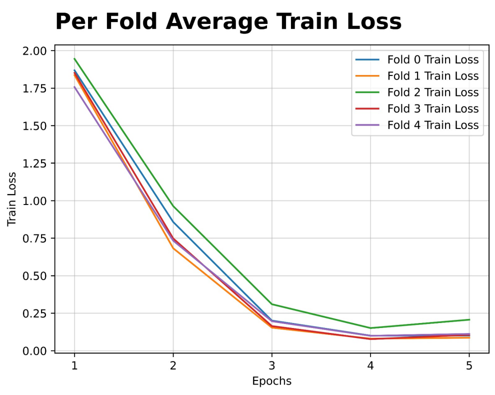

Original Accelerometer Signal
Windows
Spectrogram Images of the signal

Figure 5: Process of division and spectrogram generation of the accelerometer signal.

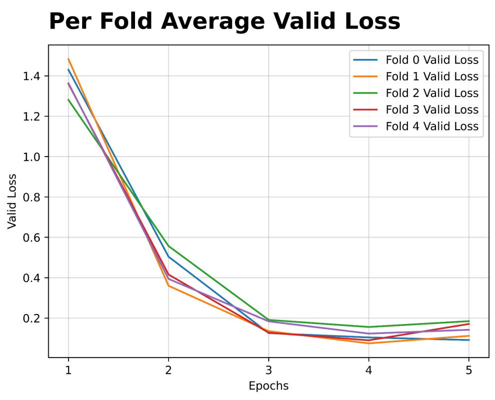
research. The popular Timm library [57] was used to acquire the pre-trained weights of the model.

(b) Training - Since this is a classification task, the cross-entropy loss was used, as mentioned in (4) as the objective function and ran for an arbitrary number of epochs. It was noticed that given a batch size of 10, the model could converge easily within 5 epochs 6, and to reduce the possibility of over-fitting, the number of epochs was fixed as 5.

Images were passed to the CNN in a randomized batched manner to obtain raw logits. In the case of classification, these logits are then converted into probabilities using the softmax function. The softmax function provides each class's probability to be the predicted class; hence, the class with the maximum probability is selected. The model achieved an accuracy score of 65% on the test set. This result indicated the need to produce a more robust model with the help of fine-tuning parameters and the dataset.



(a)



(b)

Figure 6: Train and Validation Loss for 5 epochs on all folds

5.3. Model Fine Tuning

To fine-tune the model, the following steps were followed -

(a) Normalization - Some pre-processing steps were applied to the images like image resizing and normalizing them to the ImageNet dataset mean and standard deviation. Normalizing was done concerning the mean and standard deviation of the ImageNet Dataset since pre-trained weights of models trained on the ImageNet dataset were used, and this strategy has been proved to work well [58]. After the preprocessing step, the images and the labels were shuffled randomly and passed as batches into the model to obtain the raw logits.

(b) LR Scheduling - A Learning Rate Scheduler was used to change the model's learning rate over epochs. The advantage of oscillating the learning rate within the upper and lower bound was that whenever the network got stuck at local minima(s), the current learning rate decreased to help break it out and descend into lower loss regions. CosineAnnealingLR seemed to be the best-performing scheduler in this dataset. In this function, the learning rate of each of the parameters was set to an initial learning rate that rapidly decreased to a minimum value before increasing again after running for T epochs.

$$\eta_t = \eta^i_{min} + \frac{1}{2}(\eta^i_{max} - \eta^i_{min})(1 + cos(\frac{T_{cur}}{T_i}\pi)) \tag{5}$$

As per equation, 5 learning rate is set by η^i_{min} and η^i_{max} are the upper and lower bound for learning rate and T_{cur} is the number of epochs performed since the last restart [59].

(c) Stratified K-Fold Cross Validation - A Stratified K-Fold CV strategy was used even though the current scenario had no class imbalance problem. The value of K was chosen to be 5, ensuring that sufficient models were trained on the dataset, boosting the accuracy score and increasing the model's reliability. After running the model for 5 epochs, an accuracy score of 87% on the test set was achieved.

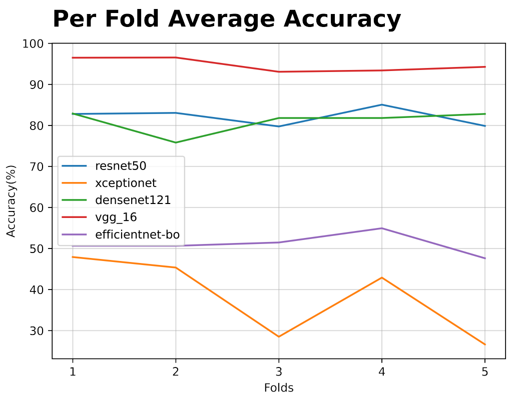
(d) Testing- Despite already having a good enough performance, the entire process was repeated using other popular architectures as backbones to compare the architectures - Efficientnet-b0, xceptionNet, densenet121, and the vgg16. The parameters are fixed to provide an unbiased replication of the process and are shown in the following Table 1. The metrics are compiled and presented in the results section.

Table 1: Fixed Hyperparameters for all model architectures

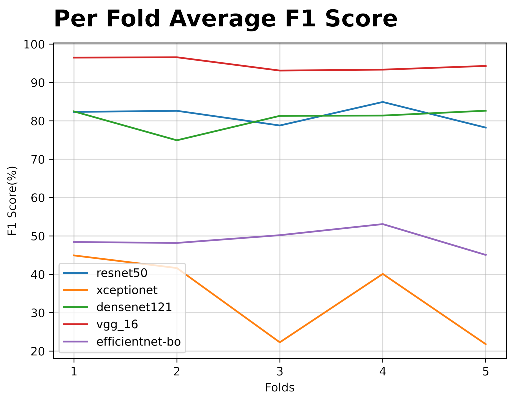
Serial Number	Parameter Description	Value
1	Optimizer	Stochastic Gradient Descent
2	Momentum	0.9
3	Learning Rate	0.0004
4	Batch Size	10
5	Scheduler	CosineAnnealingLR
6	Decay Steps	1000
7	Number of Folds	5
8	Number of Epochs	5

6. Results and Discussion

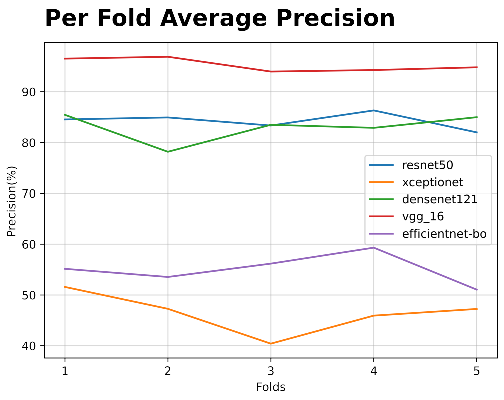
During the development of the model, the graphs shown in Figure 7 describe the average metrics per fold for all the models scored on the validation dataset considered for discussion. An interesting insight that could be inferred from Figure 7 was that in all the metrics, Xception and Efficientnet-B0 architecture performed worse than the other architectures in all the folds. The best-performing model remained VGG16 throughout all the folds, which is interesting since it predated all the other models. Alongside this, one could see that the model was not overfitting on any of the architectures because of the consistent value of the metrics over different folds.



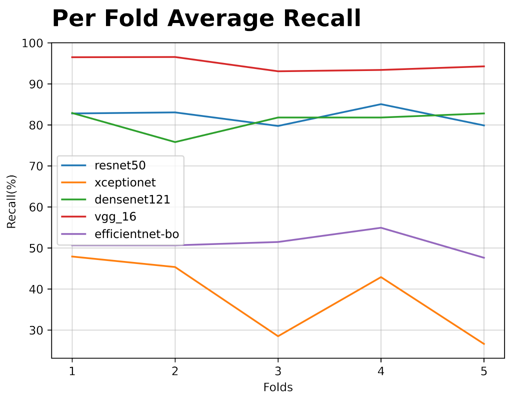
(a) a



(b) b



(c) c



(d) d

Figure 7: (a) Average Accuracy of all models per fold (b) Average F1 Score of all models per fold (c) Average Precision of all models per fold (d) Average Recall of all models per fold

After the parameters were frozen, as shown in Table 1, multiple model architectures, as discussed above, were experimented with, and the inferred results on the test dataset are compiled in Table 2. An interesting insight was that VGG16 not only had the best accuracy but also the best precision, recall, and F1 Score of all the other models, but its disadvantage was that it was the heaviest model with a little over 138 Million parameters. At the same time, the lightest model and the least number of parameters, i.e., EfficientNet-B0, failed to perform up to the level of VGG16, showcasing an interesting insight into the relationship of the number of parameters with the predictive performance.

Table 2: Model metric comparison between different architectures. The best parameters are highlighted using a bold font.

Architectures	Accuracy	Precision	Recall	F1 score	Num Parameters	Model Size
Resnet50 [56]	87.69	90.24	87.69	87.54	25.6 Mil.	180 Mb
xception [60]	45.70	55.32	45.70	40.63	22.9 Mil.	159.3 Mb
desnenet121 [61]	88.08	89.82	88.08	87.96	8.1 Mil.	53.8 Mb
vgg16 [62]	**99.41**	**99.02**	**99.02**	**99.02**	138 Mil.	112.3 Mb
efficient net-bo [63]	57.22	64.35	57.22	55.86	**5.3 Mil.**	**31 Mb**

6.1. Quantization and Deployment on Edge Device

Since the VGG16 proved to be the better model than the others considered, it was quantized to reduce the space taken by the model on an Embedded/Edge Device. With the help of TFLite by Tensorflow [64], a framework for running and training ML Models on the edge and mobile devices, the fine-tuned VGG16 model was converted into a .tflite model. Following this, with the help of the tflite Optimize library, the model was quantized and converted from a full floating point precision to an INT8 precision, resulting in a smaller model with a faster inference. As per Table 3, the models' size can be seen concerning the Quantization Algorithm. An overall 8 times size reduction could be seen with the help of fully INT8 Quantization, making it easy to deploy on an edge device.

Table 3: VGG-16 Quantized Model Size Comparison

Type of Quantization	Size
No Quantization (only TFlite)	112.1Mb
Float16 Quantization	56.03Mb
Int16 Quantization	28.01Mb
Int8 Quantization	14.1 Mb

Once the model is deployed on the edge device, it will be used to identify the fault type of the gearbox. Accelerometer data will be streamed from the machine via a database for 12 milliseconds to produce a single spectrogram. The sample will be sent to the quantized DL model for inference as shown in the 8.

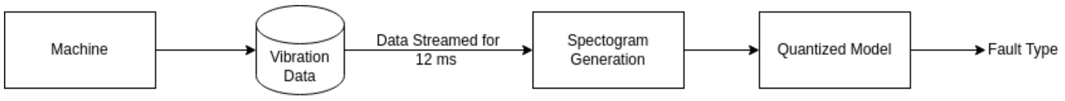


Figure 8: Illustration of Model Inference to identify and diagnose Gearbox Fault at deployment time.

7. Comparative Analysis

In Table 4, we have presented a comparative study of our model's architecture and unique features as compared to the prevalent works in the same domain. Most approaches have dealt with this problem by extracting data from an Accelerometer, including the proposed method. However, [21] shows an interesting approach of performing the same sort of experiment but with data extracted from a Hall Effect Sensor. The most popular datasets include the PHM 2009 dataset and the CWRU Open Gearbox, which have been heavily studied, showcasing that the proposed study can be replicated on similar datasets. It is worth noting that none of the works employ a DTL approach and quantization. Another interesting insight is that despite this, many models have been deemed deployable on edge devices, indicating an interesting research area where this problem can be expanded and worked on. Most works do not require domain knowledge and human intervention, but it is interesting to see how domain knowledge can be coupled with transfer learning, especially in [17]. Overall, the proposed method uses DTL and quantization to make the model trained on the PHM 2009 dataset deployable easily on edge devices. It does it in a way that no domain knowledge is required. This indicates that it not only parallels the current state-of-the-art methods but it rather outshines them.

8. Conclusion and Future Work

A quantized DTL framework is proposed and deployed on an edge device in this paper for gearbox fault diagnosis. Under eight different conditions, scalogram images were used as input to the DTL of gearbox vibration signals in this paper. A CNN model and a transfer learning technique are used to classify gearbox faults in this proposed technique. In addition, quantization and transfer learning were combined to reduce the size of the predefined CNN model, allowing for easy inference on edge and embedded devices. Using the following metrics, a 99.48 percent accuracy was achieved by transfer learning of the VGG16 model from the Imagenet Weights (F1 score, sensitivity, and recall). Aside from

Table 4: Comparison of the proposed model with state-of-the-art techniques

Research Work	Use of DTL	Model Quantized	Deployable on Edge Devices	Requires Domain Knowledge	Sensor Used	Datasets Used
[1]	✗	✗	✗	✗	Accelerometer	PHM 2009
[2]	✗	✗	✗	✗	Accelerometer	PHM 2009
[3]	✓	✗	✗	✗	Accelerometer	-
[17]	✓	✗	✗	✓	Accelerometer	Internal Dataset
[21]	✗	✗	✗	✓	Hall Effect Sensor	Internal Dataset
[28]	✗	✗	✗	✗	Accelerometer	CWRU Open Bearing
[29]	✗	✗	✓	✗	-	-
[30]	✗	✗	✓	✗	Accelerometer	CWRU Open Gearbox
[31]	✗	✗	✓	✗	Accelerometer	CWRU Open Gearbox
Proposed Method	✓	✓	✓	✗	Accelerometer	PHM 2009

that, using INT8 Quantization, model size was reduced by 4X, allowing us to deploy models on edge and embedded devices. Other models and architectures were also evaluated, but VGG16 emerged victorious. Many studies in this area have been conducted, and an IoT model has been developed. Large models, on the other hand, are useless unless they are deployed and used in the industry. As a result of quantization, the light model can be deployed on devices like the Raspberry Pi, allowing for on-the-fly inference without needing the internet or cloud computing. This will reduce the cost of gearbox repair in industries. In the future, we will focus on incomplete data robust learning based on gear dynamics using DTL, which can incorporate edge devices with gear dynamics data in terms of missing features and partially observed labels. This study introduces a new pattern by utilizing the edge computing technique, and the proposed framework has potential applications in time-sensitive fault diagnosis using dynamic models of rotating machines like high-speed machine tools, trains, and gas turbines.

Acknowledgments

Contributing authors thank all the supervisors for their kind supervision and direction.

References

[1] L. Jing, M. Zhao, P. Li, and X. Xu, "A convolutional neural network based feature learning and fault diagnosis method for the condition monitoring of gearbox," *Measurement*, vol. 111, pp. 1–10, 2017.

[2] C. Wu, P. Jiang, C. Ding, F. Feng, and T. Chen, "Intelligent fault diagnosis of rotating machinery based on one-dimensional convolutional neural network," *Computers in Industry*, vol. 108, pp. 53–61, 2019.

[3] X. Li, Y. Hu, M. Li, and J. Zheng, "Fault diagnostics between different type of components: A transfer learning approach," *Applied Soft Computing*, vol. 86, p. 105950, 2020.

[4] C. Li, R.-V. Sanchez, G. Zurita, M. Cerrada, D. Cabrera, and R. E. Vásquez, "Multimodal deep support vector classification with homologous features and its application to gearbox fault diagnosis," *Neurocomputing*, vol. 168, pp. 119–127, 2015.

[5] V. Kumar, S. Mukherjee, A. K. Verma, and S. Sarangi, "An ai-based non-parametric filter approach for gearbox fault diagnosis," *IEEE Transactions on Instrumentation and Measurement*, 2022.

[6] V. Kumar, A. Verma, and S. Sarangi, "Fault diagnosis of single-stage bevel gearbox by energy operator and j48 algorithm," in *Advances in Condition Monitoring and Structural Health Monitoring*, pp. 231–239, Springer, 2021.

[7] J. Li, D. Zhou, X. Si, M. Chen, and C. Xu, "Review of incipient fault diagnosis methods," *Control Theory & Applications*, vol. 29, no. 12, pp. 1517–1529, 2012.

[8] R. Sharifi and R. Langari, “Nonlinear sensor fault diagnosis using mixture of probabilistic pca models,” *Mechanical Systems and Signal Processing*, vol. 85, pp. 638–650, 2017.

[9] S. Krishnannair, C. Aldrich, and G. Jemwa, “Detecting faults in process systems with singular spectrum analysis,” *Chemical Engineering Research and Design*, vol. 113, pp. 151–168, 2016.

[10] Y. Yang, Y. Bai, C. Li, and Y.-N. Yang, “Application research of arima model in wind turbine gearbox fault trend prediction,” in *2018 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*, pp. 520–526, IEEE, 2018.

[11] T. Wang, Q. Han, F. Chu, and Z. Feng, “Vibration based condition monitoring and fault diagnosis of wind turbine planetary gearbox: A review,” *Mechanical Systems and Signal Processing*, vol. 126, pp. 662–685, 2019.

[12] C. Qian, J. Zhu, Y. Shen, Q. Jiang, and Q. Zhang, “Deep transfer learning in mechanical intelligent fault diagnosis: application and challenge,” *Neural Processing Letters*, pp. 1–23, 2022.

[13] C. Li, S. Zhang, Y. Qin, and E. Estupinan, “A systematic review of deep transfer learning for machinery fault diagnosis,” *Neurocomputing*, vol. 407, pp. 121–135, 2020.

[14] W. Li, R. Huang, J. Li, Y. Liao, Z. Chen, G. He, R. Yan, and K. Gryllias, “A perspective survey on deep transfer learning for fault diagnosis in industrial scenarios: Theories, applications and challenges,” *Mechanical Systems and Signal Processing*, vol. 167, p. 108487, 2022.

[15] X. Li, X. Jiang, Q. Wang, L. Yang, Z. Wang, C. Shen, and Z. Zhu, “Multi-perspective deep transfer learning model: A promising tool for bearing intelligent fault diagnosis under varying working conditions,” *Knowledge-Based Systems*, vol. 243, p. 108443, 2022.

[16] S. Shao, S. McAleer, R. Yan, and P. Baldi, “Highly accurate machine fault diagnosis using deep transfer learning,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2446–2455, 2018.

[17] Q. Qian, Y. Qin, Y. Wang, and F. Liu, “A new deep transfer learning network based on convolutional auto-encoder for mechanical fault diagnosis,” *Measurement*, vol. 178, p. 109352, 2021.

[18] X. Liu, W. Yu, F. Liang, D. Griffith, and N. Golmie, “Toward deep transfer learning in industrial internet of things,” *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12163–12175, 2021.

[19] D. Xu, T. Li, Y. Li, X. Su, S. Tarkoma, T. Jiang, J. Crowcroft, and P. Hui, “Edge intelligence: Architectures, challenges, and applications,” *arXiv preprint arXiv:2003.12172*, 2020.

[20] G. Plastiras, M. Terzi, C. Kyrkou, and T. Theocharidcs, “Edge intelligence: Challenges and opportunities of near-sensor machine learning applications,” in *2018 ieee 29th international conference on application-specific systems, architectures and processors (asap)*, pp. 1–7, IEEE, 2018.

[21] G. Qian, S. Lu, D. Pan, H. Tang, Y. Liu, and Q. Wang, “Edge computing: A promising framework for real-time fault diagnosis and dynamic control of rotating machines using multi-sensor data,” *IEEE Sensors Journal*, vol. 19, no. 11, pp. 4211–4220, 2019.

[22] M. Merenda, C. Porcaro, and D. Iero, “Edge machine learning for ai-enabled iot devices: A review,” *Sensors*, vol. 20, no. 9, p. 2533, 2020.

[23] T. Hafeez, L. Xu, and G. Mcardle, “Edge intelligence for data handling and predictive maintenance in iiot,” *IEEE Access*, vol. 9, pp. 49355–49371, 2021.

[24] P. Bellavista, R. Della Penna, L. Foschini, and D. Scotece, “Machine learning for predictive diagnostics at the edge: An iiot practical example,” in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2020.

[25] P. Patel, M. I. Ali, and A. Sheth, “On using the intelligent edge for iot analytics,” *IEEE Intelligent Systems*, vol. 32, no. 5, pp. 64–69, 2017.

[26] A. Verma, A. Goyal, S. Kumara, and T. Kurfess, “Edge-cloud computing performance benchmarking for iot based machinery vibration monitoring,” *Manufacturing Letters*, vol. 27, pp. 39–41, 2021.

[27] A. L. Michala, I. Vourganas, and A. Coraddu, “Vibration edge computing in maritime iot,” *ACM Transactions on Internet of Things*, vol. 3, no. 1, pp. 1–18, 2021.

[28] X. Zhao, K. Lv, Z. Zhang, Y. Zhang, and Y. Wang, “A multi-fault diagnosis method of gear-box running on edge equipment,” *Journal of Cloud Computing*, vol. 9, no. 1, pp. 1–14, 2020.

[29] A. Tallón-Ballesteros, “Edge analytics for bearing fault diagnosis based on convolution neural network,” *Fuzzy Systems and Data Mining VII: Proceedings of FSDM 2021*, vol. 340, p. 94, 2021.

[30] H. Li, G. Hu, J. Li, and M. Zhou, “Intelligent fault diagnosis for large-scale rotating machines using binarized deep neural networks and random forests,” *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 2, pp. 1109–1119, 2021.

[31] Q. Wang, G. Jin, Q. Li, K. Wang, Z. Yang, and H. Wang, “A fast and energy-saving neural network inference method for fault diagnosis of industrial equipment based on edge-end collaboration,” in *2021 IEEE 11th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pp. 67–72, IEEE, 2021.

[32] J. Chen and X. Ran, “Deep learning with edge computing: A review,” *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[33] H. Li, K. Ota, and M. Dong, “Learning iot in edge: Deep learning for the internet of things with edge computing,” *IEEE network*, vol. 32, no. 1, pp. 96–101, 2018.

[34] Y. Huang, X. Ma, X. Fan, J. Liu, and W. Gong, “When deep learning meets edge computing,” in *2017 IEEE 25th international conference on network protocols (ICNP)*, pp. 1–2, IEEE, 2017.

[35] S. P. Baller, A. Jindal, M. Chadha, and M. Gerndt, “Deepedgebench: Benchmarking deep neural networks on edge devices,” in *2021 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 20–30, IEEE, 2021.

[36] Z. Jiang, T. Chen, and M. Li, “Efficient deep learning inference on edge devices,” *ACM SysML*, 2018.

[37] F. Wang, M. Zhang, X. Wang, X. Ma, and J. Liu, “Deep learning for edge computing applications: A state-of-the-art survey,” *IEEE Access*, vol. 8, pp. 58322–58336, 2020.

[38] R. Hadidi, J. Cao, Y. Xie, B. Asgari, T. Krishna, and H. Kim, “Characterizing the deployment of deep neural networks on commercial edge devices,” in *2019 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 35–48, IEEE, 2019.

[39] A. Marchisio, M. A. Hanif, F. Khalid, G. Plastiras, C. Kyrkou, T. Theocharides, and M. Shafique, “Deep learning for edge computing: Current trends, cross-layer optimizations, and open research challenges,” in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 553–559, IEEE, 2019.

[40] A. Sufian, C. You, and M. Dong, “A deep transfer learning-based edge computing method for home health monitoring,” in *2021 55th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, IEEE, 2021.

[41] H. Daga, P. K. Nicholson, A. Gavrilovska, and D. Lugones, “Cartel: A system for collaborative transfer learning at the edge,” in *Proceedings of the ACM Symposium on Cloud Computing*, pp. 25–37, 2019.

[42] C. Liu, X. Wang, J. Ni, Y. Cao, and B. Liu, “An edge computing visual system for vegetable categorization,” in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 625–632, IEEE, 2019.

[43] A. Sufian, A. Ghosh, A. S. Sadiq, and F. Smarandache, “A survey on deep transfer learning to edge computing for mitigating the covid-19 pandemic,” *Journal of Systems Architecture*, vol. 108, p. 101830, 2020.

[44] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[45] A. F. Agarap, “Deep learning using rectified linear units (relu),” *arXiv preprint arXiv:1803.08375*, 2018.

[46] B. L. Kalman and S. C. Kwasny, “Why tanh: choosing a sigmoidal function,” in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, vol. 4, pp. 578–581, IEEE, 1992.

[47] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on deep transfer learning,” in *International conference on artificial neural networks*, pp. 270–279, Springer, 2018.

[48] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[49] S. K. Kumar, “On weight initialization in deep neural networks,” *CoRR*, vol. abs/1704.08863, 2017.

[50] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[51] Y. Ding, J. Liu, J. Xiong, and Y. Shi, “On the universal approximability and complexity bounds of quantized relu neural networks,” *arXiv preprint arXiv:1802.03646*, 2018.

[52] Z. Cai, X. He, J. Sun, and N. Vasconcelos, “Deep learning with low precision by half-wave gaussian quantization,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5918–5926, 2017.

[53] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, “A survey of quantization methods for efficient neural network inference,” *arXiv preprint arXiv:2103.13630*, 2021.

[54] P. Liang, C. Deng, J. Wu, Z. Yang, J. Zhu, and Z. Zhang, “Compound fault diagnosis of gearboxes via multi-label convolutional neural network and wavelet transform,” *Computers in Industry*, vol. 113, p. 103132, 2019.

[55] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in Science Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[56] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[57] R. Wightman, “Pytorch image models.” `https://github.com/rwightman/pytorch-image-models`, 2019.

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

[59] I. Loshchilov and F. Hutter, “Sgdr: Stochastic gradient descent with warm restarts,” *arXiv preprint arXiv:1608.03983*, 2016.

[60] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

[61] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[62] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[63] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.

[64] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015. Software available from tensorflow.org.