SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V11I9P104

Volume 11 Issue 9, 39-56, September 2024

Original Article

Improving Detection of Induction Motor Bearing Faults: A Study on Hyperparameter Tuning and ResNet-18 Layer Modification

Lydiah Aywa Sikinyi[1,5*], Christopher Maina Muriithi[1,2], Livingstone Ngoo[1,3], Duncan Shitubi[4]

[1]*Department of Electrical Engineering, Pan African University, Institute for Basic Sciences Technology and Innovation, Nairobi, Kenya.*
[2]*Department of Electrical and Electronics Engineering, Murang'a University of Technology, Murang'a, Kenya.*
[3]*Department of Electrical and Electronics Engineering, Multimedia University of Kenya, Nairobi, Kenya.*
[4]*Software Developer, DEFEND Organization, Toronto, Canada.*
[5]*Department of Electrical and Communication Engineering, Masinde Muliro University of Science and Technology, Kakamega, Kenya.*

[*]*Corresponding Author : Lydiah.sikinyi@gmail.com*

Received: 02 July 2024 Revised: 04 August 2024 Accepted: 02 September 2024 Published: 28 September 2024

Abstract *- Detecting incipient bearing faults of an induction motor is crucial for minimizing downtime, reducing maintenance costs, and ensuring safety. In this study, a novel approach to improve the detection accuracy of induction motor bearing faults using a combination of hyperparameter tuning and modifications to the Residual Network - 18 (ResNet-18) architecture is investigated. The effect of optimizing these aspects to enhance ResNet-18's ability to classify various fault types within a dataset of bearing vibration signals is explored. This research focuses on ResNet-18 architecture, which has demonstrated remarkable performance in various image classification tasks, to evaluate the impact of specific modifications to its layers in order to improve further its suitability for bearing fault detection. The appropriate balance between model complexity and interpretability is achieved by altering the depth, width, and skip connections within ResNet-18. Next, parameters such as batch sizes, learning rates, L2 regularization, number of epochs, and optimizer are investigated by systematically tuning these hyperparameters and applying layer modifications, a new Bayesian Optimized Squeeze and Excitation ResNet model, which has a higher training accuracy of 98.44%, a validation accuracy of 99.48%, testing accuracy of 99.48%, and lower computational cost, as compared to the ResNet-18 model, is achieved. The proposed BOSE-ResNet contributes to the development of a more effective and precise bearing fault diagnosis model while enhancing machinery reliability in industrial applications and providing valuable insights for practitioners and researchers in the field of condition-based maintenance.*

Keywords *- Bayesian optimization, Fault diagnosis, Hyperparameters, ResNet-18, Squeeze and excitation block.*

1. Introduction

The smooth operation of induction motors is vital in various industrial applications, and any malfunction can result in unexpected downtime and costly maintenance. The induction motor has numerous components that are prone to faults, such as stator windings, rotor bars, bearings, cooling systems, external connections, and terminals, to mention a few.

Bearings, which have the highest probability of failure at 41%, [1] play a pivotal role in ensuring the smooth and efficient operation of the induction motor. Accurate detection of incipient bearing faults is crucial for preventive maintenance and ensuring the reliability of induction motors. If undetected, bearing faults can lead to severe consequences such as increased energy consumption, reduced operational lifespan, and, in extreme cases, catastrophic failures. Recently, there has been a spiked interest in leveraging advanced machine-learning techniques for the early detection of faults in induction motor bearings.

Traditional fault diagnosis methods often rely on expert knowledge and manual feature engineering, which may not be scalable or adaptable to complex and evolving systems. Deep learning models, particularly Convolutional Neural Networks (CNNs), have emerged as powerful tools, allowing for timely interventions and preventing unexpected breakdowns due to their ability to learn complex patterns and representations from raw data [2]. However, very few studies have detailed the impact of tuning different hyperparameters and combining

hyperparameter optimization with ResNet-18 architecture modifications. Existing models also often suffer from high computational costs. This research aims to address these gaps by exploring how these techniques can work together to improve the detection of bearing faults in induction motors.

In this paper, the focus is on improving the detection of induction motor bearing faults by reducing the computational cost while enhancing the accuracy and efficiency of fault diagnosis through a comprehensive investigation into hyperparameter tuning and modifications to the Residual Network - 18 (ResNet-18) architecture.

ResNet-18 is a widely used Convolutional Neural Network (CNN) known for its effectiveness in image classification tasks. The architecture's distinctive feature of residual connections helps address the vanishing gradient problem, allowing for the training of deeper networks [3].

This study investigates the combined effect of hyperparameter tuning and ResNet-18 layer modification on improving the detection of induction motor bearing faults. It contributes to the development of a highly accurate and cost-efficient Bayesian Optimized Squeeze and Excitation Residual Network through modifications to the ResNet-18 architecture to tailor it specifically for the intricate features present in vibration signals associated with bearing faults.

Additionally, a comprehensive study on hyperparameter tuning, which is a crucial step in optimizing the performance of deep learning models, is done. By systematically adjusting parameters such as learning rates, batch sizes, and regularization terms, the accuracy and performance of the new model in identifying subtle variations indicative of bearing faults is enhanced.

By combining the power of transfer learning with advancements in deep learning architectures, the aim is to create a model capable of outperforming existing approaches by enhancing the model's ability to accurately classify various fault types within a dataset of bearing vibration signals, reducing the computational time, and paving the way for reliable and efficient operation of induction motors in diverse industrial settings.

The rest of the paper is organized as follows: Section 2 presents a detailed review of related works in the field of fault diagnosis using ResNets. An overview of the methodology comprising data pre-processing, ResNet-18 layer modification, hyperparameter tuning, and model evaluation is given in Section 3. The experimental findings and analysis are shown in Section 4, and Section 5 concludes by summarising the key findings and directing on possible areas for future research.

2. Related Work

In [4], researchers have looked at various approaches to improve early fault diagnosis in induction motors and prevent breakdowns. Vibration analysis is generally used for the detection of mechanical faults since it is easily measurable, highly accurate, and reliable [5]. A variety of signal processing techniques, including the Fourier Transform (FT), Empirical Mode Decomposition (EMD), and Wavelet Transform (WT), have been extensively utilized to extract features from the preprocessed vibration signal that are indicative of bearing health. Following the extraction of characteristics, machine learning or statistical methods are employed to classify the bearing condition.

These techniques analyze the vibration signals generated by the motor bearings to identify abnormalities associated with bearing faults. FT decomposes the signal into its frequency components but is unable to handle non-stationary signals. WT is more advanced and analyzes both frequency and time information, making it more suitable for non-stationary fault signatures [6]. EMD decomposes the signal into intrinsic mode functions that represent specific frequency components, which is beneficial in isolating fault-related frequencies from complex vibration signals containing multiple components and background noise.

A randomized Fisher Discriminant Analysis-based bearing fault diagnosis method was proposed in [7], which firstly required the extraction of time-domain features from the raw vibration signals. However, choosing an appropriate type of signal processing technique to be used with shallow machine learning algorithms such as Support Vector Machines (SVMs), decision trees, linear regression, and random forests, to mention a few, requires human expertise [2].

CNNs have gained popularity in the field of bearing fault diagnosis since they can automatically learn relevant features from raw data, eliminating the need for domain expertise in feature selection and allowing the model to identify the most discriminative features for the specific task at hand [8]. A CNN analyzes images and extracts important features. While increasing the number of stacked layers in a neural network can enrich features by enabling the model to identify more intricate connections within the data, this benefit is counterbalanced by the potential for vanishing/exploding gradients.

Normalized initialization and intermediate normalization layers have made significant progress in solving this issue by allowing networks with multiple layers to start converging for Stochastic Gradient Descent (SGD) with backpropagation. In [9], the problem of degradation was addressed by introducing a deep residual learning framework. Architectures like ResNet introduce shortcut connections that skip over some layers,

allowing the gradient to flow directly from earlier layers to later layers and bypassing the vanishing gradient issue in the skipped layers. The original ResNet-18 architecture, with its 18 layers, has been widely used in various applications due to its balance between performance and computational efficiency. Subsequent research has led to the development of deeper ResNet variants, such as ResNet-50 and ResNet-101, which have achieved state-of-the-art results in image classification. However, the application of these deeper architectures to bearing fault diagnosis is relatively limited due to their high computational cost.

The Domain Adaptation ResNet structure is selected in [10] to help mitigate training errors that typically arise in deeper networks, achieving a 94.22% accuracy rate. In comparison, a classification accuracy of 97.54% is achieved when the Principal Component Analysis is used to fuse features from multiple sensors and feed to a hybrid SVM-ResNet model [11]. These studies demonstrate the potential of ResNet for fault classification tasks, and the effectiveness of squeeze and excitation ResNet models is highlighted in [3] after a signal-to-image method using continuous wavelet transform outperformed the ResNet family achieving a high accuracy of 93% for rolling bearing fault diagnosis.

Various optimization techniques have been employed to find optimal hyperparameter settings. Bayesian Optimization has emerged as a popular choice due to its efficiency in exploring the search space. A Bayesian Optimization ResNet-18 was employed in [12] to diagnose motor bearing faults since after comparing it to transfer learning models such as ResNet-50, LeNet-5, VGG-16, and so on, it gave the highest classification accuracy. The ResNet-18's learning rate, momentum, and L2 regularization hyperparameters were optimized to achieve a training accuracy of 89.50%. Its diagnostic accuracy was 99.31% for faults without the addition of simulated industrial noise from the hydrogen station and 92% with the addition of noise. Jaber et al. proposed a ResNet-based deep learning multilayer fault detection model achieving an accuracy of 83.5% for the data transmission ratio, [13] while a fault diagnosis method based on the Markov transition field and residual networks was able to achieve an accuracy of greater than 98.52% in the identification of rolling bearings faults with various degrees of severity and locations [14]. However, compared to shallow neural network-based techniques, it needed a longer training time since it was trained from scratch.

From previous research, it is evident that deep learning has shown great potential in bearing fault diagnosis, but several challenges remain. It is also clear that the performance of the ResNet-18 can be improved either by fine-tuning the hyperparameters [15, 16] or by adjusting the layers [17]. The main aim of this research will, therefore, be to improve the accuracy of detecting induction motor bearing faults and reduce the computational cost of the ResNet-18 model through a combination of hyperparameter tuning and layer modification.

3. Methodology

3.1. Bearing Dataset

In this work, the Paderborn University (PU) bearing dataset is used since it is a well-known benchmark dataset widely used for research in fault diagnosis and prognostics. It consists of synchronously measured motor currents and vibration signals obtained from a test rig with high resolution and sampling rate. The dataset consists of 6 healthy states for reference and 26 damaged states of ball bearings of type 6203, of which 12 are artificially damaged bearings, and 14 are bearings with real damages caused by accelerated lifetime tests [18]. To ensure the robustness of condition monitoring methods at different operating conditions, the PU data was collected using accelerometers placed on the test rig, capturing the vibrations generated by the rotating bearings under different combinations of operating conditions, i.e. at different rotational speeds of the drive system, radial force onto the test bearing and load torque, while temperature was kept at 45-50 °C during all experiments [18]. The dataset diversity was essential for training the model to recognize and classify various fault patterns accurately. For each of the settings, 20 measurements of 4 seconds each were recorded and saved as MATLAB files [19].

Subsequently, the frequency and time values of the PU pre-processed data are arranged into a 2D matrix format to come up with images to be fed into the deep learning algorithm since CNNs are known to be great image classifiers. The images function in MATLAB, which scales image data to the full range of the current colormap and displays the image, is employed to generate a color-coded image, where the intensity of vibration is depicted across different frequencies and time intervals [20]. The original image obtained was of 656 rows, 875 columns, and 3 Red Green Blue channels, i.e. size [656x875x3]. The images were scaled and cropped to 224x224x3 RGB images to enhance the recognition of the target objects by the model and also avoid overfitting the network. The improved ResNet-18 can then interpret these images to identify patterns or anomalies indicative of faults in induction motor bearings, thereby contributing to real-time classification.

The faults from the PU dataset included healthy, outer race, inner race, and a combination of inner and outer race faults, each introduced at different levels of severity to simulate realistic degradation scenarios. Each fault type manifests distinct vibration patterns, as shown in Figure 1. A total of 2632 images were created for the 4 different classes, i.e. one class for the healthy state and three classes for the faulty states. Subsets for training, validating, and testing were created from the bearing images dataset to facilitate model evaluation, training, and validation. 15% of the images from the 4 classes were left aside for testing, while the remaining

2464 images were randomly divided to end up with 70% for training and 15% for validation. These images are then uploaded to MATLAB's Deep Network Designer App and are used to train and test the various models.

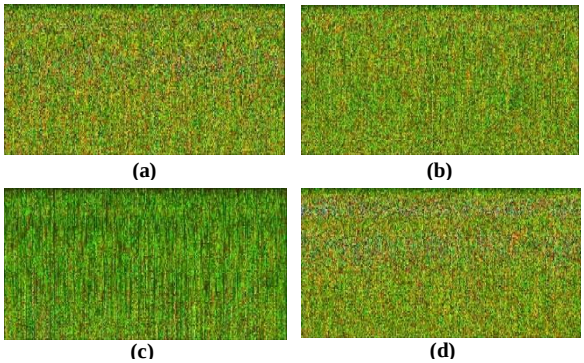


Fig. 1 Bearing data images for (a) Combination fault (N09_M07_F10_KB27_3), (b) Healthy (N15_M01_F10_K001_19), (c) Inner race fault (N09_M07_F10_KI01_7), and (d) Outer race fault (N15_M07_F04_KA05_10).

3.2. ResNet-18 Network Architecture

ResNet-18 is a CNN architecture that was introduced by Kaiming He et al. [9] after researchers discovered that merely adding more layers could lead to performance degradation.

ResNet-18 introduces residual connections, also known as skip connections, which allow the network to learn residual functions in order to address the challenge of vanishing gradients that arises when training very deep neural networks. It is well known for achieving good accuracy on image classification tasks while maintaining a relatively low computational cost compared to other deep networks.

The architecture of ResNet-18 consists of a convolutional layer followed by a max pooling layer and residual blocks, each containing a series of two or three convolutional layers with batch normalization and activation functions. The final layers include global average pooling and a fully connected layer for classification, as shown in Figure 2.

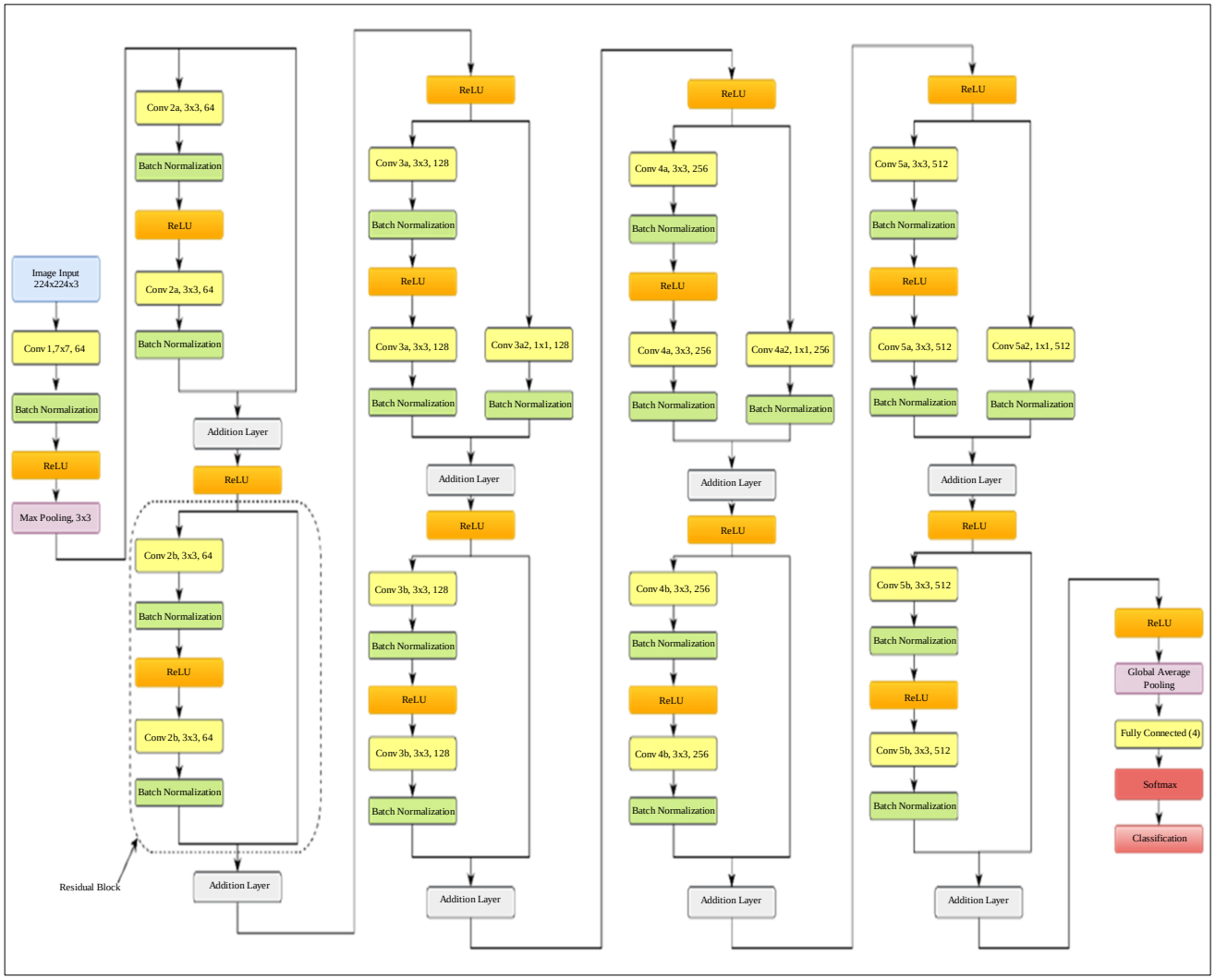


Fig. 2 The ResNet-18 architecture

3.2.1. Image Input Layer

This layer accepts Two-Dimensional (2-D) images of fixed size [224x224x3], i.e. RGB image with 224x224 pixels and 3 color channels as the neural network's input, ensuring that the input images are compatible with the subsequent layers of the network. Z-score data normalization is applied to transform the mean of all values to become zero and the standard deviation to become one.

The purpose of z-score normalization is to make different features more directly comparable and to ensure that they contribute equally to the learning process [21, 22]. It helps improve the convergence and stability of the training process, especially when features have different scales or distributions. The z-score normalization transformation for feature x is expressed as:

$$Z=(x-\mu)/\sigma \quad (1)$$

Where z is the standardized value (z-score) of the feature, x is the original feature value, μ is the mean of the feature across the training dataset, and σ is the standard deviation of where z is the standardized value (z-score) of the feature, x is the original feature value, μ is the mean of the feature across the training dataset, and σ is the standard deviation of the feature across the training dataset. By subtracting the mean, the data distribution is shifted to have a mean of zero, thus helping to prevent bias while dividing the data by the standard deviation scales, ensuring that features with different scales contribute equally during training.

3.2.2. Convolution Layer

The 2-D convolutional (conv) layer performs feature extraction by applying sliding convolutional filters (kernels) to the 2-D input to produce feature maps that highlight specific patterns or features present in the input data [23]. This layer convolves the input by calculating the dot product of the weights and the input, moving the filters along the input both vertically and horizontally, and then adding a bias term.

The weights of the filters in the convolutional layer are learnable parameters, which are optimized during the training process using Stochastic Gradient Descent with Momentum (SGDM) to minimize the loss function, enabling the network to learn meaningful representations of the input image. The convolutional layers incorporate padding and stride parameters to control the spatial dimensions of the output feature maps.

The padding adds extra border pixels around the input image, preserving spatial information and mitigating the reduction in spatial dimensions caused by the convolution operation. Stride influences the output feature maps' spatial resolution by determining the step size of the filter that slides over the input image. The first convolutional layer of the ResNet-18 has 64 filters with a kernel size of 7x7, stride of 2, and padding of 3. The data format is a string of spatial (S), Channel (C), and batch (B) characters, where each character describes the type of the corresponding data dimension. This convolution layer feeds [112(S) x 112(S) x 64(C) x 1(B)] to the batch normalization and Rectified Linear Unit (ReLU) activation layers that come after it. If nxn is the image size, fxf is the kernel size, p is the padding, s is the stride, C represents the number of channels, and N is the batch size, the output of the convolution is calculated as follows: [23]

$$\text{Output} = \left[\frac{n+2p-f}{s}+1\right] \times \left[\frac{n+2p-f}{s}+1\right] \times C \times N \quad (2)$$

3.2.3. Batch Normalization (BN) Layer

This layer is used between the convolutional layers and ReLU layers to minimize internal covariate shift, accelerate training of the convolutional neural network, and reduce network initialization sensitivity [24]. The batch normalization layer normalizes a mini-batch of data by calculating the mean (μ) and standard deviation (σ) of the activations across all observations for each channel separately. Each activation value (x_i) in the mini-batch is normalized using the formula:

$$\widehat{x_i} = \frac{x_i-\mu}{\sqrt{\sigma^2-\varepsilon}} \quad (3)$$

Where ε is a relatively small constant added to the denominator to avoid division by zero and improve numerical stability, after normalization, the input is scaled using a learnable scale factor γ and shifted by a learnable offset β.

$$y_i = \gamma.\widehat{x_i} + \beta \quad (4)$$

These learnable parameters allow the network to recover the original scale and shift the data after normalization, thus restoring the data to the desired range. They allow the network to maintain flexibility and avoid excessive normalization effects.

3.2.4. Rectified Linear Unit (ReLU) Layer

A BN layer and a ReLU activation function follow each convolutional layer. The key role of the ReLU activation function is to introduce non-linearity into the model, enabling the network to recognize intricate patterns and correlations. It does this by performing a threshold operation on each element of the input, where any value less than zero is set to zero [25], i.e.

$$f(x) = \max(0,x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (5)$$

The ReLU is a popular choice in neural networks due to its computational efficiency [26] and ability to mitigate the vanishing gradient issue.

3.2.5. Pooling Layer

Pooling is a technique used to reduce the spatial dimensions of feature maps and assists in extracting important features [27] while lowering computational complexity. The input is divided into small pooling regions, and the maximum of each region is then calculated by a 2-D max pooling layer, which conducts down-sampling. The max pooling layer of ResNet-18 features a 3 x 3 pool size, a stride of 2, and padding of 1. It reduces the influence of noise and is effective for image recognition, where identifying the most prominent edges or corners is crucial [28].

The ResNet-18 architecture also has a 2-D global average pooling layer, which is carried out down-sampling by calculating the mean of the input's height and width dimensions. It calculates the average of all activation values within the entire feature map and reduces the feature map to a single value for each channel. This layer preserves spatial information by considering all activations within the feature map. It works well with fully connected layers, effectively reducing feature maps before the final classification layer. It minimizes the number of parameters and helps avoid overfitting [29].

3.2.6. Residual Block

The residual block is a structure within the ResNet-18 architecture that includes a shortcut or skip connection that connects the output of an earlier layer to a later layer, bypassing intermediate convolutional operations. These blocks allow information to flow more smoothly through the deep network.

There are two types of skip connections: identity shortcut connection and convolutional shortcut connection, [30] as shown in Figure 3. The identity shortcut connection performs an identity mapping (no transformation), ensuring that the information passing through the shortcut remains consistent with the overall network's structure.

Convolutional shortcut connections are similar to identity blocks but include a convolutional layer in the shortcut path. The purpose of this convolutional layer is to adjust the dimensions so that the input and output match. The output of the identity shortcut connection is defined by the following equation: [23]

$$y = F(x) + x \tag{6}$$

While the output of the convolutional shortcut connection is given by [23],

$$y = F(x) + W.x \tag{7}$$

Where x is the input of the residual block, F(x) is a non-linear mapping function of the stacked operation layers in a residual unit, and W represents a linear projection. The residual block is followed by an addition layer, which adds inputs from multiple neural network layers element-wise, allowing for both the original input and the transformed output to be combined before feeding the output to the ReLU.

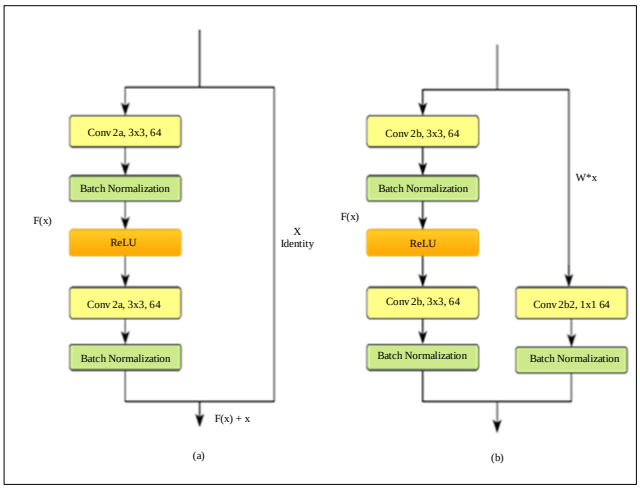


Fig. 3 Types of shortcut connections: (a) Identity shortcut connection, and (b) Convolutional shortcut connection.

3.2.7. Fully Connected Layer

A Fully Connected (FC) layer multiplies the input from the residual block by a weight matrix. Then, it adds a bias vector, performing a linear transformation to produce the four output classes for classification. A softmax layer then outputs probabilities for the various classes. Equation (8) illustrates how the softmax function uses the exponential function on each input and then normalizes the outputs to make sure they add up to 100% [31].

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{8}$$

Where $\sigma(z_i)$ is the output of and z_i is the input to the softmax function for the i-th element, n represents the total number of elements in the input vector, and j denotes the output class.

The classification layer is the final step in the network, where the output from the softmax layer is used to make a prediction. The final classification is determined by selecting the class with the highest probability. Metrics used to evaluate the performance of the model, such as accuracy, precision, recall, and F1-score, are then computed by comparing the output of the classification layer with the ground truth.

3.3. Layer Modification of ResNet-18

This study investigates the impact of specific structural modifications on the performance of the ResNet-18 architecture. The goal is to examine how the following changes affect the model's accuracy, computational efficiency, and generalization capability.

3.3.1. Removal of Residual Blocks

Residual blocks were removed one block at a time from the ResNet-18 architecture, attempting to gain insight into the network's behavior and performance under varying levels of depth and complexity by measuring the impact on model accuracy and convergence.

Firstly, residual block 5b was removed, followed by the removal of residual block 5a. Though the training time was reduced, it was evident that the test accuracy was becoming less precise. All the 'b' blocks were then removed, and since the results only got worse, the modification strategy had to be changed. The addition of several convolutional layers was the next option to experiment with.

3.3.2. Addition of Convolutional Layers

Two convolutional layers of filter size 16 and 32, each followed by BN and ReLU layers, were added before the first residual block, allowing the network to learn more low-level features from the input data. This was an attempt to help the network extract more relevant information, leading to better performance and potentially faster convergence during training.

The additional convolutional layers were given a stride of 2 in order to reduce the spatial dimensions of the feature maps earlier in the network, aiming to decrease the computational cost of the subsequent layers as they had to process feature maps with smaller spatial sizes.

3.3.3. Removal of Shortcut Connections

This alteration led to a standard feedforward architecture with sequential layers, similar to the traditional deep neural network, allowing the authors to observe how much the residual structure contributes to learning efficiency and accuracy.

The shortcut connections are the defining characteristics of ResNet-18 that allow gradients to flow more easily during backpropagation. When removed, the model may suffer from vanishing gradients, leading to difficulty in learning deeper features, thus slowing the training time and reducing the accuracy.

3.3.4. Addition of Fully Connected Layers

Fully connected layers of various output sizes were added to the network after the global average pooling layer. It is known that adding a fully connected layer can increase the model's capacity to learn high-level features before the final classification.

However, the risk of overfitting and higher computational time was to be mitigated by adding a BN layer to regularize the output, and a ReLU could also be introduced to capture non-linearity and allow the network to learn more complex patterns.

3.3.5. Addition of Squeeze and Excitation Block

Five Squeeze and Excitation (SE) blocks were added to the most promising network after the experiments above in an attempt to improve the results found [32]. Figure 4 shows the proposed model that gave the best results.

The squeeze operation involved using global average pooling [33] to reduce each channel in the feature map to a single value, creating a compact representation of the entire feature map. The squeeze operation z_c can be represented as: [33].

$$Z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j) \quad (9)$$

Where u_c represents the feature map for the c-th channel, H and W are the height and width of the feature map, and F_{sq} denotes the squeeze function.

The FC layer followed the squeeze operation to reduce the dimensionality by a factor of r, a ReLU to introduce non-linearity, another FC to restore the initial dimensionality, and a sigmoid function to produce a scaling factor between 0 and 1 for each channel [34].

The excitation operation applies a gating mechanism to the squeezed feature descriptor, followed by a sigmoid activation to obtain channel-wise scaling factors. The excitation operation s_c can be formulated as:

$$S_c = F_{ex}(z, W) = \sigma\big(g(z, W)\big) = \sigma(W_2 \delta(W_1 z)) \quad (10)$$

Where F_{ex} denotes the excitation function, g is a gating mechanism (FC with ReLU activation), W represents the learnable parameters of the gating mechanism, σ is the sigmoid activation function, δ refers to the ReLU function, $W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ *and* $W_2 \in \mathbb{R}^{C \times \frac{C}{r}}$ [33].

The SE block shown in Figure 5 then recalibrates the output of the residual block before adding it to the shortcut connection. Its final output is obtained by:

$$\tilde{x_c} = F_{scale}(u_c, s_c) = s_c . u_c \quad (11)$$

Where $\tilde{x_c}$ represents the output feature map for the c-th channel, $F_{scale}(u_c, s_c)$ refers to channel-wise multiplication between the scalar s_c and the feature map $u_c \in \mathbb{R}^{H \times W}$ [33].

This Squeeze and Excitation Residual Network (SE-ResNet) architecture, having shown exemplary performance, was chosen for further experimentation with different hyperparameters, i.e. learning rate, batch size, number of epochs, and different solvers, to find the optimal configuration for the model.

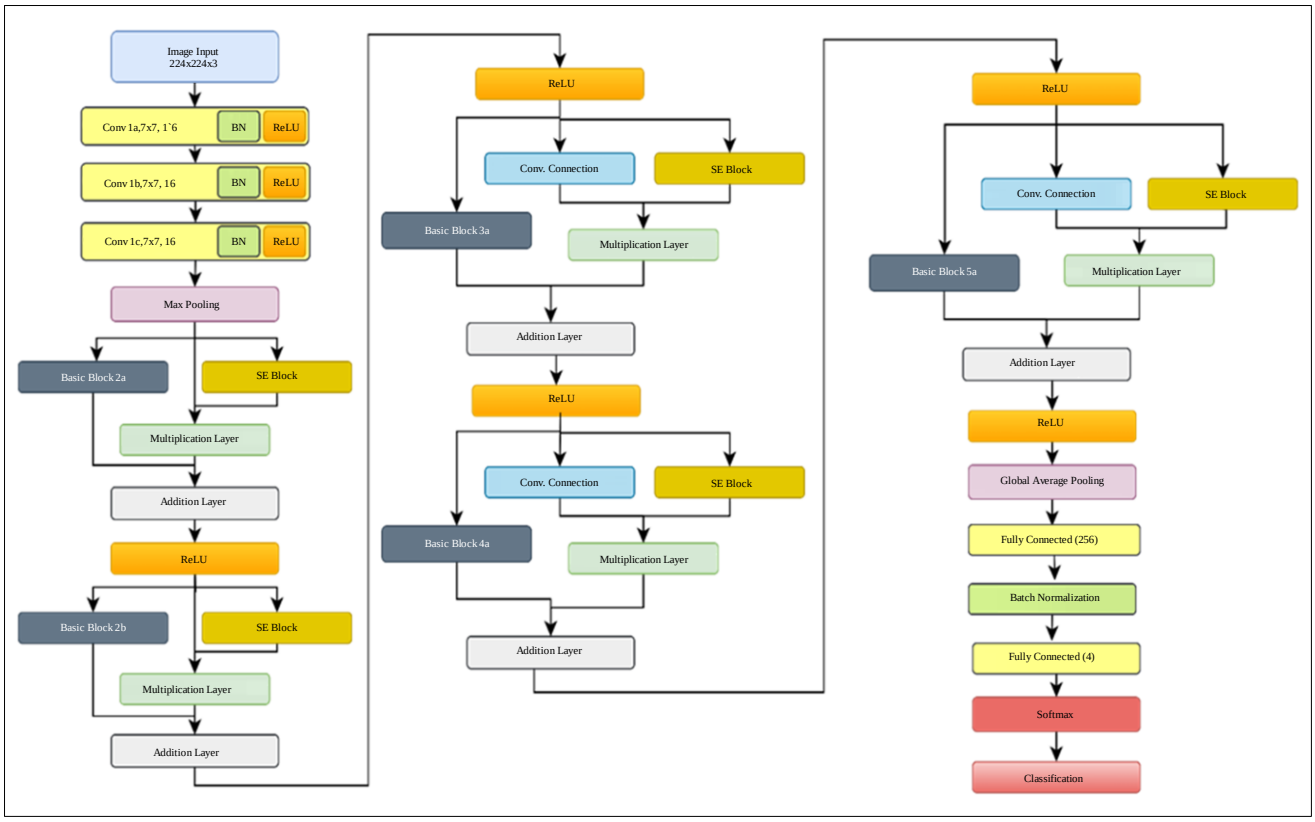


Fig. 4 The proposed squeeze and excitation ResNet

3.4. Hyperparameter Tuning

This section describes the approach used for hyperparameter tuning to identify the optimal configuration for the proposed architecture. The process aimed to enhance the model's performance by adjusting key parameters such as batch size, the learning rate, the number of epochs, and the choice of optimization algorithms (solvers). Other hyperparameters like L2 regularization, momentum, and number of hidden units were also tuned to ensure the best performance for the model.

The learning rate, which determines the step size at each iteration during optimization, was a critical hyperparameter in this work. A well-chosen learning rate significantly influences the training speed. A high learning rate might lead you past the optimal solution and into areas of higher loss, , while a small learning rate, will slow convergence or even get stuck in suboptimal solutions [35]. Using a logarithmic scale, different learning rates were experimented on.

Batch size represents the number of samples processed in one iteration before updating the model weights. The optimal batch size is also identified through experimentation varying from 32 to 256. On the other hand, the number of epochs refers to the number of times the entire dataset is used for training. This is selected carefully to ensure balance and avoid overfitting or underfitting. Optimization algorithms like Stochastic Gradient Descent with Momentum (SGDM), Adaptive Moment Estimation (Adam), and Root Mean Square Propagation (RMSProp) are responsible for updating the internal parameters (weights and biases) of the model based on the calculated loss function. Each solver was explored to choose the best-performing optimizer while keeping all parameters, such as learning rate, batch size, and number of epochs, to mention a few, constant.

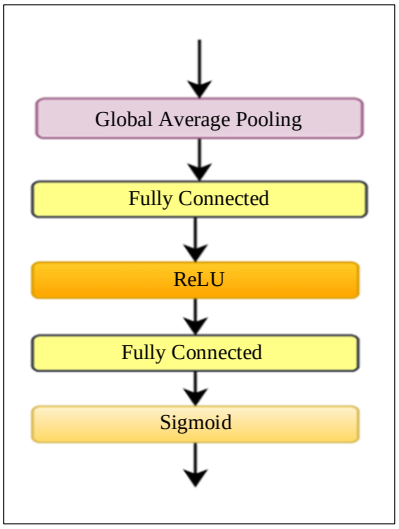


Fig. 5 The squeeze and excitation block

3.5. Bayesian Optimization of SE-ResNet

Bayesian Optimization (BO) is a statistical method for optimizing objective functions. It improves the search speed by using past performances of hyperparameters to determine future decisions [36]. The Bayesian Optimized SE-ResNet (BOSE-ResNet) was used to find optimal hyperparameters efficiently using fewer evaluations than the exhaustive sweep performed on the SE-ResNet.

Firstly, a range of possible values for the initial learning rate (0.007 – 0.012), the momentum (0.8 – 0.98), and L2-regularization (1×10^{-5} – 1×10^{-2}) were chosen based on the results from the hyperparameter tuning of the SE-ResNet. SGDM optimization algorithm was chosen, and the minibatch size was held constant at 128. The BO strategy was then used to train the model, which is done by selecting a random set of parameters and evaluating the performance. It then incorporates the new data point and updates the belief about the objective function. The next set of hyperparameters to be evaluated was repeatedly determined based on the updated belief until the stopping criterion of 30 trials was met. The maximum run time was set to infinity to ensure all 30 trials with different hyperparameter combinations were sampled.

3.6. Performance Metrics

To ensure unbiased performance evaluation and help avoid overfitting, the dataset was divided into a 70% train set, a 15% validation set, and a 15% test set. This ensured that the model was able to be tested on unseen data, thus providing insight into its ability to generalize. The performance of the proposed fault diagnosis model was evaluated using primary metrics such as accuracy, precision, F1 score, and recall. These were evaluated as follows: [37].

$$\text{Accuracy}=\frac{TP+TN}{TP+TN+FP+FN} \quad (12)$$

$$\text{Precision}=\frac{TP}{TP+FP} \quad (13)$$

$$\text{Recall (Sensitivity)}=\frac{TP}{TP+FN} \quad (14)$$

$$\text{F1-Score}=2\text{x}\frac{Precision\times Recall}{Precision+Recall} \quad (15)$$

True Positive (TP) refers to the number of cases where the model correctly predicted a positive outcome, True Negative (TN) indicates the number of cases where the model correctly predicted a negative outcome, False Positive (FP) is the number of cases where the model incorrectly predicted a positive outcome, and False Negative (FN) represents the number of cases where the model incorrectly predicted a negative outcome [37].

Using the same graph to plot the training and validation accuracy and loss curves, it was possible to assess whether the model was overfitting or underfitting. High training performance accompanied by low validation performance meant the model was overfitting while underfitting was when both training and validation performance was low. The original ResNet-18 model, without any modifications, served as the baseline; therefore, the performance of each modified ResNet-18 architecture was compared against it. The computational cost was also assessed by comparing the training time of the different models.

3.7. Software and Hardware Environment

The experiments were performed on MATLAB R2023a Deep Network Designer App using a device with the following specifications: Windows 11 Home, 11th Generation Intel (R) Core (TM) i7-1165G7 @ 2.80GHz processor, Random Access Memory (RAM) of 16 GB, and a storage of 512 GB SSD. Even though this device does not have a Graphics Processing Unit (GPU), its specifications allowed for the assessment of computational speed and training efficiency, providing insights into the model's performance in a typical environment.

Since CPUs are generally slower for deep learning training compared to GPUs due to their limited parallel processing capabilities, [4] a computationally efficient model on CPU will perform exceptionally well on GPU. However, since most industries have access to devices similar to the one used in this paper, this study shows that there will be no need to invest in a device specifically for condition monitoring of induction motors.

4. Results and Discussion

ResNet-18's concept of residual connections was introduced to enable much deeper architectures while maintaining efficient training. Its relatively shallow depth makes it computationally more efficient than deeper versions of ResNets while still providing the benefits of residual learning. With its simplicity in mind, it was chosen as the starting point for layer modification and hyperparameter tuning.

4.1. Performance of ResNet-18 Architecture

The validation dataset was used to verify the ResNet-18 model after it had been trained on the training dataset. Different learning rates varied on a logarithmic scale, i.e. 0.001, 0.01, 0.1, and 1, were experimented on, while other parameters such as a mini-batch-size of 128, validation frequency of 14 iterations, a total of 20 epochs, and the SGDM solver were held constant. A learning rate of 0.01 was selected since this experiment had the highest validation accuracy. The model was then tested on completely unseen images, and the test accuracy was 78.91%.

Figure 6 shows the results of the training and validation graphs obtained from training the ResNet-18 model using the

selected parameters above. In contrast, Table 1 shows the results of the training accuracy, validation accuracy, testing accuracy, and the time it took to perform the experiments when the learning rate was varied. The confusion matrices for the training data, the validation data, and the testing data when the learning rate is 0.01 are shown in Figure 7.

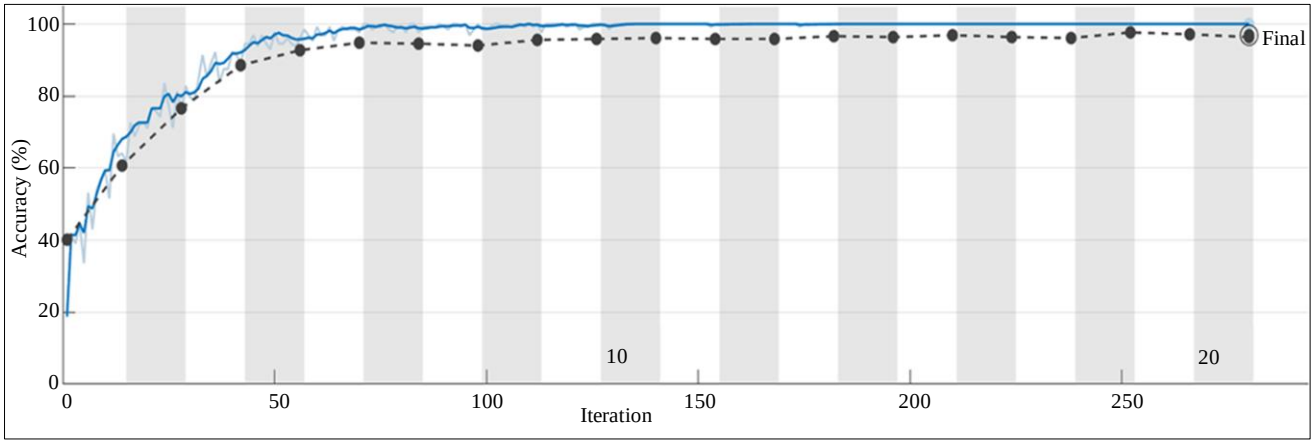
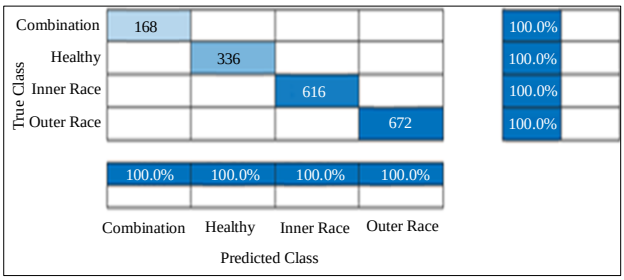


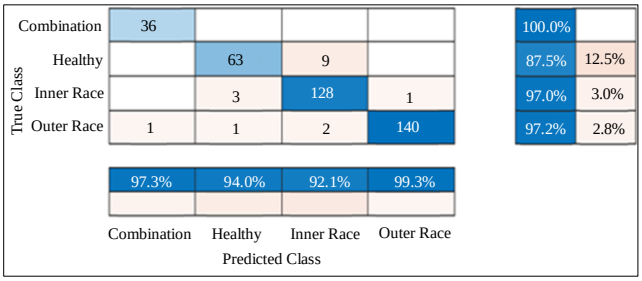
Fig. 6 Training (blue) and validation (black) graphs of ResNet-18

Table 1. Performance of ResNet-18 with various learning rates

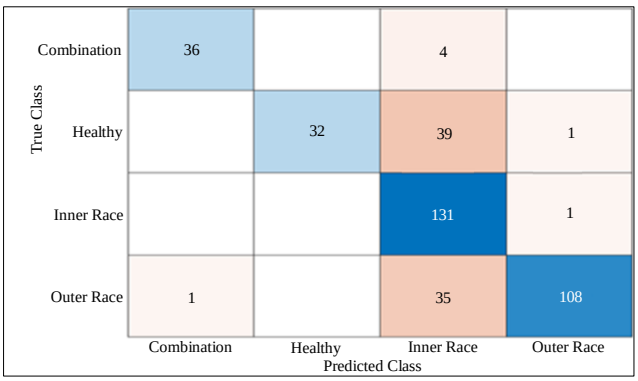
Sl. No.	Elapsed Time	Learning Rate	Training Accuracy	Validation Accuracy	Test Accuracy
1	93 min 18 sec	0.001	100%	80.47%	35.16%
2	92 min 40 sec	0.01	100%	95.57%	78.91%
3	94 min 9 sec	0.1	84.35%	82.81%	72.40%
4	91 min 33 sec	1	37.75%	36.98%	38.28%



(a)



(b)



(c)

Fig. 7 Confusion matrix for (a) Training data, (b) Validation data, and (c) Testing data when using ResNet-18 architecture.

4.2. Model Performance after Modifications to ResNet-18 Architecture

From the experiments performed on the ResNet-18 architecture, it was evident that this network was not able to perform well on unseen data. This led to its modification in an attempt to increase its efficiency. Firstly, layer 5b was removed. This led to a decrease in computational time, as expected, but the test accuracy was negatively affected. Removing layer 5a further decreased the computational time, but the test accuracy did not improve. This is because layers 5a and 5b have the largest filter size of 512 for the ResNet-18 network; thus, removing these layers reduces the overall capacity of the model to learn complex representations from the input data.

Without the final residual blocks, the model overfits the training data, as it cannot learn more complex patterns that generalize well to unseen test data [38]. Having noted the importance of the 512-filter size, the original ResNet-18 was modified to exclude layers 2b, 3b, 4b and 5b. The test accuracy further decreased, prompting the return of layer 2b to the network. With only 3b, 4b and 5b removed from the network, the test accuracy was slightly better than when all the 'b' layers were removed.

Though removing one residual block at a time led to reduced training time, it also reduced the model's capacity to learn complex features, leading to lower test accuracy. Two convolutional layers of filter sizes 16 and 32, each followed by BN and ReLU, were added to the modified network immediately after the input layer. These layers captured more fine-grained and informative features from the input, leading to faster convergence during training and better generalization of the test set.

An FC layer of output size 256, followed by a BN layer, was inserted after the global average pooling layer, which improved the network's performance. However, adding more FC layers to the network reduced the model's performance. All the skip connections were then removed from the original ResNet-18 architecture, giving a test accuracy of 97.92%, but the computational time was very high. Adding two convolutional layers of filter sizes 16 and 32 reduced the computational cost and slightly increased the test accuracy.

Squeeze and excitation blocks were then added to the modified ResNet-18 network, which had two convolutional layers added to it and layers 3b, 4b, and 5b removed. This network increased the test accuracy to 98.698%. This Squeeze and Excitation Residual Network (SE-ResNet) model is selected for further experimentation, and hyperparameters such as learning rate, batch size, number of epochs, and the type of solver are fine-tuned to achieve higher performance. Table 2 shows some of the modifications done to the ResNet-18 architecture to improve its performance.

Table 2. Model performance for different modifications to ResNet-18

Sl. No.	Network	Learning Rate	No. of Iterations	Batch Size	Validation Frequency	Validation Accuracy	Test Accuracy	Time Elapsed
1	Original ResNet-18	0.01	20	128	14	95.57%	78.906%	92 min 40 sec
2	Original ResNet-18 – Layer 5b	0.01	20	128	14	98.18%	73.177%	83 mins 19 sec
3	Original ResNet-18 – Layers 5b and 5a	0.01	20	128	14	96.35%	75.521%	79 mins 27 sec
4	Original ResNet-18 – Layers 5b, 4b and 3b	0.01	20	128	14	96.61%	75.781%	70 mins 37 sec
5	Original ResNet-18 - All 'b' Layers	0.01	20	128	14	93.49%	63.802%	63 mins 51 sec
6	Original ResNet-18 + 2 conv. layers of filter sizes 16 and 32, each followed by BN and ReLU	0.01	20	128	14	96.61%	96.875%	28 mins 23 sec
7	Original ResNet-18 + 2 conv. layers of filter sizes 16 and 32, each followed by BN and ReLU – all 'b' layers	0.01	20	128	14	96.09%	95.050%	24mins 13sec
8	Original ResNet-18 – Layers 5b, 4b & 3b + 2 conv. layers (16,32) + FC256 (followed by BN)	0.01	20	128	14	97.92%	97.656%	24 mins 35 sec

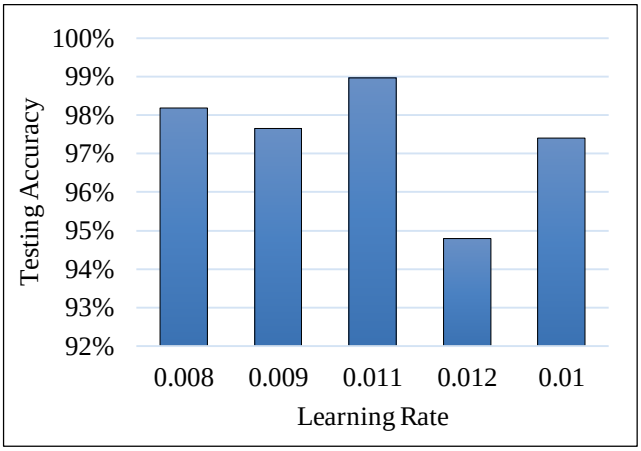
9	Original ResNet-18 – Layers 5b, 4b & 3b + 2 conv. layers (16,32) + FC256 + FC128 + FC64 (each + BN)	0.01	20	128	14	94.27%	96.354%	24 mins 39 sec
10	Original ResNet-18 – Layers 5b, 4b & 3b + 2 conv. layers (16,32)	0.01	20	128	14	96.88%	98.177%	24 mins 21 sec
11	Original ResNet-18 – all skip connections	0.01	20	128	14	97.14%	97.917%	76 mins 46 sec
12	Original ResNet-18 – all skip connections + 2 conv. layers (16,32)	0.01	20	128	14	97.66%	98.177%	28 mins 14 sec
13	Original ResNet-18 – all skip connections + 5 SE Blocks + 2conv. layers (16,32)	0.01	20	128	14	96.35%	98.438%	29 mins 12 sec
14	Original ResNet-18 – Layers 5b, 4b & 3b + 2 conv. layers (16,32) + 5 SE Blocks + FC256	0.01	20	128	14	97.66%	98.698%	22 mins 52 sec

4.3. Hyperparameter Tuning of the Proposed Model

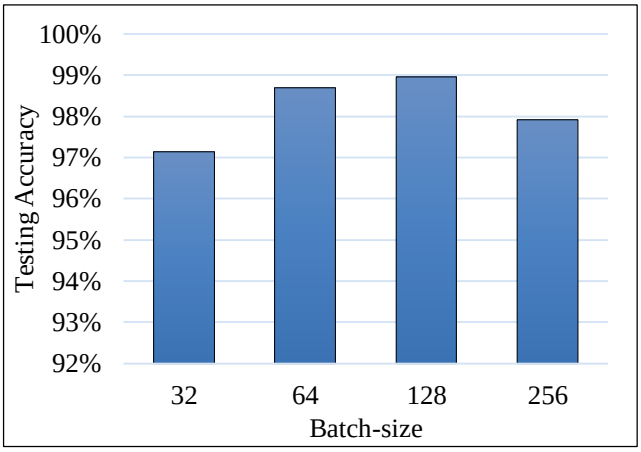
The proposed SE-ResNet was subjected to further experimentation to improve its test accuracy. In order to identify the optimal conditions, several experiments with different combinations of parameters were performed, and the results are summarized in Figure 8**.** The learning rates experimented on were 0.008, 0.009, 0.01, 0.011 and 0.012. A learning rate of 0.011 gave the best results. This parameter was then held constant while batch sizes 32, 64, 128, and 256 were experimented on. The best performance was seen with a batch size of 128.

From the graphs, it was evident that from around the 10th epoch, the training curve remained constant. Different epochs of 10, 15, 20, and 50 were therefore tested to find out the impact of the number of epochs on accuracy. It was noted that increasing the number of epochs did not improve the accuracy. The number of epochs selected to ensure that the highest accuracy was obtained in the least time was 20 epochs.

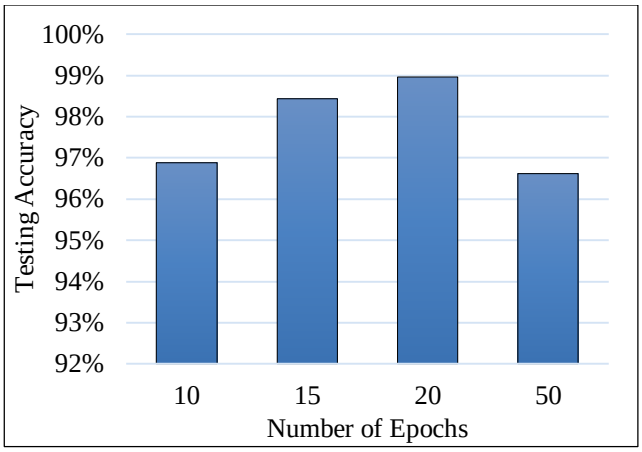
Finally, keeping all other parameters constant, different solvers were tried, i.e. SGDM, Adam and RMSProp. Table 3 shows the results from hyperparameter tuning. The SGDM is computationally cheap and relatively easy to implement. However, it can be slow to converge and requires careful tuning of the learning rate so as not to oscillate or lead to suboptimal solutions. Adam and RMSProp try to address these challenges. Adam incorporates adaptive learning rates for each parameter, estimating both the first and second moments of the gradients, leading to faster convergence and potentially smoother optimization compared to SGDM.



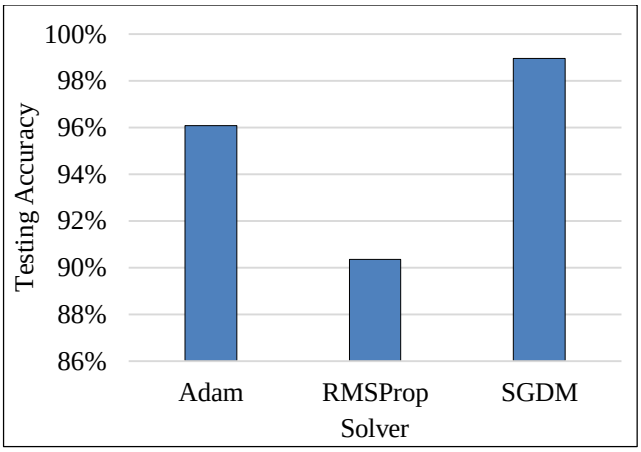
(a)



(b)



(c)



(d)

Fig. 8 Graphs of testing accuracy against (a) Learning rate, (b) Batch-size, (c) Number of epochs, and (d) Solver.

Table 3. Hyperparameter tuning of SE-ResNET

Sl. No.	Learning Rate	Number of Epochs	Batch-Size	Weight Decay	Validation Accuracy	Training Accuracy	Time Elapsed	Solver
1	0.008	20	128	0.0001	98.96%	98.177%	31 min 48 sec	SGDM
2	0.009	20	128	0.0001	98.96%	97.660%	32 min 35 sec	SGDM
3	0.011	20	128	0.0001	99.22%	98.958%	32 min 53 sec	SGDM
4	0.012	20	128	0.0001	95.83%	94.790%	33 min 22 sec	SGDM
5	0.01	20	128	0.0001	98.44%	97.396%	36 min 43 sec	SGDM
6	0.011	20	32	0.0001	96.35%	97.135%	52 mins 18 sec	SGDM
7	0.011	20	64	0.0001	98.70%	98.698%	35 mins 35 sec	SGDM
8	0.011	20	256	0.0001	98.70%	97.917%	29 mins 38 sec	SGDM
9	0.011	10	128	0.0001	97.40%	96.875%	17 mins 28 sec	SGDM
10	0.011	15	128	0.0001	98.18%	98.438%	33 mins 17 sec	SGDM
11	0.011	50	128	0.0001	98.18%	96.615%	100 mins 23 sec	SGDM
12	0.011	20	128	0.0001	96.35%	96.097%	42 mins 38 sec	SGDM
13	0.011	20	128	0.0001	98.96%	98.177%	31 min 48 sec	SGDM
14	0.011	20	128	0.0001	98.96%	97.660%	32 min 35 sec	Adam
15	0.011	20	128	0.0001	90.89%	90.365%	24 mins 30 sec	RMSProp
16	0.011	20	128	0.0005	96.88%	97.135%	24 mins 42 sec	SGDM
17	0.011	20	128	0.00005	97.14%	97.396%	26 mins 55 sec	SGDM
18	0.011	20	128	0.00001	96.09%	97.396%	41 mins 46 sec	SGDM
19	0.011	20	128	0.001	94.79%	97.917%	25 mins 51 sec	SGDM
20	0.011	20	128	0.01	97.40%	98.177%	23 mins 10 sec	SGDM

RMSProp focuses on the recent history of gradients using a decaying average of squared gradients, thus helping to adjust learning rates dynamically based on the parameter's update history. The SGDM, which adjusts the model parameters iteratively in the direction opposite the loss function's gradient, ensuring the model gradually moves towards minimizing the loss, was the best-performing solver and was therefore selected for the proposed model. With a batch size of 128 examples, weight decay of 0.0001, momentum of 0.9, and stochastic gradient descent with momentum, the SE-ResNet model was trained to produce the best results. The model needed to learn from this tiny bit of weight decay in order to decrease its training error. The weight decay term is added to the loss function, and during training, it encourages the model to learn simpler patterns by penalizing large weight values. The equation for velocity is [31],

$$v_{i+1} = 0.9.v_i - 0.0001.\lambda.w_i - \lambda.\left\langle \frac{\partial L}{\partial w}\middle| w_i \right\rangle D_i \quad (16)$$

Where, v_{i+1} is the updated velocity, v_i is the current velocity, λ is the learning rate, w_i is the current weight, $\partial L/\partial w|_{w_i}$is the gradient of the loss function L with respect to the weight w at the current iteration i, and $\left\langle \frac{\partial L}{\partial w}\middle| w_i \right\rangle$denotes the average over the mini-batch D_i. The momentum and weight

decay are 0.9 and 0.0001, respectively [26]. The equation of the updated weight can be calculated as [31]:

$$w_{i+1} = w_i + v_{i+1} \tag{17}$$

Figure 9 shows the testing confusion matrix, while Figure 10 shows the training and validation confusion matrices. The training and validation graphs were plotted on the same axis, providing a clear and direct comparison between training and validation metrics such as accuracy and loss. It was also helpful in monitoring overfitting and underfitting. Figure 11 and Figure 12 show the accuracy and loss graphs, respectively. Other performance metrics of the SE-ResNet are presented in Table 4.

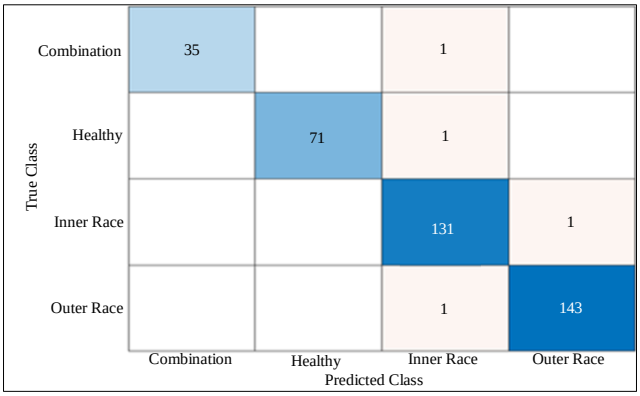
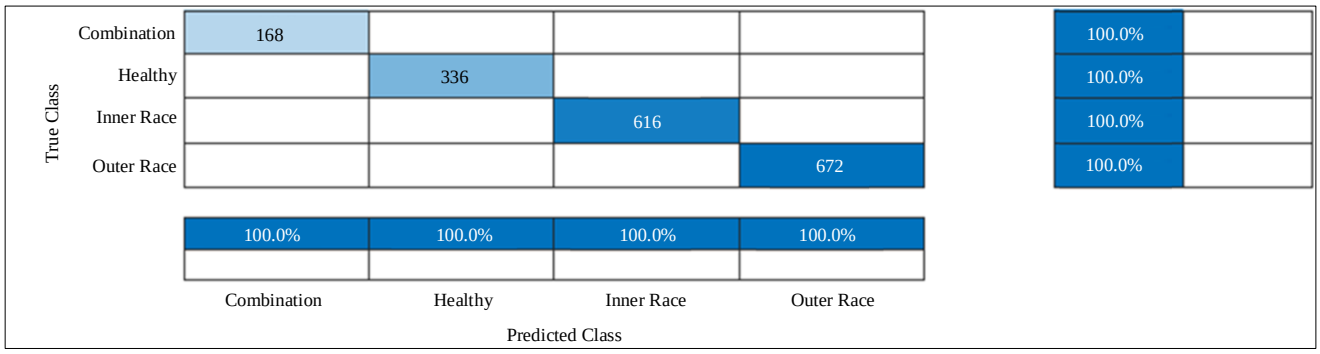
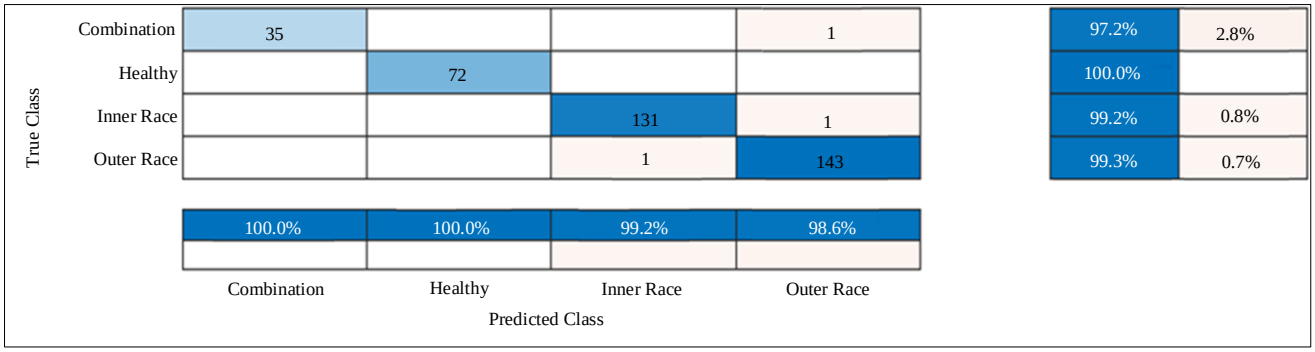


Fig. 9 Confusion matrix of testing dataset for the SE-ResNet



(a)



(b)

Fig. 10 Confusion matrix of (a) Training, and (b) Validation datasets for the SE-ResNet.

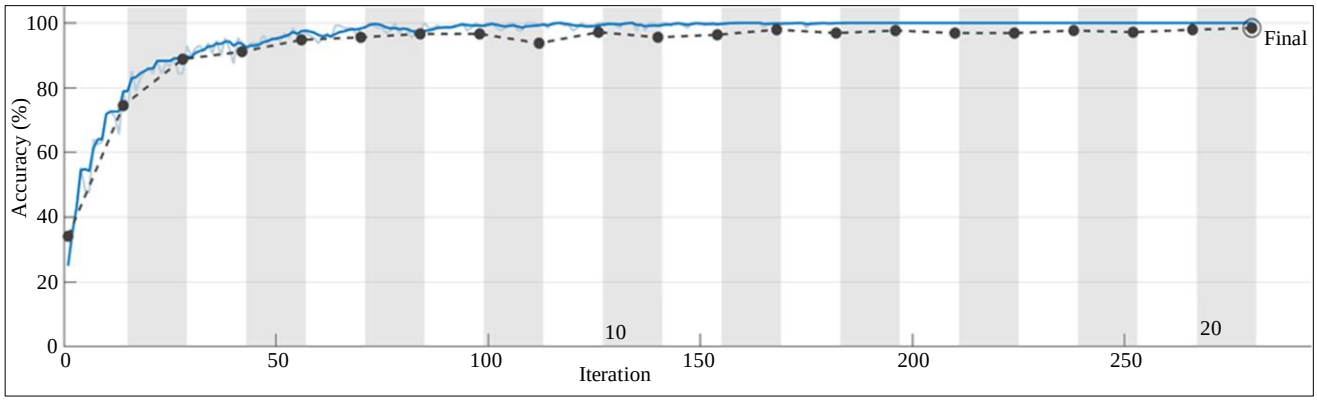


Fig. 11 Training (blue) and validation (black) accuracy graph for the SE-ResNet

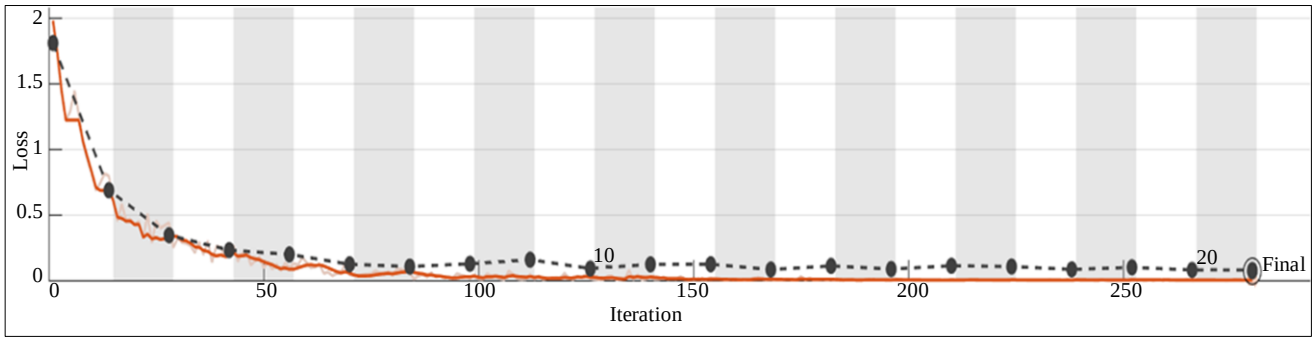


Fig. 12 Training (red) and validation (black) loss graph for the SE-ResNet

Table 4. Performance metrics for SE-ResNet

Class	Precision	Recall	F1-score
Combination	1.0000	0.9722	0.9859
Healthy	1.0000	0.9861	0.9930
Inner Race	0.9776	0.9924	0.9850
Outer Race	0.9931	0.9931	0.9931
Average	0.9927	0.9860	0.9893

4.4. Model Performance of the BOSE-ResNet Architecture

After 30 trials, the Bayesian Optimized Squeeze and Excitation Residual Network (BOSE-ResNet) achieved the highest validation accuracy when the hyperparameters were as follows: a learning rate of 0.0119, Momentum of 0.9122, and L2 regularization of 3.52×10^{-5}. The training accuracy of the model was 98.4375%, and its validation accuracy was 99.4792%.

The model achieved a test accuracy of 99.479%, which was higher than the SE-ResNet's test accuracy. This small discrepancy between the training and test accuracy shows that the model generalized well to unseen data. These results suggest that Bayesian Optimization effectively tuned the hyperparameters, leading to a highly accurate and robust BOSE-ResNet model. Figure 13 and Figure 14 show the training and validation accuracy and loss graphs, respectively, while Figure 15 shows the training and validation dataset confusion matrices of the BOSE-ResNet model. Figure 16 shows the test dataset confusion matrix.

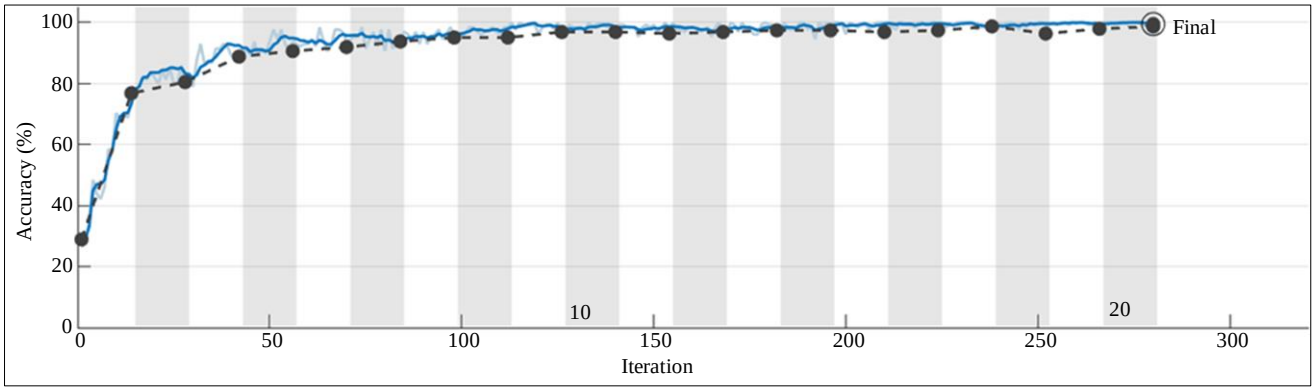


Fig. 13 Training (blue) and validation (black) accuracy graph for the BOSE-ResNet

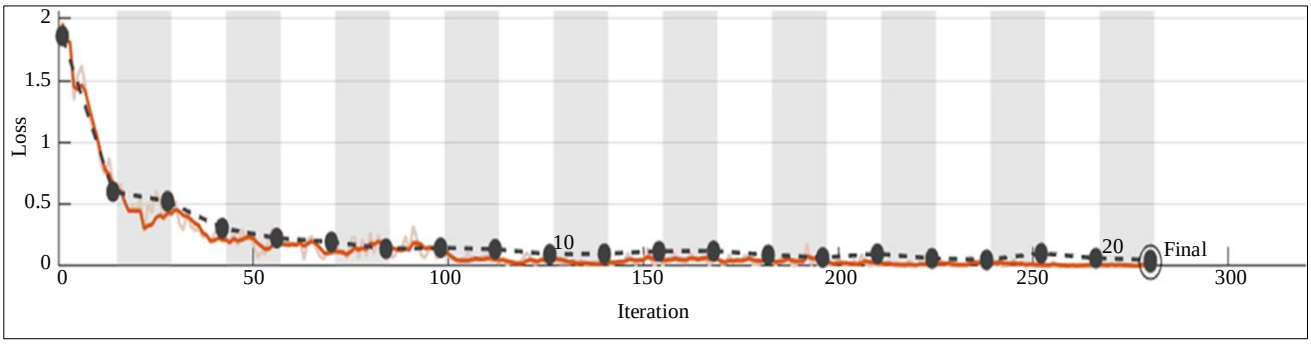
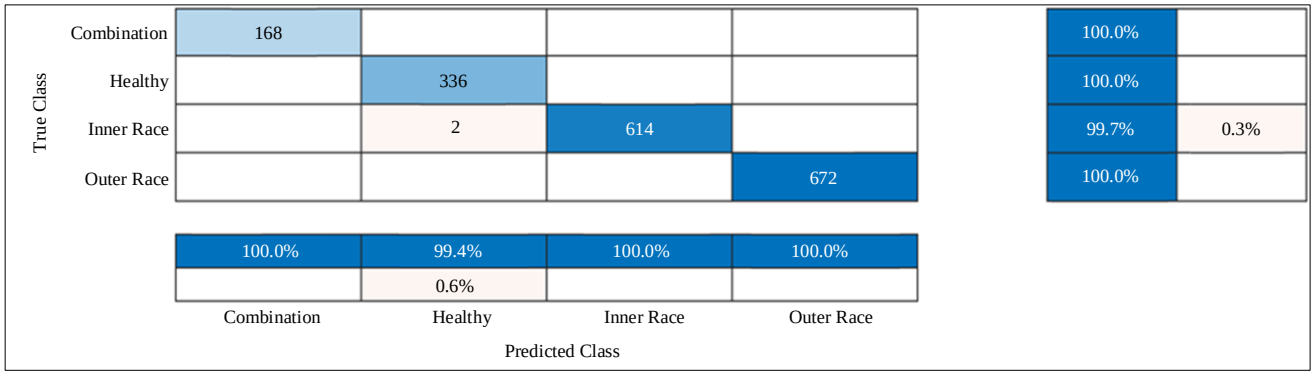
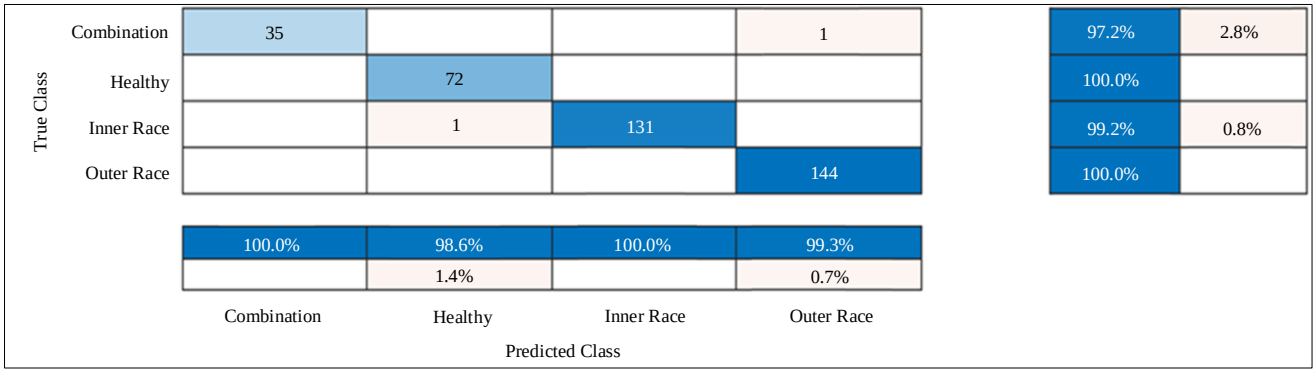


Fig. 14 Training (red) and validation (black) loss graph for the BOSE-ResNet



(a)



(b)

Fig. 15 Confusion matrix for (a) Training, and (b) Validation datasets of the BOSE-ResNet.

Fig. 16 Confusion matrix of testing dataset for the BOSE-ResNet

A detailed analysis of performance metrics indicated high levels of both precision and recall, as shown in Table 5. This indicates that the model can correctly identify most of the bearing faults (high recall) while minimizing false alarms (high precision). Notably, after only 25 minutes and 39 seconds of training, only two healthy samples were incorrectly classified as inner race bearing faults, highlighting the model's robustness in distinguishing between different fault conditions.

Table 5. Performance metrics for BOSE-ResNet

Class	Precision	Recall	F1-score
Combination	1.0000	1.0000	1.0000
Healthy	1.0000	0.9722	0.9859
Inner Race	0.9851	1.0000	0.9925
Outer Race	1.0000	1.0000	1.0000
Average	0.9963	0.9931	0.9946

5. Conclusion

This study provides insight into the impact of architectural modifications and hyperparameter tuning on the ResNet-18 model's performance, guiding further research on enhancing induction motor bearing fault detection. The BOSE-ResNet model, which incorporates squeeze and excitation blocks and additional convolutional layers to its architecture, demonstrated superior performance in terms of accuracy and computational efficiency compared to the original ResNet-18 model. By employing Bayesian Optimization, the hyperparameter space was efficiently explored, and optimal configurations for the SE-ResNet were identified. The BOSE-ResNet architecture showcased exceptional performance with an impressive test accuracy of 99.48%, and a comprehensive evaluation of the model's

performance metrics, such as F1 score, recall, and precision, proved high effectiveness and its ability to classify bearing faults accurately. The accuracy of the BOSE-ResNet in the test phase of the experiment was found to be 20.574% higher than the classical ResNet-18 (78.906%).

The reduction of computational time through structural modifications by 72.32% (from 92 mins 40 sec by the ResNet-18 model to 25 mins 39 sec by the BOSE-ResNet model) further enhanced the practicality and efficiency of the proposed approach. The insights gained from the confusion matrix analysis provided a deeper understanding of the model's classification capabilities, enabling targeted improvements for future iterations.

The high accuracy, improved computational efficiency, and robust performance presented in this study validate the efficacy of the proposed BOSE-ResNet model and pave the way for further advancements in condition monitoring and bearing fault diagnosis using deep learning techniques. With its impressive results, this approach stands as a promising solution for accurate and reliable bearing fault diagnosis, with the potential to enhance the performance and reliability of induction motors in real-world industrial settings.

Future research could explore the application of the BOSE-ResNet model to a wider range of bearing fault types and different bearing datasets. Additionally, investigating the transferability of the optimized hyperparameters to other tasks and hardware platforms would be valuable.

Funding Statement

The Pan African University Institute for Basic Sciences, Technology, and Innovation funded this research.

References

[1] Guilherme Beraldi Lucas et al., "Sensors Applied to Bearing Fault Detection in Three-Phase Induction Motors," *Engineering Proceedings*, vol. 10, no. 1, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] David Gonzalez-Jimenez et al., "Data-Driven Fault Diagnosis for Electric Drives: A Review," *Sensors*, vol. 21, no. 12, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[3] Guoguo Wu et al., "Signal-to-Image: Rolling Bearing Fault Diagnosis Using ResNet Family Deep-Learning Models," *Processes*, vol. 11, no. 5, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[4] Laith Alzubaidi et al., "Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions," *Journal of Big Data*, vol. 8, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] Niloy Sikder et al., "Induction Motor Bearing Fault Classification Using Extreme Learning Machine Based on Power Features," *Arabian Journal for Science and Engineering*, vol. 46, no. 9, pp. 8475-8491, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Dada Saheb Ramteke, Anand Parey, and Ram Bilas Pachori, "A New Automated Classification Framework for Gear Fault Diagnosis Using Fourier-Bessel Domain-Based Empirical Wavelet Transform," *Machines*, vol. 11, no. 12, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Hejun Ye et al., "Bearing Fault Diagnosis Based on Randomized Fisher Discriminant Analysis," *Sensors*, vol. 22, no. 21, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Long Wen, Xinyu Li, and Liang Gao, "A Transfer Convolutional Neural Network for Fault Diagnosis Based on ResNet-50," *Neural Computing and Applications*, vol. 32, no. 10, pp. 6111-6124, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[9] Kaiming He et al., "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp. 770-778, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[10] Yi Liu et al., "A Domain Adaption ResNet Model to Detect Faults in Roller Bearings Using Vibro-Acoustic Data," *Sensors*, vol. 23, no. 6, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Zhenzhong Xu et al., "Hybrid Multimodal Feature Fusion with Multi-Sensor for Bearing Fault Diagnosis," *Sensors*, vol. 24, no. 6, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[12] Shuyi Liu et al., "Fault Diagnosis Strategy Based on BOA-ResNet18 Method for Motor Bearing Signals with Simulated Hydrogen Refueling Station Operating Noise," *Applied Sciences*, vol. 14, no. 1, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[13] Mustafa Musa Jaber et al., "Resnet-Based Deep Learning Multilayer Fault Detection Model-Based Fault Diagnosis," *Multimedia Tools and Applications*, vol. 83, no. 7, pp. 19277-19300, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[14] Jialin Yan, Jiangming Kan, and Haifeng Luo, "Rolling Bearing Fault Diagnosis Based on Markov Transition Field and Residual Network," *Sensors*, vol. 22, no. 10, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Guoxin Sui, and Yong Yu, "Bayesian Contextual Bandits for Hyper Parameter Optimization," *IEEE Access*, vol. 8, pp. 42971-42979, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[16] Modupe Odusami et al., "Analysis of features of Alzheimer's Disease: Detection of Early Stage from Functional Brain Changes in Magnetic Resonance Images Using a Finetuned Resnet18 Network," *Diagnostics*, vol. 11, no. 6, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[17] Miao He, and David He, "Deep Learning Based Approach for Bearing Fault Diagnosis," *IEEE Transactions on Industry Applications*, vol. 53, no. 3, pp. 3057-3065, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[18] Christian Lessmeier et al., "Condition Monitoring of Bearing Damage in Electromechanical Drive Systems by Using Motor Current Signals of Electric Motors: A Benchmark Data Set for Data-Driven Classification," *PHM Society European Conference*, vol. 3, no. 1, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[19] Data Sets and Download, Design and Drive Technology (KAt), Paderborn University, 2024. [Online]. Available: https://mb.uni-paderborn.de/en/kat/research/kat-datacenter/bearing-datacenter/data-sets-and-download

[20] Display Image with Scaled Colors - MATLAB imagesc, MathWorks, 2024. [Online]. Available: https://www.mathworks.com/help/matlab/ref/imagesc.html

[21] Dimas Aryo Anggoro, and Wiwit Supriyanti, "Improving Accuracy Bb Applying Z-Score Normalization in Linear Regression and Polynomial Regression Model for Real Estate Data," *International Journal of Emerging Trends in Engineering Research*, vol. 7, no. 11, pp. 549-555, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[22] Roy Jafari, *Hands-On Data Preprocessing in Python : Learn How to Effectively Prepare Data for Successful Data Analytics*, O'Reilly, Packt Publishing, 2022. [Google Scholar] [Publisher Link]

[23] Charu C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*, Springer Cham, 2nd ed., pp. 1-529, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[24] Sergey Ioffe, and Christian Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *32nd International Conference on Machine Learning (ICML)*, vol. 37, pp. 448-456, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[25] Phil Kim, *MATLAB Deep Learning with Machine Learning, Neural Networks and Artificial Intelligence*, Apress Berkeley, CA, 1st ed., pp. 1-151, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84-90, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[27] Shuiqin Zhou et al., "Application of Convolutional Neural Network in Motor Bearing Fault Diagnosis," *Computational Intelligence and Neuroscience*, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[28] François Chollet, *Deep Learning with Python, Second Edition*, Manning, 2021. [Google Scholar] [Publisher Link]

[29] Fu Zhu et al., "An Improved MobileNet Network with Wavelet Energy and Global Average Pooling for Rotating Machinery Fault Diagnosis," *Sensors*, vol. 22, no. 12, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[30] K.A. Saneera Hemantha Kulathilake et al., "A Review on Deep Learning approaches for Low-Dose Computed Tomography Restoration," *Complex and Intelligent Systems*, vol. 9, no. 3, pp. 2713-2745, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[31] Jeff Heaton, "Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning," *Genetic Programming and Evolvable Machines*, vol. 19, no. 1-2, pp. 305-307, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[32] Meng-Hao Guo et al., "Attention Mechanisms in Computer Vision: A Survey," *Computational Visual Media*, vol. 8, no. 3, pp. 331-368, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[33] Dongwoo Lee et al., "A Study on the Super Resolution Combining Spatial Attention and Channel Attention," *Applied Sciences*, vol. 13, no. 6, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[34] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-Excitation Networks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, pp. 7132-7141, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[35] Yanzhao Wu et al., "Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks," *2019 IEEE International Conference on Big Data (Big Data)*, Los Angeles, CA, USA, pp. 1971-1980, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[36] Zhen Li, Yang Wang, and Jianeng Ma, "Fault Diagnosis of Motor Bearings Based on a Convolutional Long Short-Term Memory Network of Bayesian Optimization," *IEEE Access*, vol. 9, pp. 97546-97556, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[37] Mettu Srinivas et al., *Machine Learning Algorithms and Applications*," Scrivener Publishing LLC, 1st ed., pp. 1-335, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[38] Syifa Auliyah Hasanah et al., "A Deep Learning Review of ResNet Architecture for Lung Disease Identification in CXR Image," *Applied Sciences*, vol. 13, no. 24, 2023. [CrossRef] [Google Scholar] [Publisher Link]