IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 2 May 2024, accepted 27 May 2024, date of publication 31 May 2024, date of current version 7 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3407766

RESEARCH ARTICLE

Optimizing Detection: Compact MobileNet Models for Precise Hall Sensor Fault Identification in BLDC Motor Drives

SEUL KI HONG AND YONGKEUN LEE, (Senior Member, IEEE)
Department of Semiconductor Engineering, Seoul National University of Science and Technology, Seoul 01811, South Korea

Corresponding author: Yongkeun Lee (yklee@seoultech.ac.kr)

This work was supported by Seoul National University of Science and Technology.

ABSTRACT This paper presents a comprehensive study on fault identification in Hall sensors within Brushless Direct Current (BLDC) motor drives using neural networks. Detecting these faults is critical for optimizing motor performance, enhancing energy efficiency, and ensuring overall reliability. Our objective is to propose an accurate, compact, and efficient fault detection neural network model for real-time monitoring and swift responses to Hall sensor faults. Conventional methods are often computationally demanding and fail to detect subtle faults, leading to significant performance decline and potential catastrophic failures. To address this, we leverage MobileNet-based compact models and compare them with state-of-the-art neural network models to identify the most accurate, lightweight, and fast inference option. Our findings demonstrate that MobileNet models excel in detecting Hall sensor faults, achieving over 90% accuracy with significantly fewer parameters (less than five million) and an impressive inference time of under 25 milliseconds. This highlights MobileNet as a robust and efficient choice for Hall sensor fault detection in BLDC motor drives.

INDEX TERMS BLDC motor, hall sensor faults, MobileNet, neural networks.

I. INTRODUCTION

BLDC motors have transformed industrial applications with exceptional efficiency, low maintenance requirements, and precise control capabilities [1], [2], [3]. Their reliable performance relies on accurately detecting the rotor position [4], [5], [6] through signals from Hall sensors. Instantaneous data on the rotor position is crucial for determining precise commutation timing, ensuring optimal motor operation [7], [8]. However, obtaining this data is impossible in the presence of Hall sensor faults. The seamless coordination between rotor position detection and commutation timing is integral to maximizing BLDC motor efficiency.

Prolonged use can lead to a potential issue: displacement of the Hall sensor mount on the stator. This displacement can generate faulty signals, resulting in a lapse of commutation timing. Addressing and rectifying displacements in Hall sensor positioning is crucial for maintaining motor efficiency and reliability over time. Faulty signals from the Hall sensors displacement can lead to improper commutation, causing increased energy consumption, excessive vibration, and reduced motor efficiency [9]. In severe cases, these inaccuracies can escalate to complete motor failure, posing a risk of damage to the motor and its components.

Traditional hall sensor fault diagnosis methods [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22] for often involve intricate signal processing techniques and extensive analysis. However, these approaches can be computationally intensive, lack real-time efficiency, and may struggle to pinpoint subtle displacements effectively. The methods and their limitations are outlined below:

1) Magnetic Field Strength Measurement: This method may be affected by external magnetic fields, leading to inaccuracies. Interference from nearby magnetic sources can compromise the precision of the displacement measurement.

2) Hall Voltage Measurement: Hall voltage measurements can be affected by temperature variations. Changes in temperature can impact the characteristics of the

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenbao Liu.

Hall sensor, leading to inaccuracies in displacement measurements.

3) Angle Measurement using Hall Sensors: This method is sensitive to the alignment of the sensor and the magnet, and mechanical misalignments can introduce errors. Additionally, the resolution may be limited when converting angular measurements to linear displacement.

4) Magnetic Encoder Systems: These systems can be susceptible to electromagnetic interference, and the accuracy may degrade in harsh industrial environments with high levels of electrical noise. Traditional Hall sensor fault diagnosis methods may have limitations in terms of resolution and sensitivity, especially when high precision is required for specific applications. To maintain accuracy, traditional Hall sensor fault diagnosis systems may require periodic calibration due to factors such as component aging, temperature variations, and external influences. These traditional methods may involve complex circuitry or additional components to compensate for limitations, which can increase the overall cost and complexity of the system [13], [14], [19], [20], [21], [22].

To address the challenges and limitations inherent in motor fault diagnosis, researchers have turned to deep learning approaches [23], [24], [25], [26], [27]. Fault diagnosis in motors essentially boils down to a classification problem, where the goal is to distinguish between different types of faults. Deep learning, revered as a transformative force across various domains, owes its prominence to its unparalleled ability to tackle intricate tasks with precision and efficiency. Its rapid adoption in diverse fields, including motor fault diagnosis, underscores its capacity to delve into complex patterns within data. By automatically learning hierarchical representations, deep learning excels at feats like anomaly detection, marking a significant leap in understanding and interacting with complex data. This pervasive embrace emphasizes deep learning's pivotal role in expanding the horizons of what machines can achieve, particularly in critical areas like motor fault diagnosis, where precision and reliability are paramount. Initially, motor fault detection and diagnosis heavily relied on single, fully customized network models such as Convolutional Neural Networks (CNNs), Recursive Neural Networks (RNNs), or Long Short-Term Memory networks (LSTMs) [23], [24], [26]. CNNs demonstrated exceptional proficiency in extracting spatial information, rendering them ideal for scrutinizing visual image data to uncover motor faults effectively. Conversely, LSTMs proved adept at capturing intricate temporal dependencies inherent in sequential data, thereby proving invaluable for analyzing time-series data emanating from sensors monitoring motor performance. This tailored approach adeptly capitalized on the unique strengths of each network architecture, enabling the precise targeting of specific aspects of motor fault detection and diagnosis. Consequently, it laid a robust foundation for the development of more sophisticated techniques in this pivotal domain, ensuring enhanced accuracy and efficiency in identifying and addressing motor-related anomalies.

However, the field later transitioned towards hybrid models such as the CNN-LSTM model [25], [27], which have demonstrated superior capabilities in learning and extracting features from complex and extensive datasets. This shift was prompted by the challenges posed by the escalating complexity and scale of data for single-model neural networks. These hybrid neural network models have showcased promising results in fault detection for components like Hall sensors, achieving accuracies surpassing 98% [27]. Nevertheless, many previous studies have primarily focused on enhancing fault detection accuracy, often overlooking other critical performance metrics such as latency and model size, thus lacking comprehensive evaluation. Furthermore, these approaches typically necessitate substantial computational resources to manage the processing and analysis of large datasets throughout the training, validation, and testing phases of the network models. In our study, the aim is to develop an efficient model that can operate on mobile platforms, utilizing only minimal edge computing power. This necessitates a unique approach, focusing on three crucial performance metrics that must be met for mobile operation: maximum accuracy, minimum latency, and minimum inference time. To achieve this, we utilize a MobileNet-based neural network [28], [29], tailored specifically for mobile applications. We conduct a thorough evaluation of accuracy, latency, and inference time, targeting real-time applications for detecting Hall sensor displacement.

MobileNet was selected due to its efficiency in catering to mobile and embedded vision applications. It adopts a streamlined architecture leveraging depthwise separable convolutions, facilitating the construction of lightweight deep neural networks. Additionally, for practical implementation on mobile devices, we provide guidelines: 1) minimize the model size to less than 5 million parameters, while 2) maintaining accuracy above 90%, and 3) achieving an inference time of less than 50 milliseconds. These guidelines ensure efficient utilization on resource-constrained devices like smartphones and IoT devices. We also compare the performance with state-of-the-art neural networks, including VGG16, ResNet50, DenseNet121, InceptionV3, Xception, and EfficientNetB0 [29], [30], [31], [32], [33], [34], [35].

Ensuring that the Hall sensor displacement detection model is executable on mobile devices is crucial for real-time monitoring and troubleshooting in industrial settings. Mobile-executable models enable immediate detection of sensor errors without relying on external computing infrastructure, facilitating quick on-site response. Timely detection of displacement errors minimizes equipment damage and production downtime, as operators can take corrective actions promptly. Additionally, technicians can diagnose and address sensor issues more efficiently with the detection model readily available on their mobile devices, reducing service time. Keeping sensor data localized on mobile devices enhances privacy and security, mitigating risks associated with data transmission to external servers. Executability on mobile devices ensures continuous monitoring and diagnosis, even

in environments with limited internet connectivity, enhancing operational resilience. Finally, optimizing the model for mobile executability minimizes computational overhead and energy consumption, extending device battery life and enhancing operational efficiency in industrial environments.

As a final note, the primary focus of this study lies in the development and optimization of neural networks customized for the detection of errors related to Hall sensor displacement. It is noteworthy that our scope does not encompass error compensation and correction.

II. METHODOLOGY

A. DATA COLLECTION AND MOTOR DRIVE

The data collection process for this study comprises real-world experiments and MATLAB simulations carried out on BLDC motor drives under diverse operating conditions, ensuring the acquisition of a comprehensive and representative dataset. However, it is noteworthy that the dataset utilized for training and testing purposes was obtained through MATLAB simulations, following validation to ensure alignment with experimental results, chosen for ease of operation and convenience.

Figure 1a illustrates the system setup for BLDC motor control, comprising a motor controller, commutation logic, a mobile platform for network models, and three half-bridge circuits (three phase inverter), which are responsible for connecting the motor phases to either VCC or GND. This configuration enables the injection of current into the coils, generating the necessary magnetic fields for the different phases. A motor controller governs commutation, facilitated by the motor driver, which allows communication between the control block and the half-bridge circuits. This setup enables precise control of commutation through the circuits based on feedback from Hall position sensors, which provide information on the rotor's position to the motor controller.

To implement Hall sensor displacement or fault signals, we refined the commutation timing within the motor drive and control programming. Rather than physically adjusting the Hall sensor's position, we opted for a software-based approach.

Here's how it works: After reading the hall sensor state in the main loop, a delay function is invoked to introduce the necessary delay before commutating the motor. Within this delay function, we reset a timer/counter (TCNxT) and clear its compare match flag (OCFxA). This ensures that the program waits until the TCNxT compare match flag is set, indicating that the desired delay has been reached. The commutation function is embedded within the interrupt service routine (ISR) for the timer/counter (TCNxT). This arrangement guarantees that the delay is executed before commutating the motor.

By employing this methodology, we achieve precise control over the commutation timing without the need for physical adjustments to the Hall sensor position. Detailed specifications of the motor drive and BLDC motor are presented in Table 1.

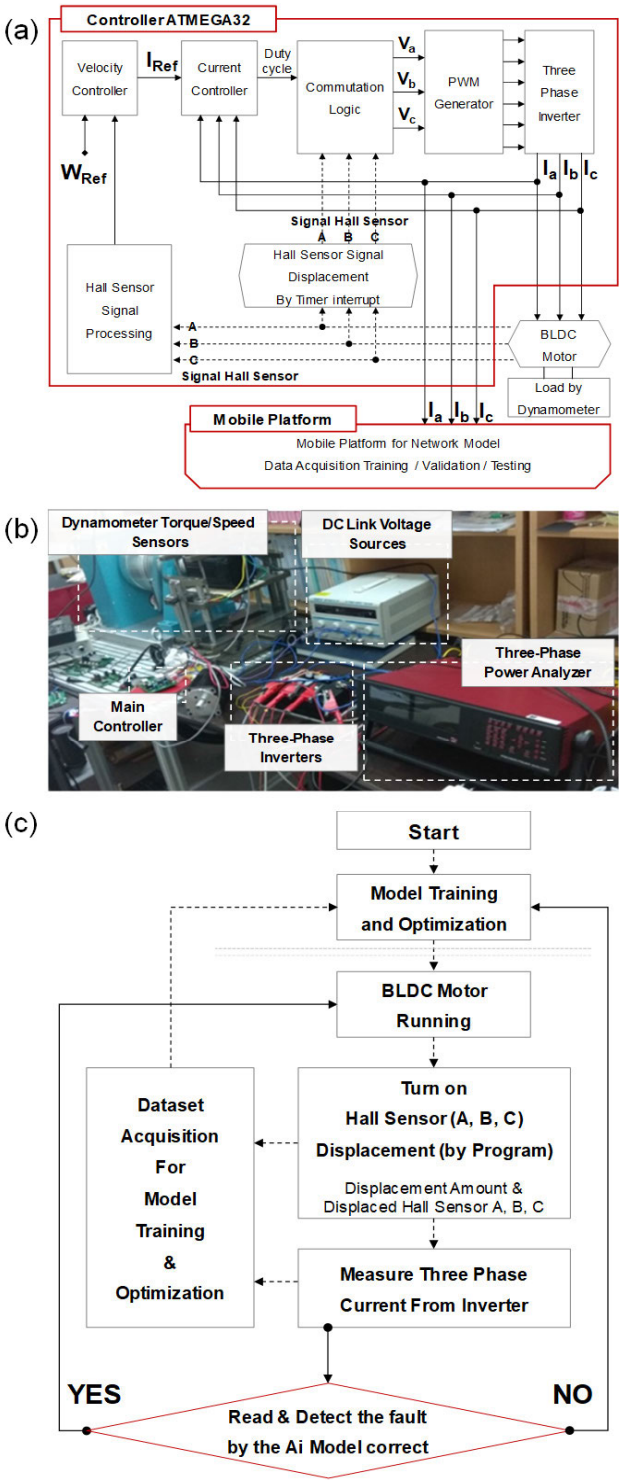


FIGURE 1. (a) A detailed description of the setup, (b) A photo for the experimental setup, and (c) An overarching flowchart depicting the diagnostic process.

In Figures 1b and 1c, the experimental setup for driving the three-phase BLDC motor is showcased, along with an overarching flowchart depicting the diagnostic process. This setup utilizes a Hysteresis Brake Dynamometer (BHD-144) by Validmagnetics, alongside a microcontroller for torque and rotor speed control. Additionally, a power analyzer

TABLE 1. The motor drive and BLDC motor specification.

Motor Drive specification	
Controller	Atmega32 (8-bit uC)
System Clock	16MHz
PWM Frequency	20KHz
On-duty Cycle	5-90%
Motor speed range	200~2000 rpm
Load torque	2~5 N-m
DC Voltage	400 V
MOSFET On-Resistance	0.1 ohms
Commutation Scheme	six-step
BLDC motor specification	
Switching frequency	20KHz
Pole pairs of BLDC motor	4
Inertia	0.0027 J
Stator phase resistance	0.0485 ohms
Stator phase inductance	0.0085 H
# of stator slot	24

TABLE 2. Six-step commutation switching patterns of the BLDC motor drive. Note that "ON+" denotes upper MOSFET On and lower MOSFET OFF, "ON−" indicates upper MOSFET OFF and lower MOSFET ON, and "OFF" signifies both upper and lower MOSFETs are turned off.

	0°~ 60°	60°~ 120°	120°~ 180°	180°~ 240°	240°~ 300°	300°~ 360°
Hall_A	1	1	1	0	0	0
Hall_B	0	0	1	1	1	0
Hall_C	1	0	0	0	1	1
A phase	OFF	ON+	ON+	OFF	ON-	ON-
B phase	ON-	ON-	OFF	ON+	ON+	OFF
C phase	ON+	OFF	ON-	ON-	OFF	ON+

(PPA-3500) by Newtons4th was employed for phase current and voltage measurement. Experimental results served solely for validation against MATLAB simulations, ensuring consistency between the two sets of data.

The dataset for training and testing comprises three-phase currents over time, chosen because they reflect Hall sensor displacement. Commutation timing for three-phase currents is determined by the combination of three Hall sensors, as outlined in Table 2. This choice is reinforced by the fact that three-phase currents are already utilized for motor speed and torque control, incurring no additional cost.

The dataset includes both normal and flawed three-phase currents, generated by introducing proper and improper commutation timing due to Hall sensor displacement at four distinct levels: zero seconds (zero PWM Pulse period displacement), 0.0001 seconds (2 PWM pulse period displacement), 0.005 seconds (100 PWM pulse period displacement), and 0.01 seconds (200 PWM pulse period displacement) in A, B, and C Hall sensors. In this study, the PWM pulse frequency is 20KHz. The speed/torque control of BLDC motor is achieved by varying the duty cycles of PWM Pulse from

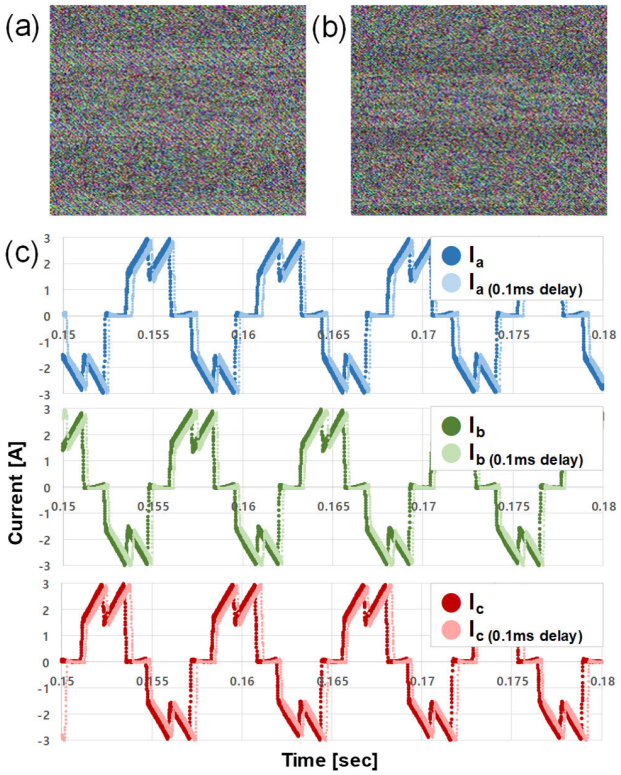


FIGURE 2. (a) Three phase current waveform in RGB without displacement in Hall sensor (b) Three phase current waveform in RGB with 0.1ms displacement in Hall sensor. (c) Three phase current waveforms with zero delay and 0.1ms displacement.

the microcontroller. This dataset is prepared to assess the detection capability at different displacement levels in Hall sensors. The BLDC motor is driven by a six-step commutation scheme [8], a common and widely used method for BLDC motor control. PWM signals drive the upper MOSFETs of phases A, B, and C to regulate the voltage across the motor windings and control motor speed. Meanwhile, constant turn on and off signals drive the lower MOSFETs to ensure proper commutation of the motor phases.

By meticulously designing the data collection to cover a broad range of motor speeds, torque levels, and commutation conditions, we aim to create a comprehensive and diverse dataset reflecting the complex dynamics of BLDC motor drives. This enriched dataset will facilitate the training and evaluation of deep learning models, enabling us to draw meaningful conclusions about Hall sensor displacement under various operating conditions.

B. DATA PREPROCESSING

The measured three phase current undergo a meticulous preprocessing phase to create suitable input data for the neural network models. Before feeding the data into the models, the phase current are normalized to ensure uniformity and to prevent any model biases towards specific features. Normalization scales the data to a common range, typically between 0 and 1, making it easier for the models to converge during training. The dataset is divided into training and testing sets

TABLE 3. MobileNet architecture where dw and FC stands for depth-wise convolution and fully connected layer.

	Input Size	Operator	Filter Size	Number of Filter	Activation function	Stride
MobileNet	224^2x3	Conv	3x3x3	32	ReLu	2x2
	112^2x32	Conv-dw	3x3	32	ReLu	1x1
	112^2x32	Conv	1x1x32	64	ReLu	1x1
	112^2x64	Conv-dw	3x3	64	ReLu	2x2
	$56^2\times$ 64	Conv	1x1x64	128	ReLu	1x1
	$56^2\times$ 128	Conv-dw	3x3	128	ReLu	1x1
	$56^2\times$ 128	Conv	1x1x128	128	ReLu	1x1
	$56^2\times$ 128	Conv-dw	3x3	128	ReLu	2x2
	28^2x128	Conv	1x1x128	256	ReLu	1x1
	28^2x256	Conv-dw	3x3	256	ReLu	1x1
	28^2x256	Conv	1x1x256	256	ReLu	1x1
	28^2x256	Conv-dw	3x3	256	ReLu	2x2
	14^2x256	Conv	1x1x256	512	ReLu	1x1
	5(14^2x512)	Conv-dw-Expansion	3x3	512	ReLu	1x1
	5(14^2x512)	Conv-Expansion	1x1x512	512	ReLu	1x1
	14^2x512	Conv-dw	3x3	512	ReLu	2x2
	7^2x512	Conv	1x1x512	1024	ReLu	1x1
	7^2x1024	Conv-dw	3x3	1024	ReLu	2x2
	7^2x1024	Conv	1x1x1024	1024	ReLu	1x1
	7^2x1024	Avg-Pool	7x7		ReLu	1x1
	1^2x1024	FC	1024x1000			1x1
	1^2x1000	Softmax	Classifier			1x1

using a 4:1 ratio. The training set, representing 80% of the data, is further split into training and validation sets with a 3:1 ratio (60% for training, 20% for validation).

The original data is structured as a matrix of size 14700 × 3 as depicted in Figure 2c, with each row representing phase current at 14700 different sampling times for each phase. To facilitate effective spatial feature extraction, the data is then transformed into RGB images of size 224 × 224×3 as illustrated in Figures 2a and 2b, resulting in a three-dimensional input shape, exploiting its ability to detect intricate spatial patterns that may reveal valuable information about Hall sensor inaccuracies. To enhance to learn spatial patterns effectively, data augmentation techniques are applied. Augmentation involves random transformations, such as rotations, flips, and shifts, to the RGB images. This process creates additional training samples, increasing the model's exposure to diverse patterns and improving its robustness [36]. For the case where there might be an imbalance in the data distribution, balancing techniques are applied to ensure equal representation of different classes (e.g., normal and faulty Hall sensor readings). This balanced distribution prevents the models from being biased towards the majority class and ensures fair evaluation of their accuracy in detecting Hall sensor inaccuracies.

By preprocessing the data through normalization, splitting, augmentation, sequence formation, padding, and dataset balancing, we create well-structured and representative input data [36], [37]. These carefully prepared datasets, tailored to each model's requirements, will empower the deep learning models to effectively learn from the data, improving their performance in detecting faulty Hall sensors in BLDC motor drives.

C. MOBILENETV1 AND V2 FEATURES AND ARCHITECTURE

MobileNet is a convolutional neural network (CNN) architecture designed for efficient use on mobile devices with limited computational resources. Key features of the MobileNet architecture [28], [29] as shown in Table 3 include:

1) Depthwise Separable Convolution [28], [29], [33]: It consists of two steps: depthwise convolution and pointwise convolution. Depthwise convolution applies a single filter per input channel, resulting in a set of feature maps. Pointwise convolution performs a 1 × 1 convolution to combine the output of depthwise convolution across channels as shown in Fig. 3.

2) Parameter Reduction: MobileNet reduces the number of parameters by using 1 × 1 convolutions (pointwise convolutions) to decrease the dimensionality of the data before and after the depthwise separable convolution. This helps in reducing the computational cost while retaining the representational capacity of the network.

3) Width Multiplier: MobileNet introduces a hyperparameter called the "width multiplier" (denoted as α) that scales the number of channels in each layer. By adjusting the width multiplier, developers can trade-off between the model size and its accuracy, making it adaptable to various resource constraints.

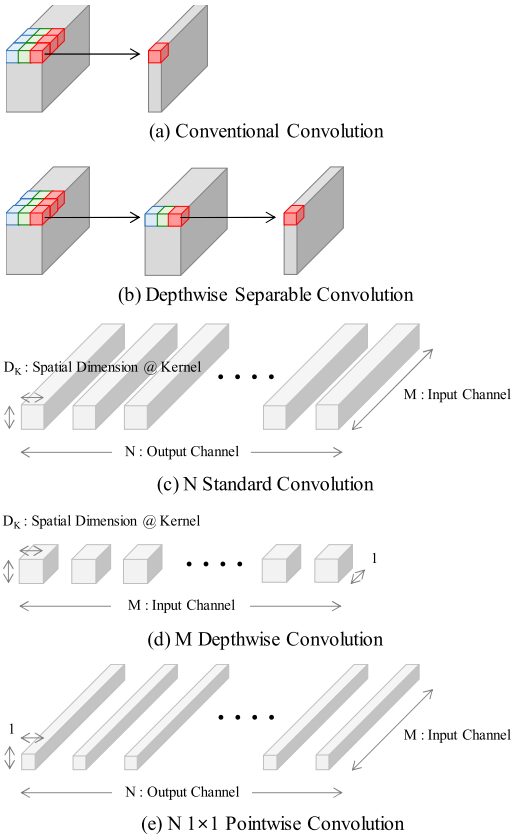


FIGURE 3. (a) Conventional Convolution, (b) Depthwise separable Convolution, (c) N-Standard, (d) M Depthwise, and (e) N 1 × 1 Pointwise Convolution neural network filters.

4) Resolution: Another hyperparameter in MobileNet is the input resolution (denoted as ρ). It allows for adjusting the resolution of the input images, providing further flexibility in balancing accuracy and computational cost.

5) Inverted Residuals with Linear Bottlenecks [29]: MobileNetV2, an improvement over the original MobileNetV1, introduces inverted residuals with linear bottlenecks. The inverted residual structure helps in capturing and propagating information through the network more effectively.

The overall design of MobileNet is focused on providing efficient yet accurate feature extraction suitable for mobile vision applications. MobileNetV2 builds upon the success of MobileNetV1 by introducing improvements in architecture design, including inverted residuals and linear bottleneck layers, to achieve higher accuracy and efficiency for deep learning inference on mobile and edge devices.

The MobileNet architecture is defined in Table 3. All layers are followed by a batchnorm [36] and ReLU nonlinearity with the exception of the final fully connected layer which has no nonlinearity and feeds into a softmax layer for classification. Down sampling is handled with strided convolution in the depthwise convolutions as well as in the first layer. A final average pooling reduces the spatial resolution to 1 before the fully connected layer. Counting depthwise and pointwise convolutions as separate layers, MobileNet has 28 layers.

TABLE 4. Comparing the number of parameters across different types of convolutional layers.

Convolutional Layer Used for one expansion	Number of Parameters
Traditional Convolutional Layers	17,920
Depth wise Separable Convolution Layers	2,860
Linear Bottleneck with Depth wise Separable Convolution Layers	2,447
Inverted Residual with Depth wise Separable Convolution Layers	2,447

Above, we present a comparison of the number of parameters for four different convolutional layers, all with the same hyperparameters in the Table 4: 3 input channels, 64 output channels, 3 × 3 kernel size, one expansion for linear bottleneck and inverted residual structures and a total of 10 layers. As illustrated in the Table 4, adopting these efficient convolutional layers as building blocks significantly reduces the overall number of parameters.

D. TRANSFER LERANING [39] AND MODEL OPTIMIZATION

The MobileNetV1 and V2, as well as other state-of-the-art models for comparison, were modified for our specific classification task using transfer learning, leveraging pretraining on the extensive and diverse ImageNet dataset [38]. This involved tailoring the ImageNet pre-trained base models with our specific additional top layers to seamlessly integrate with the features learned by the base model. Adjustments to hyperparameters, including the number of units in the dense layer, allowed for flexibility in fine-tuning the model's performance were done.

In our study, we extensively utilized transfer learning techniques to adapt pretrained models for our specific classification task of detecting displacement of Hall sensor in BLDC motor drives. Firstly, we employed fine-tuning of pretrained models, including MobileNetV1, MobileNetV2, and other state-of-the-art architectures. This process involved unfreezing some or all of the layers of the pretrained models and retraining them on our dataset. By doing so, the models were able to adapt their learned representations to our task, enhancing their performance.

Additionally, we leveraged feature extraction from the pretrained base models, particularly those trained on the ImageNet dataset. This involved utilizing the GlobalAveragePooling2D layer after the base model to extract essential features from the entire image. These features were then fed into additional top layers for classification, allowing us to utilize the learned representations without modifying the pretrained model's weights.

Furthermore, we performed architecture modification to tailor the pretrained models to our task. This included adding specific additional top layers, such as a fully connected layer with 1024 units and ReLU activation function, to enhance the model's capacity to capture intricate patterns relevant to our classification task.

TABLE 5. Optimized hyperparameters.

Hyperparameter	Values
Batch Size	32
Epoch #	10
Learning Rate	0.0001
Optimizer	Adams
Loss function	categorical_crossentropy
Activation function	Softmax

Lastly, hyperparameter tuning played a crucial role in optimizing the performance of the pretrained models. We fine-tuned hyperparameters as shown in Table 5, such as the number of units in the dense layer, learning rate, batch size, and regularization techniques, based on the characteristics of our dataset and the specific requirements of our classification task. Through these comprehensive transfer learning techniques, we ensured the effectiveness and suitability of the pretrained models for our task while achieving optimal performance and generalization ability.

The modified models underwent extensive training, fine-tuning, optimization, and evaluation using the preprocessed dataset to detect displacement of Hall sensor in BLDC motor drives. Throughout the training process, the models' performance was continuously monitored using appropriate loss functions and accuracy metrics. The loss function quantified the disparity between predicted output and actual ground-truth labels, while the accuracy metric measured correct predictions. By optimizing these metrics, our goal was to achieve the highest possible accuracy in detecting displacement of Hall sensor while maintaining a balance between the models' training performance and generalization ability. In the paper, we opted to denote them by their respective pre-trained model names for the sake of simplification, even though all models listed in the table have undergone transfer learning.

To prevent overfitting [20] and enhance generalization [21], [22], the models were optimized with a batch size of 32, a learning rate of 0.0001, and limited to 15 epochs. A small batch size ensured more frequent updates to the model's parameters, while a lower learning rate helped fine-tune the models' weights gradually. Limiting the number of epochs prevented the models from memorizing the training data, encouraging them to focus on learning essential features that could generalize to unseen data.

The training, validation, and testing of our machine learning model are conducted on a processor with specifications tailored for mobile devices: an x64-based Intel(R) Core(TM) i7-10510U mobile CPU @ 1.80GHz, complemented by 16.0GB of RAM, and running on a 64-bit operating system. This deliberate choice ensures a comprehensive evaluation of the machine learning model's performance under conditions representative of typical usage on mobile platforms.

A mobile processor, such as the i7-10510U mobile, is meticulously crafted for portable devices, emphasizing ultra-low power consumption and sleek form factors, making it an ideal fit for thin laptops, tablets and smartphones. The 'U' in i7-10510U mobile signifies "ultra-low power," indicating its suitability for ultra-thin notebooks and similar style laptops or tablets where battery life takes precedence over raw performance. With a typical thermal design power (TDP) of around 15W, its efficiency is paramount, though it may compromise on speed.

While we've achieved successful deployment of MobileNet on the Android platform (e.g., Samsung Galaxy S with Snapdragon® 8 Gen 2), via conversion from Keras-based models to TFLite models, not all models exhibit seamless deployment. Our study doesn't singularly focus on porting networks to Android or Apple iOS platforms; rather, we emphasize demonstrating their compatibility with mobile-based CPUs in general. To benchmark the evaluated networks, we opt for the i7-10510U mobile CPU due to its comprehensive support for executing all considered networks effectively.

III. EXPERIMENTAL RESULTS

This study employs a comprehensive set of performance metrics, including precision, recall, F1 score, and accuracy, to meticulously evaluate the delicate balance between false positives and false negatives, providing a nuanced assessment of the predictive prowess of the model. Precision, embodying the accuracy of positive predictions, is computed as the ratio of True Positives to the sum of True Positives and False Positives. In parallel, recall, signifying the model's proficiency in capturing all positive instances, is derived from the ratio of True Positives to the sum of True Positives and False Negatives. The F1 score, functioning as the harmonic mean of precision and recall, offers a holistic measure that effectively considers both false positives and false negatives, calculated as 2*(Precision * Recall) / (Precision + Recall). Lastly, accuracy, a fundamental metric, signifies the proportion of correctly predicted instances (comprising true positives and true negatives) out of the total instances in the dataset.

In addition, the study recognizes the significance of measuring inference time, synonymous with prediction time, as a critical aspect for evaluating a machine learning model's real-world performance. To achieve this, timing measurements are seamlessly integrated directly into the code, ensuring a straightforward approach to measuring inference time. However, it is acknowledged that this method may introduce some overhead due to the profiling code.

The Efficiency Score (ES) introduced in this study serves as a significant metric to holistically evaluate model performance by considering accuracy, latency, and model simplicity. The ES formula, denoted as

$$ES = W\_A \times N\_A + W\_L \times N\_L + W\_P \times N\_P \quad (1)$$

encapsulates a balanced consideration of these factors. Here, W_A, W_L, and W_P denote weight factors for accuracy, latency, and parameters, respectively, while N_A, N_L, and N_P represent normalized values of accuracy, latency, and parameters, respectively.

In certain scenarios, the weight factors may be adjusted to prioritize specific aspects of model performance. For instance, if minimizing inference time is paramount, a higher weight can be assigned to latency (W_L), resulting in greater emphasis on reducing latency in the ES calculation. Conversely, if maximizing accuracy is the primary concern, a higher weight may be assigned to accuracy (W_A), leading to increased importance placed on achieving high accuracy levels.

Each factor is assigned an equal weight of 1/3 in this study as a starting point, ensuring equal importance is given to accuracy, latency, and model simplicity. However, individual may adjust these weight factors based on the specific requirements of their application or domain. For example, in safety-critical applications where accuracy is of utmost importance, a higher weight may be assigned to accuracy to ensure robust performance.

The construction of the ES formula enables a unified metric for evaluating overall model efficiency, as it appropriately weighs the individual contributions of accuracy, latency, and parameters. Normalized values of accuracy (N_A), latency (N_L), and parameters (N_P) ensure a balanced consideration of these critical aspects in evaluating model efficiency. By normalizing these values, the ES formula allows for fair comparisons across models with varying performance metrics.

In practical terms, the ES provides individuals with a comprehensive measure of model efficiency, taking into account both predictive accuracy and computational efficiency. A higher ES indicates better overall performance, considering the trade-offs between accuracy, latency, and model simplicity. This meticulous consideration underscores the commitment to obtaining accurate insights into the model's efficiency during prediction tasks, facilitating informed decision-making in model selection and deployment. Adjusting the weight factors allows for flexibility in prioritizing specific aspects of model performance according to the requirements of the application or domain.

The evaluation of various network architectures for the 0.0001-second displacement of a Hall sensor yielded diverse performance outcomes across accuracy, latency, and model parameters as shown in Table 6. DenseNet121 exhibited the highest accuracy among the evaluated models, achieving an impressive 97%. This underscores its proficiency in accurately classifying sensor displacements at the specified time interval. MobileNetV1 and InceptionV3 also demonstrated notable accuracy, registering at 94%.

MobileNetV1 emerged as the leader in latency performance, boasting an inference time of 0.0201 seconds. EfficientNetB0 followed closely with a latency of 0.0391 seconds. On the contrary, VGG 16 exhibited the highest latency among the models, requiring 0.1956 seconds for inference.

When considering model complexity in terms of parameters, MobileNetV2 showcased efficiency with the lowest parameter count at 3.57 million. EfficientNetB0 also demonstrated simplicity with 5.36 million parameters. In contrast, InceptionV3, Xception and ResNet50 presented higher complexity with parameter counts of 23.9million, 22.9million and 25.7 million, respectively.

TABLE 6. Analysis of detection accuracy, latency, and model parameters for 0.0001-second displacement of Hall sensor using various network architectures.

Network	Accuracy	Latency, Inference Time (sec)	Params (Millions)
MobileNetV1	0.94	0.0201 sec	4.28 M
MobileNetV2	0.91	0.0250 sec	3.57 M
VGG 16	0.92	0.1956 sec	15.2 M
ResNet50	0.62	0.0673 sec	25.7 M
DenseNet121	0.97	0.0752 sec	8.09 M
InceptionV3	0.94	0.0462 sec	23.9 M
Xception	0.83	0.0713 sec	22.9 M
EfficientNetB0	0.48	0.0391 sec	5.36 M

Table 7 presents the precision, recall, and F1 scores for various network architectures when detecting a 0.0001-second displacement of a Hall sensor. Precision represents the accuracy of positive predictions, Recall measures the model's ability to capture positive instances, and F1 is the harmonic mean of precision and recall. Notably, DenseNet121 exhibits high precision, recall, and F1 scores, indicating superior classification performance. MobileNetV1, MobileNetV2, VGG 16, InceptionV3, and Xception also demonstrate commendable scores. In contrast, ResNet50 and EfficientNetB0 show comparatively lower scores, suggesting challenges in precision and recall.

The efficiency scores as shown in Table 8, calculated as a composite metric considering accuracy, latency, and model complexity, provide a holistic perspective on the performance of various network architectures for the 0.0001-second displacement of a Hall sensor. MobileNetV1 and MobileNetV2 emerged as top performers with efficiency scores of 0.97 and 0.95, respectively. These models showcase a favorable trade-off between accuracy, latency, and model complexity, making them highly efficient choices for applications requiring optimal overall performance.

While DenseNet121 exhibited high accuracy (97%), its efficiency score of 0.83 suggests a slightly higher trade-off between accuracy, latency, and model complexity compared to MobileNetV1 and MobileNetV2. VGG 16, ResNet50, InceptionV3, EfficientNetB0 and Xception displayed lower efficiency scores, indicating that these models may not provide the same level of overall performance when considering the three key aspects.

The efficiency scores as shown in Table 8, calculated as a composite metric considering accuracy, latency, and model complexity, provide a holistic perspective on the performance of various network architectures for the

IV. DISCUSSION

In Table 6, the comparative analysis of various network architectures for a 0.0001-second displacement of a Hall sensor

TABLE 7. Analysis of detection accuracy including Precision, Recall and F1 for 0.0001-second displacement of Hall sensor using various network architectures. m and w stand for macro average and weighted average.

Network	Type	Precision	Recall	F1
MobileNetV1	m	0.95	0.95	0.94
	w	0.9	0.94	0.94
MobileNetV2	m	0.91	0.91	0.91
	w	0.91	0.91	0.91
VGG 16	m	0.93	0.93	0.92
	w	0.93	0.92	0.92
ResNet50	m	0.62	0.61	0.60
	w	0.62	0.62	0.61
DenseNet121	m	0.97	0.97	0.97
	w	0.97	0.97	0.97
InceptionV3	m	0.94	0.94	0.94
	w	0.94	0.94	0.94
Xception	m	0.83	0.83	0.83
	w	0.83	0.83	0.83
EfficientNetB0	m	0.24	0.50	0.33
	w	0.23	0.48	0.41

TABLE 8. Analysis of Efficiency Score for 0.0001-second displacement of Hall sensor using various network architectures. The N_A, N_L, and N_P represent the normalized values of accuracy, latency, and parameters.

Network	N_A	N_L	N_P	Efficiency Score
MobileNetV1	0.94	1.00	0.97	0.97
MobileNetV2	0.88	0.97	1.00	0.95
VGG 16	0.90	0.00	0.47	0.46
ResNet50	0.29	0.73	0.00	0.34
DenseNet121	1.00	0.69	0.80	0.83
InceptionV3	0.94	0.85	0.08	0.62
Xception	0.71	0.71	0.13	0.52
EfficientNetB0	0.00	0.89	0.92	0.60

reveals intriguing insights into their performance metrics. MobileNetV1 and MobileNetV2 emerge as models striking a commendable balance between accuracy, latency, and model simplicity. DenseNet121 stands out for its exceptional accuracy and classification metrics, albeit with a slightly higher latency. On the other hand, VGG 16 and InceptionV3 achieve high accuracy at the expense of increased latency and a larger number of parameters. EfficientNetB0, while boasting low latency, sacrifices accuracy and exhibits less robust classification metrics.

Examining Table 7, which delves into precision, recall, and F1 scores for the 0.0001-second displacement scenario, reinforces the prowess of DenseNet121. It consistently achieves the highest precision, recall, and F1 scores across both macro and weighted averages. MobileNetV1, MobileNetV2, VGG 16, InceptionV3, and Xception consistently deliver commendable precision, recall, and F1 scores. However, ResNet50 lags behind with lower precision, recall, and F1 scores, indicative of comparatively weaker classification performance. EfficientNetB0 faces challenges, reflected in its lower precision, recall, and F1 scores.

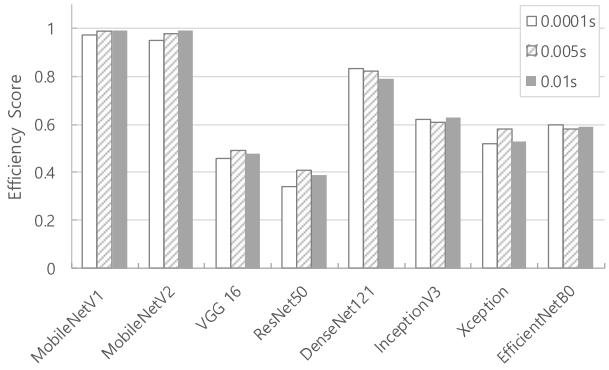


FIGURE 4. The efficiency score variation in relation to displacement for various network architectures".

To provide a holistic perspective on model performance, a composite metric, the efficiency score, was introduced in Table 8. this score emphasizes models achieving higher accuracy with lower latency and fewer parameters. MobileNetV1 and MobileNetV2 lead the efficiency scores, emphasizing a favorable trade-off between accuracy, latency, and model complexity.

Table 9 extends the analysis by exploring detection accuracy, precision, recall, F1 score, latency and efficiency as a function of displacement intervals (0.0001, 0.005, and 0.01 seconds) for each network architecture. Fig. 4 illustrates the results of this analysis. ‘‘MobileNetV1 and MobileNetV2 showcase consistent high accuracy, precision, recall, and F1 scores with relatively low latency, and their accuracy improves with increasing displacement. Most network models exhibit an improvement in accuracy, precision, recall, and F1 scores with a slight decrease in latency as the displacement interval increases. Notably, MobileNetV1 and MobileNetV2 exhibit efficiency gains with greater displacement intervals, while other networks appear unaffected by changes in the displacement interval.

Training, validation, and testing of all models were conducted on a processor platform tailored for mobile applications, rather than relying on GPU acceleration.: the x64-based Intel(R) Core(TM) i7-10510U mobile CPU @ 1.80GHz, supported by 16.0GB of RAM. The performance scores presented in the tables of the experimental results can be confidently affirmed and reproduced, provided the models are run on hardware with specifications matching or exceeding those mentioned above.

In addressing concerns regarding additional processor costs, our strategy involves seamlessly integrating the model directly into the available processors within the system. This approach leverages shared computation power and minimizes the need for supplementary processors and associated expenses. It’s important to note that while the model training phase may necessitate significant computational power and higher hardware specifications for the processor, the post-deployment operational requirements are notably less demanding. This observation affords us substantial flexibility to adjust hardware specifications to levels much lower than

TABLE 9. Analysis of detection accuracy, latency, and model parameters for 0.0001, 0.005 and 0.01 second displacement of Hall sensor using various network architectures where P, R, F1, IFT and Eff stand for precision, recall, F1, inference time and efficiency score respectively.

Network	Metric Type	0.0001 sec	0.005 sec	0.01 sec
MobileNetV1	P	0.95	0.98	1.00
	R	0.94	0.98	1.00
	F1	0.94	0.98	1.00
	IFT	0.02s	0.018s	0.018s
	Eff	**0.97**	**0.99**	**0.99**
MobileNetV2	P	0.91	0.96	0.99
	R	0.91	0.95	0.99
	F1	0.91	0.95	0.99
	IFT	0.025s	0.023s	0.021s
	Eff	**0.95**	**0.98**	**0.99**
VGG 16	P	0.93	0.98	0.97
	R	0.93	0.98	0.96
	F1	0.92	0.98	0.97
	IFT	0.196s	0.174s	0.16s
	Eff	**0.46**	**0.49**	**0.48**
ResNet50	P	0.62	0.67	0.73
	R	0.61	0.67	0.66
	F1	0.60	0.67	0.62
	IFT	0.067s	0.069s	0.086s
	Eff	**0.34**	**0.41**	**0.39**
DenseNet121	P	0.97	0.99	0.99
	R	0.97	0.99	0.99
	F1	0.97	0.99	0.99
	IFT	0.075s	0.071s	0.076s
	Eff	**0.83**	**0.82**	**0.79**
InceptionV3	P	0.94	0.95	0.99
	R	0.94	0.95	0.99
	F1	0.94	0.94	0.99
	IFT	0.046s	0.048s	0.045s
	Eff	**0.62**	**0.61**	**0.63**
Xception	P	0.83	0.98	0.97
	R	0.83	0.98	0.97
	F1	0.83	0.98	0.97
	IFT	0.071s	0.077s	0.089s
	Eff	**0.52**	**0.58**	**0.53**
EfficientNetB0	P	0.24	0.25	0.25
	R	0.50	0.50	0.50
	F1	0.33	0.33	0.33
	IFT	0.039s	0.044s	0.038s
	Eff	**0.60**	**0.58**	**0.59**

those of the aforementioned hardware platform, ensuring cost-effectiveness without compromising performance.

However, it's crucial to acknowledge that reducing hardware specifications may result in an associated increase in inference time. Although we haven't explored this aspect yet, it remains an area for potential investigation and optimization, underscoring our commitment to continuously enhancing the efficiency and effectiveness of our solution.

In summary, the analyses collectively provide a nuanced understanding of each network architecture's performance, guiding the selection of models based on specific requirements such as accuracy, latency, and model simplicity across different displacement intervals.

V. CONCLUSION

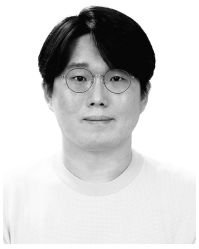
In the realm of Hall sensor displacement detection, our analysis reveals that MobileNetV1 and MobileNetV2 stand out as exemplary models, striking a commendable balance between accuracy, latency, and model simplicity. These architectures prove particularly efficient for applications where optimal performance across these three crucial dimensions is paramount. DenseNet121, while exhibiting a slightly higher latency, emerges as a leader in precision and robust classification metrics, making it a compelling choice for scenarios prioritizing precision over rapid response. Nevertheless, it's essential to navigate the trade-offs associated with other models. VGG 16 and ResNet50, while achieving high accuracy, come at the cost of increased latency and a larger number of parameters. Understanding these trade-offs is pivotal for applications where accuracy remains a priority despite potential delays. On the contrary, EfficientNetB0, with its low latency, faces challenges in delivering robust classification metrics, prompting careful consideration in scenarios where accuracy and precision are paramount.

The introduction of an efficiency score, factoring in accuracy, latency, and the number of parameters, provides a comprehensive metric for model assessment. MobileNetV1 and MobileNetV2 lead in this efficiency metric, underscoring their favorable trade-off between accuracy, latency, and model complexity. As we move forward, the displacement-dependent performance analysis sheds light on the models' adaptability to varying displacement intervals, demonstrating improvements in accuracy metrics with a simultaneous decrease in latency. The findings guide informed model selection, ensuring a judicious balance across accuracy, latency, and model simplicity in diverse application scenarios. Continuous monitoring and evaluation will further refine these insights, contributing to ongoing advancements in sensor displacement detection technologies.

REFERENCES

[1] T. Shi, Y. Guo, P. Song, and C. Xia, "A new approach of minimizing commutation torque ripple for brushless DC motor based on DC–DC converter," *IEEE Trans. Ind. Electron.*, vol. 57, no. 10, pp. 3483–3490, Oct. 2010.
[2] I. Boldea, L. N. Tutelea, L. Parsa, and D. Dorrell, "Automotive electric propulsion systems with reduced or no permanent magnets: An overview," *IEEE Trans. Ind. Electron.*, vol. 61, no. 10, pp. 5696–5711, Oct. 2014.
[3] X. Nian, F. Peng, and H. Zhang, "Regenerative braking system of electric vehicle driven by brushless DC motor," *IEEE Trans. Ind. Electron.*, vol. 61, no. 10, pp. 5798–5808, Oct. 2014.
[4] J. Fang, W. Li, and H. Li, "Self-compensation of the commutation angle based on DC-link current for high-speed brushless DC motors with low inductance," *IEEE Trans. Power Electron.*, vol. 29, no. 1, pp. 428–439, Jan. 2014.
[5] Y. Liu, J. Zhao, M. Xia, and H. Luo, "Model reference adaptive control-based speed control of brushless DC motors with low-resolution Hall-effect sensors," *IEEE Trans. Power Electron.*, vol. 29, no. 3, pp. 1514–1522, Mar. 2014.
[6] X. Song, B. Han, S. Zheng, and J. Fang, "High-precision sensorless drive for high-speed BLDC motors based on the virtual third harmonic back-EMF," *IEEE Trans. Power Electron.*, vol. 33, no. 2, pp. 1528–1540, Feb. 2018.
[7] Q. Zhang and M. Feng, "Fast fault diagnosis method for Hall sensors in brushless DC motor drives," *IEEE Trans. Power Electron.*, vol. 34, no. 3, pp. 2585–2596, Mar. 2019.

[8] D. Mohanraj, R. Aruldavid, R. Verma, K. Sathiyasekar, A. B. Barnawi, B. Chokkalingam, and L. Mihet-Popa, "A review of BLDC motor: State of art, advanced control techniques, and applications," *IEEE Access*, vol. 10, pp. 54833–54869, 2022.
[9] S. Sashidhar, V. Guru Prasad Reddy, and B. G. Fernandes, "A single-stage sensorless control of a PV-based bore-well submersible BLDC motor," *IEEE J. Emerg. Sel. Topics Power Electron.*, vol. 7, no. 2, pp. 1173–1180, Jun. 2019.
[10] M. Ahmed Baba, M. Naoui, and M. Cherkaoui, "Fault-tolerant control strategy for Hall sensors in BLDC motor drive for electric vehicle applications," *Sustainability*, vol. 15, no. 13, p. 10430, Jul. 2023.
[11] J. S. Park, K.-D. Lee, S. G. Lee, and W.-H. Kim, "Unbalanced ZCP compensation method for position sensorless BLDC motor," *IEEE Trans. Power Electron.*, vol. 34, no. 4, pp. 3020–3024, Apr. 2019.
[12] X. Zhang, Y. Zhao, H. Lin, S. Riaz, and H. Elahi, "Real-time fault diagnosis and fault-tolerant control strategy for Hall sensors in permanent magnet brushless DC motor drives," *Electronics*, vol. 10, no. 11, p. 1268, May 2021.
[13] L. Dong, J. Jatskevich, Y. Huang, M. Chapariha, and J. Liu, "Fault diagnosis and signal reconstruction of Hall sensors in brushless permanent magnet motor drives," *IEEE Trans. Energy Convers.*, vol. 31, no. 1, pp. 118–131, Mar. 2016.
[14] G. Scelba, G. De Donato, M. Pulvirenti, F. G. Capponi, and G. Scarcella, "Hall-effect sensor fault detection, identification, and compensation in brushless DC drives," *IEEE Trans. Ind. Appl.*, vol. 52, no. 2, pp. 1542–1554, Mar. 2016.
[15] X. Sun, T. Li, Z. Zhu, G. Lei, Y. Guo, and J. Zhu, "Speed sensorless model predictive current control based on finite position set for PMSHM drives," *IEEE Trans. Transport. Electrific.*, vol. 7, no. 4, pp. 2743–2752, Dec. 2021.
[16] X. Sun, T. Li, X. Tian, and J. Zhu, "Fault-tolerant operation of a six-phase permanent magnet synchronous hub motor based on model predictive current control with virtual voltage vectors," *IEEE Trans. Energy Convers.*, vol. 37, no. 1, pp. 337–346, Mar. 2022.
[17] S. S. Kuruppu, S. G. Abeyratne, and S. Hettiarachchi, "Modeling and detection of dynamic position sensor offset error in PMSM drives," *IEEE Access*, vol. 11, pp. 36741–36752, 2023.
[18] P. H. Kumar and V. T. Somasekhar, "An enhanced fault-tolerant and autoreconfigurable BLDC motor drive for electric vehicle applications," *IEEE J. Emerg. Sel. Topics Ind. Electron.*, vol. 4, no. 1, pp. 368–380, Jan. 2023.
[19] M. Ebadpour, N. Amiri, and J. Jatskevich, "Fast fault-tolerant control for improved dynamic performance of hall-sensor-controlled brushless DC motor drives," *IEEE Trans. Power Electron.*, vol. 36, no. 12, pp. 14051–14061, Dec. 2021.
[20] X. Zhang, H. Lin, and Y. Zhao, "Fault detection and compensation strategy of brushless DC motor with fault in Hall sensors," in *Proc. 22nd Int. Conf. Electr. Mach. Syst. (ICEMS)*, Harbin, China, Aug. 2019, pp. 1–6.
[21] M. Aqil and J. Hur, "A direct redundancy approach to fault-tolerant control of BLDC motor with a damaged Hall-effect sensor," *IEEE Trans. Power Electron.*, vol. 35, no. 2, pp. 1732–1741, Feb. 2020.
[22] A. Mousmi, A. Abbou, and Y. El Houm, "Binary diagnosis of Hall effect sensors in brushless DC motor drives," *IEEE Trans. Power Electron.*, vol. 35, no. 4, pp. 3859–3868, Apr. 2020.
[23] S. Lu, G. Qian, Q. He, F. Liu, Y. Liu, and Q. Wang, "In situ motor fault diagnosis using enhanced convolutional neural network in an embedded system," *IEEE Sensors J.*, vol. 20, no. 15, pp. 8287–8296, Aug. 2020.
[24] Z. Li, Y. Wang, and J. 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.
[25] M. Akmal, "Tensor factorization and attention-based CNN-LSTM deep-learning architecture for improved classification of missing physiological sensors data," *IEEE Sensors J.*, vol. 23, no. 2, pp. 1286–1294, Jan. 2023.
[26] I.-H. Kao, W.-J. Wang, Y.-H. Lai, and J.-W. Perng, "Analysis of permanent magnet synchronous motor fault diagnosis based on learning," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 2, pp. 310–324, Feb. 2019.
[27] K. S. K. Chu, K. W. Chew, and Y. C. Chang, "Fault-diagnosis and fault-recovery system of Hall sensors in brushless DC motor based on neural networks," *Sensors*, vol. 23, no. 9, p. 4330, Apr. 2023.
[28] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," 2018, *arXiv:1801.04381*.
[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[31] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2016, *arXiv:1608.06993*.
[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*.
[33] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 2016, *arXiv:1610.02357*.
[34] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015, *arXiv:1512.00567*.
[35] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*.
[36] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory To Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[37] N. S. Keskar and R. Socher, "Improving generalization performance by switching from Adam to SGD," 2017, *arXiv:1712.07628*.
[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.
[39] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," 2019, *arXiv:1911.02685*.



SEUL KI HONG received the B.S., M.S., and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 2009, 2011, and 2015, respectively. He was a Senior Engineer with Samsung Electronics, South Korea. He is currently an Assistant Professor with the Department of Semiconductor Engineering, Seoul National University of Science and Technology, Seoul, South Korea.

YONGKEUN LEE (Senior Member, IEEE) received the B.S. degree in material engineering from Iowa State University, Ames, IA, USA, in 1991, the M.S. degree in material science from Columbia University, New York, NY, USA, in 1993, and the Ph.D. degree in materials engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1996. His professional journey includes roles, such as an Assistant Professor with Nanyang Technological University, Singapore, and a Principal Engineer with Samsung Electronics LCD Business, South Korea. From 2007 to 2009, he was with the Dean of the Graduate School of Nano IT Design Fusion, Seoul National University of Science and Technology, Seoul, South Korea. Currently, he hold the position of a Professor with the Department of Semiconductor Engineering, Seoul National University of Science and Technology.