Fast and Accurate Design of BLDC Motors Using Bayesian Neural Networks

Original Scientific Paper

Son T. Nguyen *
Hanoi University of Science and Technology
School of Electrical and Engineering,
Faculty of Electrical Engineering
Dai Co Viet Street, Hanoi, Vietnam
son.nguyenthanh@hust.edu.vn

Trung T. Cao
Hanoi University of Science and Technology
School of Electrical and Engineering,
Faculty of Electrical Engineering
Dai Co Viet Street, Hanoi, Vietnam
trung.caothanh@hust.edu.vn

Tu M. Pham
Hanoi University of Science and Technology
School of Electrical and Engineering,
Faculty of Electrical Engineering
Dai Co Viet Street, Hanoi, Vietnam
tu.phamminh@hust.edu.vn

Tinh V. Lai
Hanoi University of Science and Technology
School of Electrical and Engineering,
Faculty of Electrical Engineering
Dai Co Viet Street, Hanoi, Vietnam
tinh.lv240456e@sis.hust.edu.vn

Anh Hoang
Hanoi University of Science and Technology
School of Electrical and Engineering,
Faculty of Electrical Engineering
Dai Co Viet Street, Hanoi, Vietnam
anh.hoang@hust.edu.vn

Hoang Q. Ha
Hanoi University of Science and Technology
School of Electrical and Engineering,
Faculty of Electrical Engineering
Dai Co Viet Street, Hanoi, Vietnam
hoang.hq240414e@sis.hust.edu.vn

*Corresponding author

Abstract *– Brushless direct current (BLDC) motors are gaining popularity over traditional direct current (DC) motors due to their higher efficiency, compact size, and precise control capabilities. This study proposes a fast and accurate approach to BLDC motor design using a Bayesian neural network (BNN). The BNN, a specialized form of the multi-layer perceptron (MLP), offers strong resistance to overfitting and performs effectively with noisy or limited datasets, making it well-suited for complex motor design problems. In the proposed method, the BNN is applied within an inverse modeling framework to map desired motor performance parameters to the corresponding design variables. A dataset for an outer-rotor BLDC motor—containing both design parameters and the resulting output torque—is generated through finite element analysis (FEA). Finally, a demonstration of BLDC motor design using the BNN validates the effectiveness of the proposed approach.*

Keywords: *BLDC motors, Bayesian neural networks, finite element analysis*

Received: March 18, 2025; Received in revised form: August 19, 2025; Accepted: October 1, 2025

1. INTRODUCTION

Brushless DC (BLDC) motors have been extensively studied in recent decades due to their high efficiency, reliability, and precise motion control capabilities. Their compact design and lightweight construction facilitate accurate speed and torque regulation, making them well-suited for modern engineering applications [1]. Unlike traditional brushed DC motors, BLDC motors employ electronic commutation instead of mechanical commutators. Consequently, they have been widely adopted in diverse fields such as industrial automation, electric vehicles, drones, medical devices, and home appliances, where precise speed control, low maintenance, and high efficiency are critical requirements.

Finite element analysis (FEA) is essential for designing and optimizing electromagnetic devices such as BLDC motors. It enables engineers to evaluate motor performance under various operating conditions, and

by simulating electromagnetic behaviors and mechanical stresses, it allows for precise design adjustments before production. This approach reduces development costs while improving efficiency, reliability, and overall performance [2].

A major challenge in BLDC motor design is cogging torque, which affects smooth operation and overall efficiency. Extensive research has been conducted to analyze and mitigate this issue. Studies indicate that factors such as stator tooth width and slot–pole alignment significantly influence cogging torque and can be optimized to enhance motor performance [3]. In outer-rotor BLDC motors, optimizing the stator core design is an effective strategy for reducing cogging torque [4], while in inner-rotor BLDC motors, segmenting the rotor's permanent magnets is commonly employed to minimize cogging effects [5]. Furthermore, field-oriented control (FOC) is an advanced technique that reduces cogging torque by incorporating dominant harmonics from the cogging torque waveform into the q-axis current reference, thereby counteracting torque ripples and minimizing speed variations [6].

BLDC motors with trapezoidal back electromotive force (BEMF) traditionally require six rotor position signals for inverter control, typically detected by Hall-effect sensors embedded in the motor. While effective, these sensors increase manufacturing costs and are sensitive to temperature variations, which can reduce system reliability. To overcome these limitations, sensorless control techniques have been extensively developed over the past two decades. A common approach estimates rotor position and regulates speed by detecting BEMF zero crossings from terminal voltages [7]; however, this method performs poorly at low speeds due to weak induced voltages. To address this issue, a novel sensorless position detection technique based on a speed-independent position function has been proposed [8], significantly improving estimation accuracy and enhancing system performance across a wide speed range.

The design of electromagnetic devices—such as electric motors, transformers, and inductors—is a complex process that requires balancing multiple performance criteria, including efficiency, thermal management, weight, and material costs [9, 10]. This challenge is often formulated as an optimization problem, where the objective is to minimize or maximize a specific cost function, such as power loss, torque ripple, or electromagnetic interference. Stochastic optimization methods, including genetic algorithms (GA) [11], particle swarm optimization (PSO) [12], and simulated annealing (SA) [13], are widely applied because of their effectiveness in exploring complex, multi-dimensional design spaces. These methods use iterative procedures to evaluate design parameters at each step, progressively refining solutions to approach an optimal configuration.

Artificial neural networks (ANNs) are transforming the design of electromagnetic devices by enhancing simulation efficiency, optimizing design parameters, and addressing complex inverse problems. These computational models can learn and represent intricate nonlinear relationships, making them especially valuable in scenarios where conventional physics-based methods are limited or computationally expensive [14, 15]. A key advantage of ANNs is their ability to process large datasets and detect patterns that traditional methods may overlook, enabling more accurate and faster performance predictions for electromagnetic devices. When integrated with FEA, ANNs facilitate more efficient and precise optimization of permanent magnet (PM) motors [16]. For example, ANNs trained on FEA-generated data can model complex electromagnetic behaviors and predict motor performance under varying conditions. This hybrid approach allows for rapid evaluation of design alternatives, significantly reducing the time and cost associated with physical prototyping and extensive simulation runs.

Bayesian neural networks (BNNs) extend traditional multi-layer perceptron (MLP) neural networks by incorporating principles of Bayesian inference [17]. Unlike conventional MLPs, which learn fixed point estimates for their weights and biases, BNNs treat these parameters as probability distributions, enabling the quantification of uncertainty in predictions. In this study, BNNs are applied to the accurate design of an outer-rotor BLDC motor. FEA was performed to generate a dataset for training the network. Once trained, the BNN computes optimal motor design parameters to achieve a specified target torque. The remainder of this paper is organized as follows: Section 2 discusses the theory of BNNs and their application to regression problems; Section 3 introduces FEA for electromagnetic devices, with emphasis on BLDC motors; Section 4 details the proposed design methodology for the outer-rotor BLDC motor using BNNs; and Section 5 presents the conclusions and outlines directions for future research.

2. BAYESIAN NEURAL NETWORKS FOR REGRESSION PROBLEMS

2.1. MULTI-LAYER PERCEPTRON NETWORKS

A MLP neural network takes a vector of real-valued inputs and computes one or more activation values for the output layer. In a network with a single hidden layer, as illustrated in Fig. 1, the activation of the output layer is calculated as follows:

$$a_k(x) = b_k + \sum_{j=1}^{m} w_{kj} \tanh\left(\bar{b}_j + \sum_{i=1}^{d} \bar{w}_{ji} x_i\right) = b_k + \sum_{j=1}^{m} w_{kj} y_j \quad (1)$$

Here, x_i are real inputs, $\bar{w}_{ji}$ is the weight on the connection from the input unit i to the hidden unit j; similarly, w_{kj} is the weight on the connection from the hidden unit j to the output unit k. The $\bar{b}_j$ and b_k are the biases of the hidden and output units. These weights and biases are the parameters of the MLP neural network. Then the activation $a_k(x)$ are used to compute the outputs of the output layer by using a "linear" activation function as follows:

$$z_k = a_k(x) \tag{2}$$

The training of an MLP neural network aims to minimize a data error function structured in the following way:

$$E_D = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{c}\left(z_k^n - t_k^n\right)^2 \tag{3}$$

In which z_k^n is the k-th output corresponding to the n-th training pattern and t_k^n is the k-th target corresponding to the n-th training pattern.

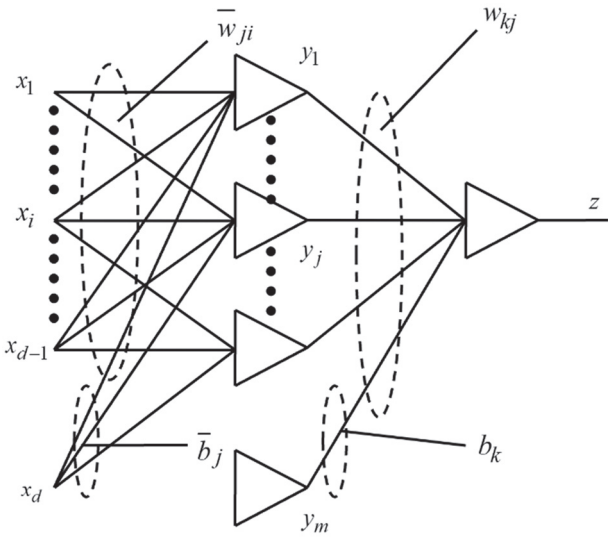

Fig. 1. Structure of MLP neural networks

2.2. NETWORK REGULARIZATION

In MLP neural networks, regularization is employed to prevent any weights and biases from becoming excessively large, as large weights and biases can lead to poor generalization on new test cases. To address this issue, a weight function, E_W, is added to the error function to penalize large weights and biases. Specifically, for regression problems, this approach is utilized to enhance model performance and a total error function, $S(w)$, is defined as follows:

$$S(w) = \beta E_D + \alpha E_W \tag{4}$$

Where β and α are non-negative parameters, also known as " hyperparameters ", need to be determinded. The weight function, E_W, usually originates from the theory of weight priors having the following form:

$$E_W = \frac{1}{2}\|w\|^2 \tag{5}$$

Where w is the vector of the weights and biases in the network.

2.3. BAYESIAN INFERENCE

In Bayesian inference for MLP neural networks, β and α can be automatically determined. This process considers the Gaussian probability distributions of the weights and biases which can give the best generalization. In particular, the vector of weights and biases, w, in the network is adjusted to their most probable values given the training data D. Specifically, the posterior distribution of the vector of weights and biases can be computed using Bayes' rule as follows:

$$p(w|D) = \frac{p(D|w)\,p(w)}{p(D)} \tag{6}$$

In this formula, $p(D|w)$ represents the likelihood function, which captures the information derived from observations. In contrast, the prior distribution $p(w)$ incorporates information based on background knowledge. The denominator, $p(D)$, known as the evidence for the network.

The requirement of small values of weights and biases in the MLP neural network suggests the use of a Gaussian prior distribution for the vector of weights and biases as follows:

$$p(w) = \frac{1}{Z_W(\alpha)}\exp\left(-\frac{\alpha}{2}\|w\|^2\right) \tag{7}$$

Where $ZW(\alpha)$ is a normalization constant given by:

$$Z_W(\alpha) = \left(\frac{2\pi}{\alpha}\right)^{W/2} \tag{8}$$

In equation (8), W is the number of weights and biases in the network. The likelihood function is given by:

$$p(D|w) = \frac{1}{Z_D(\beta)}\exp\left(-\frac{\beta}{2}E_D\right) \tag{9}$$

Where $Z_D(\beta)$ is a normalization factor given by:

$$Z_D(\beta) = \left(\frac{2\pi}{\beta}\right)^{N/2} \tag{10}$$

In equation (10), N is the number of training patterns. At the most probable vector of the weights and biases, , the Hessian matrix of the total error function, A, can be evaluated as follows:

$$A = \nabla\nabla S(w_{MP}) = \beta H + \alpha I \tag{11}$$

Where I is the identity matrix. $H=\nabla\nabla E_D(w_{MP})$ is the Hessian matrix of the data error function at the most probable vector, w_{MP} of weights and biases.

If the posterior distribution of weights and biases is assumed as a Gaussian, then it is given by:

$$p(w|D) = \frac{1}{Z_S}\exp\left(-S(w_{MP}) - \frac{1}{2}\Delta w^T A \Delta w\right) \tag{12}$$

Where $\Delta w = w - w_{MP}$ and Z_S is a normalization constant given by:

$$Z_S = \exp\left(-S(w_{MP})\right)(2\pi)^{W/2}\left(\det A\right)^{-1/2} \tag{13}$$

Re-arranging (6) gives:

$$p(D) = \frac{p(D|w)\,p(w)}{p(w|D)} \tag{14}$$

Substituting (7), (9) and (12) into (14) results in:

$$p(D)=\exp\left(-S(w_{MP})\right)\left(\left(\frac{\beta}{2\pi}\right)^{N/2}\right)\left(\alpha^{W/2}\right)\left(\det A\right)^{-1/2} \quad (15)$$

Taking the logarithm of (15) gives:

$$\ln p(D)=-S(w_{MP})+\frac{N}{2}\ln(\beta)-\frac{N}{2}\ln(2\pi)+ +\frac{W}{2}\ln(\alpha)-\frac{1}{2}\ln(\det A) \quad (16)$$

In this context, the variable *ln p* (*D*) is referred to as the "log evidence". To optimize the log evidence *ln p* (*D*) with respect to α, it is necessary to compute a partial derivative of the log evidence as follows:

$$\frac{\partial \ln p(D)}{\partial \alpha}=-E_W(w_{MP})+\frac{W}{2\alpha}-\frac{1}{2}\frac{\partial\left(\ln(\det A)\right)}{\partial \alpha} \quad (17)$$

In (17), $\frac{\partial(ln(det\,A))}{\partial \alpha}$ is computed as follows:

$$\frac{\partial\left(\ln(\det A)\right)}{\partial \alpha}=\sum_{i=1}^{W}\frac{\alpha}{\lambda_i+\alpha} \quad (18)$$

Where λ_i are the eigenvalues of the Hessian matrix of the data error function, $H=\nabla\nabla E_D(w_{MP})$. Substituting (18) into (17) gives:

$$\frac{\partial \ln p(D)}{\partial \alpha}=-E_W(w_{MP})+\frac{W}{2\alpha}-\frac{1}{2}\sum_{i=1}^{W}\frac{\alpha}{\lambda_i+\alpha} \quad (19)$$

The optimal value of α is determined when the right-hand side of equation (19) is equal to zero to obtain the following equation:

$$2\alpha E_W(w_{MP})=W-\sum_{i=1}^{W}\frac{\alpha}{\lambda_i+\alpha} \quad (20)$$

The right-hand side of equation (20) is equal to a value γ defined as follows:

$$\gamma=W-\sum_{i=1}^{W}\frac{\alpha}{\lambda_i+\alpha}=\sum_{i=1}^{W}\frac{\lambda_i}{\lambda_i+\alpha} \quad (21)$$

If $\lambda_i \gg \alpha$, γ is approximately equal to 1. Whereas, if $\lambda_i \leq \alpha$, γ is near to 0. γ is used to measure the number of "well-determined" parameters in the network.

From equations (20) and (21), α can be determined as follows:

$$\alpha=\frac{\gamma}{2E_W(w_{MP})} \quad (22)$$

Similarly, to optimize the log evidence *ln p* (*D*) with respect to β, it is also necessary to compute a partial derivative of the log evidence as follows:

$$\frac{\partial \ln p(D)}{\partial \beta}=-E_D(w_{MP})+\frac{N}{2\beta}-\frac{1}{2}\frac{\partial\left(\ln(\det A)\right)}{\partial \beta} \quad (23)$$

In (23), $\frac{\partial(ln(det\,A))}{\partial \beta}$ is computed as follows:

$$\frac{\partial\left(\ln(\det A)\right)}{\partial \beta}=\frac{1}{\beta}\sum_{i=1}^{W}\frac{\alpha}{\lambda_i+\alpha} \quad (24)$$

Substituting (24) into (23) gives:

$$\frac{\partial \ln p(D)}{\partial \beta}=-E_D(w_{MP})+\frac{N}{2\beta}-\frac{1}{2\beta}\sum_{i=1}^{W}\frac{\lambda_i}{\lambda_i+\alpha} \quad (25)$$

The optimal value of β is determined when the right-hand side of equation (25) is equal to zero to obtain the following equation:

$$2\beta E_D(w_{MP})=N-\sum_{i=1}^{W}\frac{\lambda_i}{\lambda_i+\alpha}=N-\gamma \quad (26)$$

From equation (26), β can be determined as follows:

$$\beta=\frac{N-\gamma}{2E_D(w_{MP})} \quad (27)$$

Choosing the number of input, hidden, and output nodes in a BNN for regression is similar in principle to standard neural networks, but the Bayesian framework adds a probabilistic perspective that helps control overfitting and provides uncertainty estimates.

- The number of input nodes is equal to number of features (independent variables) in dataset.
- A single output node (the predicted continuous value).
- Number of hidden nodes can be determined from heuristic starting points. For example, the number of hidden nodes can be computed by the following rule:

$$N_{hidden}=\frac{N_{in}+N_{out}}{2} \quad (28)$$

Where N_{in}, N_{out} and N_{hidden} are numbers of input nodes, number of output nodes and number of hidden nodes, respectively.

2.4. THE QUASI-NEWTON METHOD

The training of an MLP neural network involves minimizing the total error function, $S(w)$, through an iterative process. The quasi-Newton method, which extends the gradient descent method, can be effectively used to minimize this error function. In Newton method, the vector of weights and biases of the network can be updated as follows:

$$w_{m+1}=w_m-A_m^{-1}g_m \quad (29)$$

The vector $-Am^{-1}\,g_m$ is called the "Newton direction" or the "Newton step". However, the evaluation of the Hessian matrix, Am^{-1}, can be very computational. From equation (29), we can form the relationship between the weight vectors at steps *m* and *m*+1 as follows:

$$w_{m+1}=w_m-\alpha_m F_m g_m \quad (30)$$

From equation (30), if $\alpha_m=1$ and $F_m=A_m^{-1}$, we have the Newton method, while if $Fm=I$, we have the gradient descent method with the learning rate α_m.

F_m can be chosen to approximate the Hessian matrix. In addition, F_m must be positive definite so that for small α_m we can obtain a descent method. In practice, the value of α_m can be found by a "line search". Equation (30) is known as the quasi-Newton condition. The most successful method to compute F_m is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula as follows:

$$F_{m+1} = F_m + \frac{pp^T}{p^T v} - \frac{(F_m v)v^T F_m}{v^T F_m v} + \left(v^T F_m v\right)uu^T \quad (31)$$

Where p, v and u are defined as:

$$p = w_{m+1} - w_m \quad (32)$$

$$v = g_{m+1} - g_m \quad (33)$$

$$u = \frac{p}{p^T v} - \frac{F_m v}{v^T F_m v} \quad (34)$$

Finally, training MLP neural networks using Bayesian inference involves several key steps as follows:

Step 1: Initialize the weights and biases for the network, and initialize the values for the hyperparameters β and α.

Step 2: Minimize the cost function $S(w)$ (4) using the quasi-Newton method and calculate γ as follows:

$$\gamma_{old} = \sum_{i=1}^{W} \frac{\lambda_i}{\lambda_i + \alpha_{old}} \quad (35)$$

Where λ_i are the eigenvalues of the Hessian matrix of the data error function, $H=\nabla\nabla E_D$.

Step 3: When the cost function has reached a local minimum, re-estimate the values of the hyperparameters as follows:

$$\alpha_{new} = \frac{\gamma_{old}}{2E_W} \quad (36)$$

$$\beta_{new} = \frac{N - \gamma_{old}}{2E_D} \quad (37)$$

Step 4: Repeat steps 2 and 3 until the convergence.

3. FEA FOR BLDC MOTORS

The operating principle of many electromechanical devices is based on electromagnetic theory, and these devices can often be mathematically described using partial differential equations (PDEs). Analyzing such devices therefore requires methods for solving PDEs. Since analytical techniques often fail to provide accurate solutions, the finite element method (FEM) has emerged as a powerful tool for addressing PDEs in the analysis of electromagnetic systems. The FEA process for a specific electromagnetic device typically involves four key steps:

- Discretize the solution region into finite elements.
- Derive the governing equations for each individual element.
- Assemble all the finite elements within the solution region.
- Solve the resulting set of equations.

In this study, Finite Element Method Magnetics (FEMM), a free and open-source computational tool, is utilized to analyze the performance and characteristics of a BLDC motor. FEMM has gained popularity in both academic research and industrial applications due to its accessibility, efficiency, and capability to handle complex electromagnetic problems using FEM [18].

To set up FEMM, the problem type must first be defined, typically as a planar or axisymmetric magnetic problem with appropriate units. The geometry of the device, such as the stator, rotor, air gap, and windings, is then drawn or imported, and each region is assigned suitable material properties from the FEMM library or custom definitions. Boundary conditions, excitations (such as currents or permanent magnet properties), and circuit parameters are specified to represent the operating conditions. The model is then discretized using a finite element mesh, refined in critical regions like the air gap for higher accuracy. Finally, the analysis is run, and the postprocessor is used to visualize field distributions and extract key performance quantities such as flux linkage, torque, or inductance.

A key advantage of FEMM is its ability to integrate with MATLAB. This interaction enables users to define complex electromagnetic problems, execute simulations, and extract results programmatically within the MATLAB environment. Through MATLAB commands, researchers can automate the analysis process, perform parametric studies, and optimize motor designs more efficiently. This integration is particularly valuable for iterative design workflows, where multiple simulations are required to evaluate the impact of design parameter variations on motor performance. Fig. 2 shows a commercial outer-rotor BLDC motor, while Fig. 3 illustrates its 2D finite element analysis (FEA) conducted using FEMM.

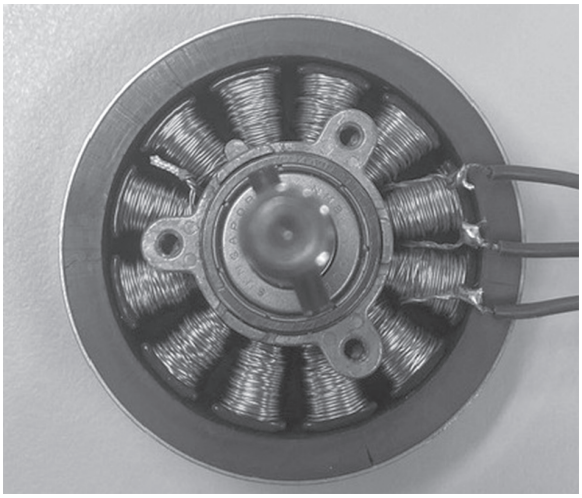

Fig. 2. Commercial outer-rotor BLDC motor

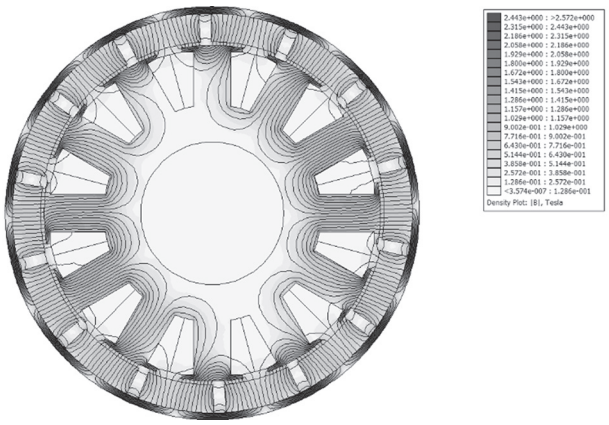

Fig. 3. 2D-FEA of a commercial outer-rotor BLDC motor

4. DESIGN OF BLDC MOTORS USING BAYESIAN NEURAL NETWORKS

This section presents a detailed procedure for designing a small-scale outer-rotor BLDC motor. The motor's dimensional and electrical parameters are as follows:

- Stator outer radius (*rso*).
- Stator inner radius (*rsi*).
- Magnet thickness (*dm*).
- Can thickness (*dc*).
- Depth of slot opening (*ds*).
- Pole fraction spanned by the magnet (*fm*).
- Pole fraction spanned by the iron (*fp*).
- Width of the tooth as a fraction of the pole pitch at the stator (*ft*).
- Back iron thickness as a fraction of tooth thickness (*fb*).
- Stator to magnet mechanical clearance (*go*).
- Axial length of the machine (*hh*).
- Peak current density in the winding (*Jpk*).

Fig. 4 illustrates the process of generating the dataset used for training the Bayesian neural network (BNN). The dataset was constructed to capture the complex nonlinear relationships between the design parameters of the BLDC motor and the corresponding output torque values. To ensure diversity and representativeness, multiple variations of key design parameters—such as stator and rotor dimensions, air gap length, winding configurations, and material properties—were systematically simulated. Each configuration was analyzed using FEA, producing torque outputs under specified operating conditions. This process resulted in a dataset consisting of 2000 distinct patterns ($N = 2000$), derived from a commercial outer-rotor BLDC motor. The large number of samples provides sufficient coverage of the design space, allowing the BNN to learn both linear and nonlinear dependencies effectively. By incorporating such a dataset, the network is better equipped to generalize across unseen configurations, thereby improving its predictive accuracy and robustness in motor design optimization. The ranges for the design parameters and output torque of the motor are provided in Table 1. Lastly, the dataset was utilized to train the BNN according to the procedure outlined in Fig. 5. The BNN has the following structure:

- An input corresponding to the desired output torque
- Six units in the hidden layer
- Twelve outputs in the output layer, representing the twelve design parameters that need to be determined

In this research, the training algorithm of the BNN is based on the quasi-Newton optimization method. This approach is chosen because it provides a balance between computational efficiency and convergence accuracy. Unlike traditional gradient descent methods, which may suffer from slow convergence or becoming trapped in local minima, quasi-Newton methods approximate the Hessian matrix of second-order derivatives to achieve faster and more stable convergence. By leveraging curvature information of the error surface, the algorithm can adjust the learning step more intelligently, thereby reducing the number of iterations required to reach an optimal solution. This makes the quasi-Newton method particularly suitable for training complex models such as BNNs, where robustness and efficiency are essential in handling high-dimensional parameter spaces and ensuring reliable generalization.

The number of hidden nodes is initially determined using a heuristic approach, guided by the sizes of the input and output layers. This provides a reasonable starting point, ensuring that the network has sufficient capacity to capture the underlying nonlinear relationships without becoming overly complex. To enable effective learning from the training data, the number of training epochs is set to 1000. This choice balances providing enough iterations for convergence with avoiding excessive training that could lead to overfitting. By combining an informed initialization of hidden nodes with an adequate number of training epochs, the BNN is structured to achieve reliable performance, accurately capturing the mapping between design parameters and output responses with both precision and stability.

Table 2 presents the variations in hyperparameters across different re-estimation periods. These adjustments result from the Bayesian optimization process, which systematically refines hyperparameters to enhance model accuracy and stability. By periodically re-estimating and updating these parameters, the BNN sustains optimal performance throughout training, leading to more accurate predictions and improved generalization to unseen data.

Once trained, the BNN acts as an effective mapping tool, translating desired output torque values into corresponding motor design parameters. Fig. 5 illustrates the relationship between target torque and the associated design variables. BNN's ability to accurately map

these relationships is crucial for optimizing BLDC motor designs, as it streamlines the design process and reduces the need for extensive trial-and-error experimentation.

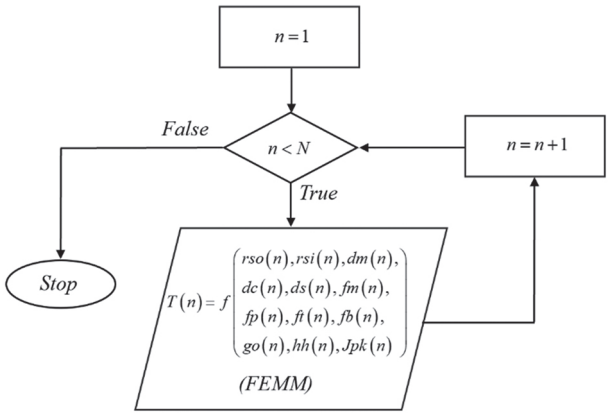

Fig. 4. Principle of generating the dataset for the BNN training

Table 1. Ranges of the design parameters and output torque of the motor

Design Parameters	Ranges
rso(mm)	[22.5004 27.4921]
rsi(mm)	[9.0005 10.9982]
dm(mm)	[3.6000 4.3998]
dc(mm)	[0.9001 1.0996]
ds(mm)	[0.4500 0.5500]
fm	[0.7715 0.9428]
fp	[0.6302 0.7700]
ft	[0.9000 1.0998]
fb	[0.9000 1.0999]
go(mm)	[0.4500 0.5499]
hh(mm)	[22.5019 27.4992]
Jpk(*MA*/m^2)	[0.7201 0.8798]
Output Torque (N.m)	[0.0470 0.2275]

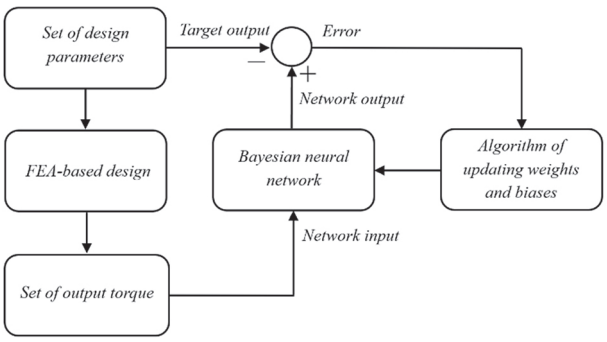

Fig. 5. Principle of updating the weights and biases of the BNN

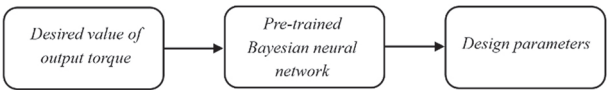

Fig. 6. Principle of calculating the design parameters of the motor using the pre-trained BNN

Table 2. Change of the hyperparameters according to the periods of re-estimation

Re-estimation Periods	α	β
1	0.3304	479.271
2	1.2190	479.4954
3	2.5027	479.3550

The final motor design parameters, obtained after completing the optimization process, are outlined in Table 3. These parameters represent the optimized configuration derived from the BNN's predictions, reflecting a balance between performance objectives and design constraints. The information in Table 3 serves as a comprehensive summary of the key design variables, providing a clear reference for evaluating the effectiveness of the BNN-driven optimization approach.

Table 3. Design parameters of the motor obtained after the design process

Parameters	Values
rso(mm)	25.1423
rsi(mm)	10.0374
dm(mm)	3.9904
dc(mm)	1.0000
ds(mm)	0.4988
fm	0.8576
fp	0.6998
ft	0.9950
fb	0.9980
go(mm)	0.4985
hh(mm)	25.0006
Jpk(*MA*/m^2)	0.8011

Table 4 compares the target output torque with the actual output torque, showing only a very small percentage difference. This minimal error confirms that the BNN can accurately predict motor performance by capturing the complex relationships between design parameters and torque. The low error rate demonstrates both high precision and strong reliability, ensuring that the predicted torque is almost identical to the desired target. Such accuracy is crucial for optimizing BLDC motor designs, as it enables improved performance, reduces the number of design iterations, and increases confidence in the model's predictions.

This research does not incorporate optimization techniques for BLDC motor design—such as cost minimization, compact dimensions, or material efficiency—into its framework. Instead, the focus is placed on developing a fast and accurate design methodology. While this approach provides valuable insights into the design process, the absence of optimization considerations limits its applicability in scenarios where economic feasibility, space constraints, or manufacturing efficiency are critical. Therefore, integrating optimization strategies in future studies would enhance the practical relevance of the proposed design method.

Table 4. Comparison between the target and true torques

Target Torque (N.m)	True Torque (N.m)	Error (%)
0.1	0.0967	3.3
0.125	0.1256	-0.48
0.175	0.1751	-0.0571
0.2	0.2018	-0.9

5. CONCLUSIONS

This study highlights the effectiveness of BNNs in optimizing the design of BLDC motors, demonstrating their potential as a powerful alternative to conventional optimization methods. To support the training process, FEA was employed to generate a comprehensive dataset that accurately captures the complex nonlinear relationships between design parameters—such as dimensions, material properties, and winding configurations—and motor performance characteristics, including torque, efficiency, and thermal behavior. This data set enables BNN to learn these intricate mappings and provide reliable performance predictions across a wide range of operating conditions.

A key contribution of this research is the integration of Bayesian optimization for hyperparameter tuning of the multi-layer perceptron (MLP) structure underlying the BNN in BLDC motor design. Unlike manual trial-and-error methods, Bayesian optimization systematically explores the hyperparameter space in a data-driven manner, leading to improved training accuracy, enhanced stability, and reduced computational cost. This approach also minimizes the risk of overfitting while significantly improving the model's ability to generalize to unseen design scenarios.

Looking forward, future research will extend the application of BNNs in BLDC motor design optimization, with a particular focus on addressing practical constraints such as reducing material costs, improving energy efficiency, and enhancing manufacturability. Moreover, incorporating multi-objective optimization—balancing trade-offs between cost, weight, torque ripple, and thermal performance—could further advance the applicability of BNN-based methods in industrial motor design.

ACKNOWLEDGEMENT

This research is funded by Hanoi University of Science and Technology (HUST) under project number T2024-PC-059.

6. REFERENCES:

[1] D. Mohanraj, R. Aruldavid, R. Verma, K. Sathiyasekar, A. B. Barnawi, B. Chokkalingam, "A Review of BLDC Motor: State of Art, Advanced Control Techniques, and Applications", IEEE Access, Vol. 10, 2022, pp. 54833-54869.

[2] T.-Y. Lee, M.-K. Seo, Y.-J. Kim, S.-Y. Jung, "Motor Design and Characteristics Comparison of Outer-Rotor-Type BLDC Motor and BLAC Motor Based on Numerical Analysis", IEEE Transactions on Applied Superconductivity, Vol. 26, No. 4, 2016, pp. 1-6.

[3] Y. L. Karnavas, I. D. Chasiotis, A. D. Gkiokas, "An Investigation Study Considering the Effect of Magnet Type, Slot Type and Pole-Arc to Pole-Pitch Ratio Variation on PM Brushless DC Motor Design", Proceedings of the 5th International Conference on Mathematics and Computers in Sciences and Industry, Corfu, Greece, 25-27 August 2018, pp. 7-13.

[4] K.-J. Han, H.-S. Cho, D.-H. Cho, H.-K. Jung, "Optimal core shape design for cogging torque reduction of brushless DC motor using genetic algorithm", IEEE Transactions on Magnetics, Vol. 36, No. 4, 2000, pp. 1927-1931.

[5] R. Setiabudy, H. Wahab, Y. S. Putra, "Reduction of cogging torque on brushless direct current motor with segmentation of magnet permanent", Proceedings of the 4th International Conference on Information Technology, Computer, and Electrical Engineering, Semarang, Indonesia, 18-19 October 2017, pp. 81-86.

[6] M. Sumega, P. Rafajdus, G. Scelba, M. Stulrajter, "Control Strategies for the Identification and Reduction of Cogging Torque in PM Motors", Proceedings of the International Conference on Electrical Drives & Power Electronics, The High Tatras, Slovakia, 24-26 September 2019, pp. 74-80.

[7] X. Song, B. Han, K. Wang, "Sensorless Drive of High-Speed BLDC Motors Based on Virtual Third-Harmonic Back EMF and High-Precision Compensation", IEEE Transactions on Power Electronics, Vol. 34, No. 9, 2019, pp. 8787-8796.

[8] T. Li, J. Zhou, "High-Stability Position-Sensorless Control Method for Brushless DC Motors at Low Speed", IEEE Transactions on Power Electronics, Vol. 34, No. 5, 2019, pp. 4895-4903.

[9] Y. Zhao, S. L. Ho, W. N. Fu, "A Novel Fast Remesh-Free Mesh Deformation Method and Its Application to Optimal Design of Electromagnetic Devices", IEEE Transactions on Magnetics, Vol. 50, No. 11, 2014.

[10] H. R. E. H. Bouchekara, "Optimal design of electromagnetic devices using a black-hole-based opti-

mization technique", IEEE Transactions on Magnetics, Vol. 49, No. 12, 2013, pp. 5709-5714.

[11] J. Gao, L. Dai, W. Zhang, "Improved genetic optimization algorithm with subdomain model for multi-objective optimal design of SPMSM", CES Transactions on Electrical Machines and Systems, Vol. 2, No. 1, 2018, pp. 160-165.

[12] J. Hwan Lee, J.-W. Kim, J.-Y. Song, D.-W. Kim, Y.-J. Kim, S.-Y. Jung, "Distance-Based Intelligent Particle Swarm Optimization for Optimal Design of Permanent Magnet Synchronous Machine", IEEE Transactions on Magnetics, Vol. 53, No. 6, 2017.

[13] T. Renyuan, S. Jianzhong, L. Yan, C. Xiang, "Optimization of electromagnetic devices by using intelligent simulated annealing algorithm", IEEE Transactions on Magnetics, Vol. 34, No. 5, 1998, pp. 2992-2995.

[14] D. Cherubini, A. Fanni, A. Montisci, P. Testoni, "Inversion of MLP neural networks for direct solution of inverse problems", IEEE Transactions on Magnetics, Vol. 41, No. 5, 2005, pp. 1784-1787.

[15] I. Marinova, C. Panchev, D. Katsakos, "A neural network inversion approach to electromagnetic device design", IEEE Transactions on Magnetics, Vol. 36, No. 4, 2000, pp. 1080-1084.

[16] L. Hadjout, N. Takorabet, R. Ibtiouen, S. Mezani, "Optimization of instantaneous torque shape of PM motors using artificial neural networks based on FE results", IEEE Transactions on Magnetics, Vol. 42, No. 4, 2006, pp. 1283-1286.

[17] S. T. Nguyen, T. M. Pham, A. Hoang, L. V. Trieu, T. T. Cao, "Bayesian Inference for Regularization and Model Complexity Control of Artificial Neural Networks in Classification Problems", Bayesian Inference-Recent Trends, IntechOpen, 2023.

[18] Finite Element Method Magnetics, https://www.femm.info/wiki/HomePage (accessed: 2025)