Journal of Science and Technology – Smart Systems and Devices
Volume 36, Issue 1, January 2026, 018-024

Static Hand Gesture Recognition Using a Low-Cost Data Glove and Bayesian Neural Network

Son T. Nguyen*, Tu M. Pham, Anh Hoang, Trung T. Cao, Quang M. Tran
Hanoi University of Science and Technology, Ha Noi, Vietnam
**Corresponding author email: son.nguyenthanh@hust.edu.vn*

Abstract

Gesture recognition has become an important focus in human-machine interaction (HMI). Static hand gesture recognition is particularly useful for detecting the intuitive intentions of individuals who are deaf or mute. In recent years, various data gloves have been developed to capture static hand gestures. These gloves, worn like regular gloves, serve as input devices for HMI systems. A low-cost data glove can be built using flex sensors to detect finger bending, enabling the collection of data on finger positions for different static gestures. This data can then be interpreted by a computer program. While many commercial data gloves are bundled with software, they are often prohibitively expensive. This research develops a low-cost data glove using flex sensors and an Arduino Nano. For accurate static gesture recognition, a Bayesian neural network (BNN) is employed to classify different gestures. To optimize training, the scaled-conjugate gradient method, an efficient, automated algorithm, is used to update the network's weights and biases.

Keywords: Data glove, static hand gesture recognition, Bayesian neural network.

1. Introduction

In the context of human–machine interface (HMI) systems for individuals who are deaf or mute, machine learning (ML) plays a critical role in enabling effective communication. One practical approach is the use of static or dynamic hand gestures, which provide a natural and intuitive medium for expressing a user's intentions. Such gestures can be recognized by the system and translated into meaningful commands or messages, thereby facilitating communication with others or activating assistive devices [1-6]. By leveraging ML algorithms, these systems can achieve higher recognition accuracy, adapt to individual gesture variations, and maintain reliable performance under diverse operating conditions.

Static hand gesture recognition methods can be broadly divided into data glove–based and vision-based approaches. The former relies on gloves equipped with sensors to capture finger-bending data [7-11], while the latter uses cameras to acquire images of gestures [12-14]. Although vision-based methods are popular, they can be affected by poor lighting conditions, a limitation not present in data glove-based methods.

Data gloves enable direct and straightforward collection of gesture data through a dedicated device and computer program. While many commercial data gloves exist, they are often prohibitively expensive. In ML applications, artificial neural networks (ANNs) are widely used for solving nonlinear regression and classification problems [15, 16]. The development of ANN-based gesture recognition requires a labeled dataset, in which different gesture patterns are tagged with corresponding target outputs during the data collection phase.

The primary contribution of this work is the detailed, step-by-step development of a low-cost data glove suitable for recognizing static gestures based on a Bayesian neural network (BNN). The motivation for using a BNN to recognize static hand gestures lies in its inherent ability to quantify prediction uncertainty alongside classification accuracy. In real-world human–machine interaction scenarios, sensor noise, variations in hand posture, and environmental factors such as lighting can introduce ambiguity in gesture recognition. A BNN not only learns the mapping between gesture features and their corresponding classes but also provides probabilistic confidence estimates, enabling the system to identify uncertain predictions and handle them appropriately. This capability is crucial for safety-critical or assistive applications, where incorrect gesture interpretation could lead to undesirable outcomes. Additionally, BNNs offer improved generalization in limited-data settings by incorporating prior knowledge into the learning process, which is particularly beneficial for static gesture datasets with small sample sizes.

The remainder of this paper is organized as follows. Section 2 introduces the Bayesian neural network (BNN) model designed for multi-class classification tasks. In this section, the fundamental concepts of the Bayesian framework are presented, with particular

p-ISSN 3093-3285
e-ISSN 3093-3315
https://doi.org/10.51316/jst.187.ssad.2026.36.1.3
Received: Mar 18, 2025; Revised: Aug 19, 2025;
Accepted: Sep 8, 2025; Online: Dec 20, 2025.

emphasis on its role in neural network regularization. The approach enables the automatic and efficient determination of optimal regularization parameters through Bayesian inference, thereby reducing the need for manual tuning and mitigating overfitting. Section 3 describes in detail the design and development of the proposed low-cost data glove, including its hardware architecture, data acquisition process, and the implementation of static hand gesture recognition using the proposed BNN. Section 4 concludes the paper by summarizing the main findings, highlighting the practical implications of the work, and suggesting potential directions for future research.

2. Bayesian Neural Network for Classification

2.1. Multi-Layer Perceptron Neural Networks

As BNNs are an extended version of conventional MLP neural networks, this part provides the theory of MLP neural networks [15, 16]. An MLP neural network takes in a vector of real inputs, x_i, and from them compute one or more values of the output layer, $z_k(x, w)$. With a one hidden layer network, as shown in Fig. 1, the value of the k-th output is computed as follows:

$$z_k(x,w) = f_0\left(b_k + \sum_{j=1}^{M} w_{kj} \tanh\left(\bar{b}_j + \sum_{i=1}^{d} \bar{w}_{ji} x_i\right)\right) \quad (1)$$

where $\bar{w}_{ji}$ is the weight on the connection from input unit i to hidden unit j; similarly, w_{kj} is the weight on the connection from hidden unit j to output unit k. $\bar{b}_j$ and b_k are the biases of the hidden and output units; f_0 is the output layer activation function.

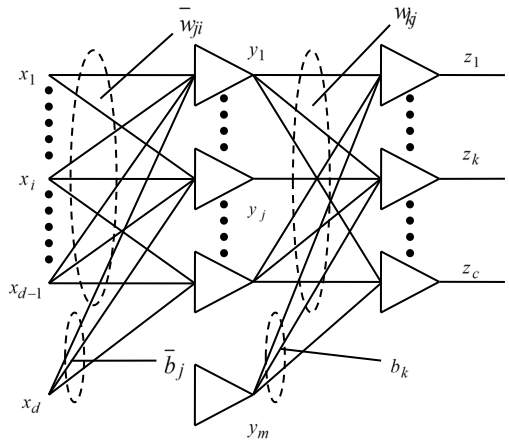

Fig. 1. MLP neural network

MLP neural networks can be used to define probabilistic models for regression and classification tasks by using the network outputs to define the conditional distribution for one or more targets. In c - class classification problems, the target variables are discrete class labels indicating one of c possible classes. The "softmax" model can be used to define the conditional probabilities of the various classes of a network with c output units as follows:

$$z_k(x) = \frac{\exp(a_k(x))}{\sum_{k'=1}^{c} \exp(a_{k'}(x))} \quad (2)$$

For multi-class classification problem, the "entropy" error function with N sample patterns should be used and have the following form:

$$E_D = -\sum_{n=1}^{N}\sum_{k=1}^{c} t_k^n \ln z_k^n \quad (3)$$

where z_k^n is the k-th output corresponding to the n-th pattern. t_k^n is the k-th target corresponding to the n-th pattern.

2.2. Network Regularization

In MLP neural networks, appropriate regularization should be used to prevent any weights and biases from becoming too large because large weights and biases may give poor generalization for new cases. Therefore, a weight decay term is added to the data error function, E_D, to penalize large weights and biases. Specifically, for classification problems, a cost function is defined as follows:

$$S(w) = E_D + \sum_{g=1}^{G} \xi_g E_{W_g} \quad (4)$$

where $S(w)$ is known as a cost function, ξ_g is a non-negative parameter for the distribution of network parameters such as weights and biases. ξ_g is also known as weight decay parameters, also known as hyperparameters. Finally, E_{W_g} is the weight error for the g-th group of weights and biases, and G is the number of groups of weights and biases in the neural network. E_{W_g} has the following form:

$$E_{W_g} = \frac{1}{2}\|w_g\|^2 \qquad g = 1,...,G \quad (5)$$

2.3. Bayesian Inference

Adaptive parameters of an MLP neural network, including weights and biases, can be conveniently grouped into a single W-dimensional weight vector, w According to the Bayesian inference, the posterior distribution of the vector of weights and biases given a data set D can be expressed as follows:

$$p(w \mid D, \psi) = \frac{p(D \mid w, \psi) p(w \mid \psi)}{p(D \mid \psi)} \quad (6)$$

where $\psi = \{\xi_1,...,\xi_G\}$ and (6) is the first level of the inference. $p(w \mid \psi)$ is the weight prior determined using the theory of prior. The prior distribution of the weights is given by:

$$p(w \mid \psi) = \frac{1}{Z_W(\psi)} \exp\left(-\sum_{g=1}^{G} \xi_g E_{W_g} \right) \quad (7)$$

$$Z_W(\psi) = \prod_{g=1}^{G} \left(\frac{2\pi}{\xi_g} \right)^{W_g/2} \quad (8)$$

where $Z_W(\psi)$ is the normalization constant and W_g is the number of weights and biases in the g-th group. $p(D \mid w, \psi)$ is the dataset likelihood and $p(D \mid \psi)$ is the evidence for ψ or the normalization factor. If the dataset is identically independently distributed, the dataset likelihood is:

$$p(D \mid w, \psi) = \exp(-E_D) \quad (9)$$

Assuming that the cost function $S(w)$ has a single minimum at the most probable weight vector w_{MP} and $S(w)$ can locally be approximated as a quadratic form obtained by the second-order Taylor series expansion of $S(w)$ as follows:

$$S(w) \approx S(w_{MP}) + \frac{1}{2}(w - w_{MP})^T A (w - w_{MP}) \quad (10)$$

In (10), A is the Hessian matrix of the cost function at w_{MP} and is given by:

$$A = \nabla\nabla S(w_{MP}) = H + \sum_{g=1}^{G} \xi_g I_g \quad (11)$$

where $H = \nabla\nabla E_D(w_{MP})$ is the Hessian matrix of the data error function at w_{MP} and I_g is the diagonal matrix having ones along the diagonal that picks off weights in the g-th group. After the training phase, the posterior distribution of weights can be derived as:

$$p(w \mid D, \psi) = \frac{1}{Z_S(\psi)} \exp(-S(w)) \quad (12)$$

where $Z_S(\psi)$ is the normalization constant for the approximating Gaussian, and is therefore given by:

$$Z_S(\psi) \approx \exp(-S(w_{MP}))(2\pi)^{W/2} (\det A)^{-1/2} \quad (13)$$

Again, using the Bayes' theorem, the posterior distribution of the weight decay parameters can be expressed as follows:

$$p(\psi \mid D) = \frac{p(D \mid \psi) p(\psi)}{p(D)} \equiv p(D \mid \psi) p(\psi) \quad (14)$$

where $p(\psi)$ is the prior distribution of the weight decay parameters, and we simply assume that this distribution is uniform. However, this will be ignored subsequently, because to infer the value of the weight decay parameters, we only need to find the values of the weight decay parameters to maximize $p(D \mid \psi)$. Rearranging (6) gives:

$$p(D \mid \psi) = \frac{p(D \mid w, \psi) p(w \mid \psi)}{p(w \mid D, \psi)} \quad (15)$$

Since all the terms on the right side of (15) are determined from (7), (9), and (12), (15) yields:

$$p(D \mid \psi) = \frac{Z_S(\psi)}{Z_W(\psi)} \quad (16)$$

Taking the derivative of $\ln p(D \mid \psi)$ with respect to ξ_g gives:

$$\frac{\partial}{\partial \xi_g} \ln p(D \mid \psi) = \frac{W_g}{2\xi_g} - E_{W_g} - \frac{1}{2} tr\left(A^{-1}\right) I_g \quad (17)$$

Let this derivative be zero, we can determine ξ_g as follows:

$$2\xi_g E_{W_g} = W_g - \xi_g tr\left(A^{-1}\right) I_g \quad (18)$$

The right-hand side of (18) is equal to a value γ_g defined as follows:

$$\gamma_g = W_g - \xi_g tr\left(A^{-1}\right) I_g \quad (19)$$

γ_g is called the number of "well-determined" parameters in weight group g. Substituting (19) into (18) and rearranging (18) give:

$$\xi_g = \frac{\gamma_g}{2E_{W_g}} \quad (20)$$

The terms ξ_g and γ_g can be later used with several formulas to compute the logarithm of the evidence in Bayesian model comparison. The optimal model is selected corresponding to the highest logarithm of the evidence.

2.4. Scaled Conjugate Gradient Algorithm

The main problem when training MLP neural networks using the conventional gradient descent method is that values for the learning rate and momentum must be chosen appropriately. As this procedure is inefficient, this section focuses on the conjugate gradient algorithm, which is a fast-training algorithm automatically determining the search direction and step size.

The conjugate gradient algorithm starts by searching in the negative gradient on the first iteration. At the

m-th step, a line search is performed to find the step size α_m as below:

$$\alpha_m = \frac{g_m^T d_m}{d_m^T A d_m} \tag{21}$$

where g_m and d_m are the gradient and search direction at the m-th step.

The new search direction is then given by:

$$d_{m+1} = -g_{m+1} + \beta_m d_m \tag{22}$$

where β_m can be determined using the Polak-Rebiere formula as follows:

$$\beta_m = \frac{(g_{m+1} - g_m)^T g_{m+1}}{g_{m+1}^T g_m} \tag{23}$$

In the conjugate gradient algorithm, as the cost function $S(w)$ is not a quadratic form, then the Hessian matrix A may not be positive definite and in this case, the parameter update formula (21) may increase the function value. This can be overcome by adding a non-negative multiple λ_m of the unit matrix to the Hessian A to obtain $A + \lambda_m I$. The expression (21) becomes:

$$\alpha_m = \frac{g_m^T d_m}{d_m^T A d_m + \lambda_m \|d_m\|^2} \tag{24}$$

The denominator of (24) can be written as:

$$\delta_m = d_m^T A d_m + \lambda_m \|d_m\|^2 \tag{25}$$

If δ_m is negative, we can increase the value of λ_m to make δ_m is positive. Let the raised value of λ_m be $\overline{\lambda}_m$, then the corresponding raised value of δ_m is given by:

$$\overline{\delta}_m = \delta_m + (\overline{\lambda}_m - \lambda_m)\|d_m\|^2 \tag{26}$$

To $\overline{\delta}_m$ is greater than zero, $\overline{\lambda}_m$ can be chosen as follows [14]:

$$\overline{\lambda}_m = 2\left(\lambda_m - \frac{\delta_m}{\|d_m\|^2}\right) \tag{27}$$

Substituting (27) into (26) gives:

$$\overline{\delta}_m = -\delta_m + \lambda_m \|d_m\|^2 = -d_m^T A d_m \tag{28}$$

This value is positive and used as the denominator in (24) to compute the step-size α_m. To find λ_{m+1}, a comparison parameter is first defined as:

$$\Delta_m = \frac{2[S(w_m) - S(w_m + \alpha_m d_m)]}{\alpha_m d_m^T d_m} \tag{29}$$

Then the value of λ_{m+1} can be adjusted using the following prescriptions:

- If $\Delta_m > 0.75$, then $\lambda_{m+1} = \lambda_m / 2$
- If $0.25 < \Delta_m < 0.75$, then $\lambda_{m+1} = \lambda_m$
- If $\Delta_m < 0.25$, then $\lambda_{m+1} = 4\lambda_m$
- If $\Delta_m < 0$, then $\lambda_{m+1} = 4\lambda_m$ and take no step

3. Static Hand Gesture Recognition Using A Low-Cost Data Glove

This section provides a comprehensive explanation of the methodology used for recognizing static hand gestures through the integration of a low-cost data glove and a BNN. The data glove functions as the primary sensing interface, equipped with multiple flex sensors strategically positioned along the fingers. These sensors continuously measure the degree of finger bending and hand configuration, thereby capturing the spatial characteristics of each static gesture. The analog signals produced by the flex sensors are converted into digital data and preprocessed to eliminate noise, normalize variations, and extract relevant features representing the user's hand posture.

The processed sensor data are then supplied as input vectors to BNN, which serves as the classification engine of the system. The BNN utilizes probabilistic inference to evaluate the likelihood of each gesture class given the input data, effectively modeling uncertainties that may arise from sensor inaccuracies, user differences, or environmental disturbances. Through this probabilistic learning framework, the BNN can distinguish subtle differences among hand postures and assign each input pattern to its corresponding predefined gesture category.

3.1. The Data Glove

Static hand gestures can be captured using a low-cost data glove, as depicted in Fig. 2.

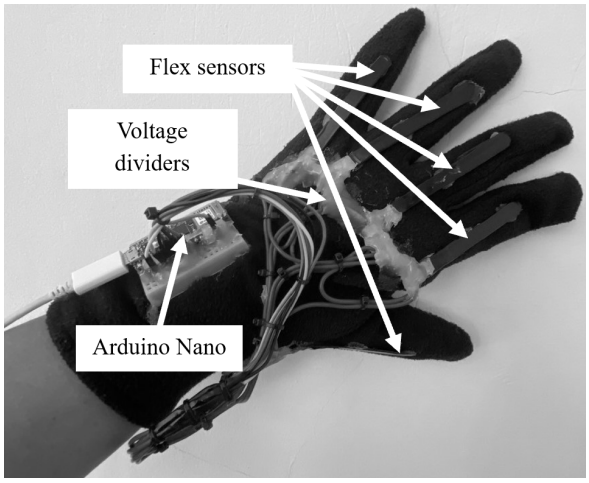

Fig. 2. A low-cost data glove using five flex sensors and the Arduino Nano

This glove is constructed from readily available components, making it an economical and practical tool for gesture recognition. It consists of the following main elements:

- *A standard glove* acts as the foundation for attaching electronic components while ensuring user comfort and hand flexibility.
- *Five flex (bend) sensors* are used in the system, each functioning as a variable resistor whose resistance increases in proportion to the degree of bending. These sensors are strategically positioned along the length of each finger to accurately capture the extent of finger flexion. When the user performs static hand gestures, bending at the finger joints causes the corresponding sensor's conductive layer to stretch, thereby increasing its electrical resistance. This resistance change is converted into a varying voltage through a voltage divider circuit, producing an analog signal that reflects the finger's bending angle. The collection of these signals from all five sensors forms a gesture-specific signature, which can then be processed and classified by the recognition algorithm.
- *Five voltage divider circuits* are implemented, one for each flex sensor, to convert the sensors' resistance changes into measurable voltage signals that can be read by the processing unit. In each voltage divider, the flex sensor is connected in series with a fixed resistor, and the output is taken from the junction between them. As a finger bends, the corresponding flex sensor's resistance increases, altering the voltage ratio across the divider. This change results in a rise in the output voltage, which is directly proportional to the degree of bending within the sensor's operating range. By continuously monitoring the voltages from all five dividers, the system can capture a real-time electrical profile of the user's hand posture. These voltage signals are then digitized by an analog-to-digital converter (ADC) and sent to the gesture recognition algorithm, forming the basis for accurate classification of static hand gestures.
- *An Arduino Nano board* is a compact and breadboard-friendly microcontroller platform based on the ATmega328 chip, serving as the core data acquisition and communication unit in the system. Its role is to continuously collect analog voltage signals from the five flex sensors via its built-in analog-to-digital converter, converting them into digital values for further processing. The board's small physical footprint, low power consumption, and affordability make it particularly well-suited for integration into wearable devices such as data gloves, where space and weight are critical design considerations. In this application, the Arduino Nano not only reads and organizes sensor data but can also perform basic preprocessing as signal filtering or scaling-before transmitting the information to a computer or external processor via USB or serial communication. This efficient combination of sensing, digitization, and transmission capabilities enables seamless real-time hand gesture capture in a portable and cost-effective form factor.

The Arduino Nano board is programmed via the Arduino IO Package for Simulink, enabling direct communication between the hardware and the Simulink environment. In this setup, a Simulink model acquires five analog voltage signals from the voltage divider circuits, each corresponding to the bending level of a flex sensor. Arduino Nano's 10-bit analog-to-digital converters read these inputs from pins A0 through A4, producing raw integer values in the range of 0 to 1023. These values are then normalized by dividing by 1023, converting them into a range between 0 and 1 for consistent processing by the BNN. During data collection, each session runs for 5 seconds with a sampling interval of 0.1 seconds, generating approximately 51 samples per gesture. Each sample represents a snapshot of a static hand gesture's flex sensor reading. The complete set of samples forms a labeled training dataset, which is essential for training the BNN to accurately recognize and classify various static hand gestures based on the captured sensor data.

Figure 3 illustrates six distinct static hand gestures employed in the gesture recognition process. These gestures are systematically grouped into five separate classes according to the specific configurations and positions of the fingers and palm. In this classification scheme, Gestures 1 and 2 are categorized under Class 1, while Gestures 3, 4, 5, and 6 correspond to Classes 2, 3, 4, and 5, respectively. This structured classification enables the recognition system to accurately distinguish different hand postures, ensuring that each gesture is uniquely associated with a predefined category.

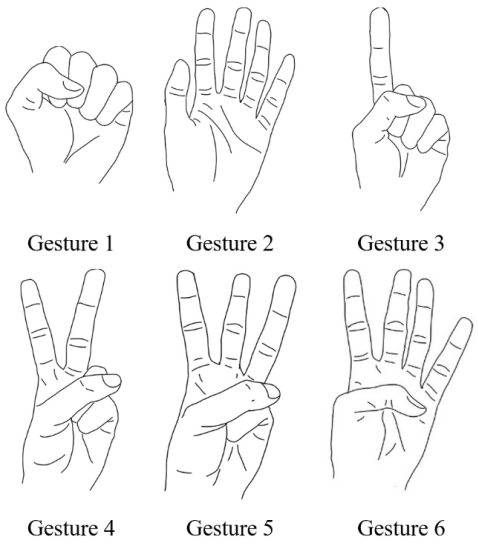

Fig. 3. Six static hand gestures

Table 1. Number of collected static hand gestures.

Gestures	Number of patterns	Class
Gestures 1 and 2	102	1
Gesture 3	51	2
Gesture 4	51	3
Gesture 5	51	4
Gesture 6	51	5

Table 1 shows the distribution of data patterns collected for each gesture class. Each data pattern corresponds to a single measurement recorded by the flex sensors while a specific hand gesture is being performed. These data patterns reflect the distinctive sensor output profiles that represent the degree of finger flexion and the overall hand posture characteristic of each gesture.

3.2. Network Architecture

The BNN used for classifying static hand gestures is designed with a specific architecture to effectively process data collected from flex sensors. This architecture consists of the following components.

- *Input layer:* It contains six nodes, of which five correspond to the analog signals obtained from the five flex sensors embedded in the data glove. Each flex sensor measures the degree of finger flexion, providing continuous data related to the curvature of each finger. The sixth node is an augmented bias input with a constant value of 1, which enhances the model's learning capacity by allowing the network to adjust the decision boundary more flexibly.
- *Hidden layer:* The network includes a hidden layer with six neurons, which process the incoming data from the input layer by first computing a weighted sum of the inputs and then applying a nonlinear activation function. This combination allows the model to capture and represent complex, non-linear relationships between the flex sensor signals and the corresponding hand gestures. The choice of six neurons strikes a balance between model complexity and computational efficiency, ensuring that the network has sufficient capacity to learn gesture-specific patterns while still generalizing well across different gesture classes without overfitting.
- *Output layer:* The BNN has five output nodes, each corresponding to a specific gesture class. The outputs represent a probability distribution over the five gesture classes, allowing the network to account for uncertainty in its predictions.
- *Output layer:* The BNN's output layer consists of five nodes, each representing one of the predefined gesture classes. The outputs form a probability distribution across these classes, indicating the likelihood of each gesture given the input data. This probabilistic representation enables the network to account for uncertainty in its predictions, providing not only the most likely gesture but also a measure of confidence for each classification.

3.3. Network Training Procedure

There are four weight decay parameters ξ_1, ξ_2, ξ_3 and ξ_4 corresponding distributions from the weights between the input nodes to the hidden nodes, the bias input node to the hidden nodes, the hidden nodes to the output nodes, and the bias hidden node to the output nodes. The training procedure was then implemented as follows [15, 16]:

1) The weights and biases were initialized randomly and choosing initial values for the hyperparameters,
2) The network was trained to minimize the cost function $S(w)$ using the SCG algorithm,
3) When the network training reached a local minimum, the values of the hyperparameters were re-estimated as follows:

$$\xi_g^{new} = \frac{\gamma_g^{old}}{2E_{W_g}} \tag{30}$$

4) Steps 2 and 3 were repeated until convergence was achieved (the total error term was smaller than a pre-determined value and did not change significantly in subsequent iterations).

The final values of the weights and biases obtained from training the BNN were subsequently stored to deploy a standalone embedded system for static hand gesture recognition. This deployment was implemented using an Arduino Nano board, allowing the system to function independently without relying on an external computer. By transferring the trained parameters to the Arduino Nano, the neural network's decision-making capabilities were embedded directly into the microcontroller. This enabled real-time processing and classification of static hand gestures, making the system portable and efficient. The optimized weights and biases allowed the Arduino Nano to perform accurate gesture recognition while maintaining low power consumption and a compact form factor, suitable for practical applications in wearable devices, human-computer interaction, and assistive technologies.

4. Conclusion

The study demonstrates the feasibility and effectiveness of integrating low-cost hardware with advanced machine-learning techniques for static hand gesture recognition. The affordability and simplicity of the data glove, combined with the flexibility and power of the BNN, present a scalable and adaptable solution

suitable for both research and real-world applications. This approach provides a cost-effective alternative to more expensive commercial gesture recognition systems while maintaining robust performance. Practical applications for this system extend to various fields, including sign language interpretation, where it could assist individuals with speech or hearing impairments by translating hand gestures into text or speech in real-time. The system could facilitate intuitive, gesture-based command inputs for operating machinery or robotic arms in robotic control. Furthermore, human-computer interaction could be enhanced by enabling hands-free operation of devices through natural hand gestures. Future research may focus on several key areas to further optimize the system. Enhancing the BNN's architecture, such as refining the number of hidden units or exploring alternative activation functions, may improve accuracy and computational efficiency. Additionally, real-time processing capabilities could be enhanced by optimizing the system's code and hardware setup, allowing for faster gesture recognition with minimal latency. Another valuable direction would be to expand the gesture vocabulary, enabling the recognition of a wider array of static and dynamic gestures to accommodate diverse applications. Overall, this study highlights a promising approach to static hand gesture recognition using affordable and accessible technology combined with advanced probabilistic machine learning. With further development and refinement, this system could play a pivotal role in advancing gesture-based interfaces, improving accessibility, and enabling more natural HMI systems.

Acknowledgements

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

References

[1] Derek W. Orbaugh Antillon, Christopher R. Walker, Samuel Rosset, Iain A. Anderson, Glove-based hand gesture recognition for diver communication, IEEE Transactions on Neural Networks and Learning Systems, vol. 34, no. 12, pp. 9874-9886, Dec. 2023, https://doi.org/ 10.1109/TNNLS.2022.3161682.

[2] Mingzhang Pan, Yingzhe Tang, Hongqi Li, State-of-the-art in data gloves: A review of hardware, algorithms, and applications, IEEE Transactions on Instrumentation and Measurement, vol. 72, Feb. 2023, https://doi.org/ 10.1109/TIM.2023.3243614.

[3] Jun Ma, Xunhuan Ren, Hao Li, Wenzu Li, Viktar Yurevich Tsviatkou, Anatoliy Antonovich Boriskevich, Noise-against skeleton extraction framework and application on hand gesture recognition, IEEE Access, vol. 11, pp. 9547-9559, Jan. 2023, https://doi.org/ 10.1109/ACCESS.2023.3240313.

[4] Abdirahman Osman Hashi, Siti Zaiton Mohd Hashim, Azurah Bte Asamah, A systematic review of hand gesture recognition: An Update From 2018 to 2024, IEEE Access, vol. 12, pp. 143599-143626, Jul. 2024, https://doi.org/ 10.1109/ACCESS.2024.3421992.

[5] Lin Guo, Zongxing Lu, Ligang Yao, Human-Machine interaction sensing technology based on hand Gesture recognition: A Review, IEEE Transactions on Human-Machine Systems, vol. 51, no. 4, pp. 300-309, Aug. 2021, https://doi.org/ 10.1109/THMS.2021.3086003.

[6] Yongfeng Dong, Jielong Liu, Wenjie Yan, Dynamic hand gesture recognition based on signals from specialized data glove and deep learning algorithms, IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 2509014-2509014, May, 2021, https://doi.org/ 10.1109/TIM.2021.3077967.

[7] Youngmo Han, A low-cost visual motion data glove as an input device to interpret human hand gestures, IEEE Transactions on Consumer Electronics, vol. 56, no. 2, pp. 501-509, May, 2010, https://doi.org/ 10.1109/TCE.2010.5505962.

[8] Minhyuk Lee, Joonbum Bae, Deep learning based real-time recognition of dynamic finger gestures using a data glove, IEEE Access, vol. 8, pp. 219923-219933, Nov. 2020, https://doi.org/ 10.1109/ACCESS.2020.3039401.

[9] Yuichiro Mori, Masahiko Toyonaga, Data-glove for Japanese sign language training system with gyro-sensor, 2018 Joint 10th SCIS and 19th ISIS, Toyama, Japan, 2018.

[10] Yeongyu Park, Jeongsoo Lee, Joonbum Bae, Development of a wearable sensing glove for measuring the motion of fingers using linear potentiometers and flexible wires, IEEE Transactions on Industrial Informatics, vol. 11, no. 11, pp. 198-206, Dec. 2014, https://doi.org/ 10.1109/TII.2014.2381932.

[11] C. Jose L. Flores, A. E. Gladys Cutipa, R. Lauro Enciso, Application of convolutional neural networks for static hand gestures recognition under different invariant features, In 2017 IEEE XXIV (INTERCON), pp. 1-4.

[12] Amrutnarayan Panigrahi, Jaganath Prasad Mohanty, Ayas Kanta Swain, Kamalakanta Mahapatra, Real-time efficient detection in vision based static hand gesture recognition, 2018 IEEE iSES, Hyderabad, India, pp. 265-268, 2018.

[13] Raj Patel, Jash Dhakad, Kashish Desai, Tanay Gupta, Stevina Correia, Hand gesture recognition system using convolutional neural networks, 2018 4th ICCCA, Greater Noida, India. pp. 1-6, 2018.

[14] Paulo Trigueiros, Fernando Ribeiro, Luís Paulo Reis, A comparison of machine learning algorithms applied to hand gesture recognition, 7th Iberian CISTI, Madrid, Spain, pp. 1-6. IEEE, 2012.

[15] Christopher M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[16] Ian Nabney, NETLAB: Algorithms for Pattern Recognition, Springer, 2002.