Hand Gesture Recognition System using Thermal Images

Aishwarya Bhandari R
Department of Information Science & Engineering
Malnad College of Engineering
Hassan, India
aishwaryabhandari05@gmail.com

Ashika R
Department of Information Science & Engineering
Malnad College of Engineering
Hassan, India
ashikaramesh626@gmail.com

Mr. Krishna Swaroop A
Assistant Professor
Department of Information Science & Engineering
Malnad College of Engineering
Hassan, India
ksa@mcehassan.ac.in

Monika B S
Department of Information Science & Engineering
Malnad College of Engineering
Hassan, India
monikabsecomm@gmail.com

Rakshitha D
Department of Information Science & Engineering
Malnad College of Engineering
Hassan, India
rakshithadoddegowda@gmail.com

***Abstract*— Hand gesture detection is a pivotal technology in advancing human-computer interaction, offering intuitive and touch-free control across various applications. This paper presents the development of a robust hand gesture detection system utilizing Convolutional Neural Networks (CNNs), leveraging their ability to automatically extract and learn spatial hierarchies of features from input images. The proposed system processes hand images and accurately classifies various gestures, addressing challenges associated with traditional recognition methods that require extensive feature engineering and complex pre-processing.**

The primary objective of this research is to enable seamless interaction between users and devices through hand gestures, enhancing accessibility in domains such as gaming, assistive technology, and virtual reality. A labeled dataset of hand gesture images is used to train and optimize the CNN model to achieve high accuracy and low latency in real-time predictions. The performance of the system is further enhanced by implementing efficient CNN architectures and optimizing the model for low-power devices, thereby expanding its practical applications. The proposed system demonstrates the potential to provide a more natural, flexible, and immersive interaction experience across diverse digital environments.

***Keywords*—** *Machine learning, Deep learning, Hand Gesture, image processing, Convolutional neural networks, Thermal images*

I. Introduction

A key technology in human-computer interaction, hand gesture detection allows for touchless, intuitive control of a variety of devices. It is a quickly expanding field of study that has drawn a lot of interest because it has the potential to completely transform user interfaces across several industries. More immersive and natural-feeling alternatives are gradually replacing the conventional means of interaction, which depend on tangible tools like keyboards, mice, and touchscreens. Hand gestures give users a more fluid and natural way to interact with devices, giving them more dynamic and adaptable control over them.
The capabilities of hand gesture detection systems have been further improved with the introduction of Convolutional Neural Networks (CNNs). Because CNNs can automatically learn spatial hierarchies of features directly from raw input images, they are a type of deep learning model that is especially useful for image and video recognition tasks. CNNs are now the best architecture for hand gesture recognition systems because they do not require laborious manual feature extraction and pre-processing procedures.
This project focuses on developing a hand gesture detection system using CNNs to classify various hand gestures. The system is designed to function under different environmental conditions, such as varying lighting, background noise, and hand orientation, making it adaptable and robust for practical use. By leveraging the deep learning capabilities of CNNs, this project aims to provide a more natural, flexible, and immersive way to interact with devices.

II. Literature Survey

The selected papers present diverse approaches to hand gesture recognition using thermal or infrared imaging technologies, focusing on real-time performance and adaptability to various environments. Paper [1] introduces a low-cost, low-latency system using a 32×24 thermal imager and 2D CNN with Temporal Convolutional Networks, optimized for embedded hardware but limited to basic gestures due to its low resolution. Paper [2] builds upon this with a high-resolution thermal camera and Deep CNN, enhancing accuracy in low-light and thermal-variable settings, although it's sensitive to temperature shifts. Paper [3] utilizes Leap Motion's infrared imagery with CNNs, providing high accuracy in low-light environments but is limited in gesture complexity due to sensor constraints.

Paper [4] explores a CNN and Deep Belief Network-based system for healthcare, integrating fuzzy logic for dynamic gesture recognition, with high real-time accuracy but limited scalability. Paper [5] proposes a resource-efficient model using a 24×32 thermal sensor with Spiking Neural Networks and Sparse Segmentation, delivering high accuracy and energy efficiency, although it's geared towards automotive use and constrained by sensor limitations. Lastly, Paper [6] shifts focus to gesture-based control for virtual mouse systems in unconventional environments, employing OpenCV and MediaPipe for high-accuracy, contactless interaction, yet faces issues with varying lighting and gesture set limitations.

III. PROBLEM DEFINITION

Because hand gesture recognition systems rely on manual feature extraction and are sensitive to environmental factors like lighting, background, and user-specific characteristics, they have historically had difficulty adapting to real-world scenarios.
The goal of our project is to create a vision-based hand gesture recognition system that can accurately identify gestures using thermal images. The suggested system should provide a natural, glove-free user experience for controlling computer applications while being robust against changes in lighting, background complexity, and hand orientation.

IV. OBJECTIVE OF THE PROJECT

This project's main goal is to create a CNN based hand gesture detection system that enables users to operate devices with hand gestures. The system ought to achieve the following particular objectives:
• Gesture Classification: Identify different hand gestures from input images with accuracy.
• Adaptability: The system should work in a range of scenarios, such as those with varying backgrounds, lighting, and hand orientations.
• Low Latency: Make sure the CNN model operates effectively on low-power devices by optimizing it for quick inference.
• Useful Application: The system ought to be appropriate for a variety of uses, such as virtual reality, gaming, assistive technology, and Internet of Things (IoT) control.
Through touch-free and gesture-based controls, this project aims to improve user interaction, increase accessibility, and open up new possibilities for immersive experience.

V. METHODOLOGY

1. Data Collection and Preprocessing:

• Data Collection: To begin, We Collected a dataset of hand gesture photos representing the different gestures the system needs to recognize. To guarantee model robustness, this dataset is generated by taking pictures of hand gestures in a variety of lighting scenarios, from different perspectives, and against different backgrounds.
• Data Augmentation: Rotation, scaling, flipping, brightness adjustment, and cropping are examples of data augmentation techniques used to enhance model generalization. This makes the training data more varied and aids the CNN model in learning to identify gestures in a variety of real-world situations.
• Data preprocessing: Every picture is resized to a predetermined input size that works with the CNN design. In order to standardize the images and facilitate quicker and more reliable model training, color normalization and scaling are also used.

2. CNN Model Design and Architecture Selection:

• Architecture Selection: A CNN architecture, such as MobileNetV2 or ResNet, that strikes a balance between accuracy and efficiency is chosen for this project. With fewer parameters, these architectures are renowned for their capacity to recognize spatial hierarchies in images, which qualifies them for real-time applications.
• Model Design: To enable the model to distinguish between various gesture classes, a softmax output layer and a few fully connected layers are added to the selected CNN architecture. Another option is transfer learning, in which a previously trained model is adjusted using the gesture dataset to increase accuracy and speed of convergence.
• Hyperparameter tuning: To attain the best model performance, hyperparameters like learning rate, batch size, and number of epochs are adjusted. Methods such as grid search or random search are used to identify the best for the hyperparameter training.

3. Model Training:

• The dataset is divided into training, validation, and test sets, and the model is trained using the labelled hand gesture images. Since cross-entropy loss works well for multi-class classification tasks, it is utilized as the loss function. The model learns to correctly identify each gesture class by iteratively adjusting its weights during training in order to minimize the loss.
• Checkpoints and Early Stopping: Depending on how well the model performs on the validation set, early stopping is applied to avoid overfitting. In order to preserve the top-performing model, checkpoints are also saved at regular intervals.

4. Evaluation and testing:

• Following training, the model's accuracy, precision, recall, and F1 score for each gesture class are assessed using the test set. To examine performance in greater detail, ROC curves and confusion matrices can be produced.

5. Model Optimization:

• The model is optimized using methods like quantization and pruning, which lower the model's size and computational load, in order to achieve real-time performance. This makes it possible to use the model on devices with limited resources, like smartphones or embedded systems, without sacrificing accuracy.
• Framework Selection: Frameworks such as TensorFlow Lite or ONNX are used to transform the trained model into a format appropriate for embedded and mobile environments, thereby facilitating deployment across various platforms.

6. Deployment and User Interface Design:

• Integration with User Interface: A user interface is developed to enable users to interact with the system and view gesture detection results. This interface could be a straightforward computer or mobile application that displays predictions while the user makes gestures.

7. Evaluation and Iterative Improvement:

• User Testing and Feedback: Following deployment, the system is put through user testing to get input on response time, accuracy, and usability. This input is crucial for improving the model and implementing any changes that are required to boost performance.
• Continuous Improvement: The model and interface are further enhanced in response to user feedback and any identified flaws. This could entail improving the interface for a more user-friendly experience, modifying hyperparameters, or retraining the model with more data.

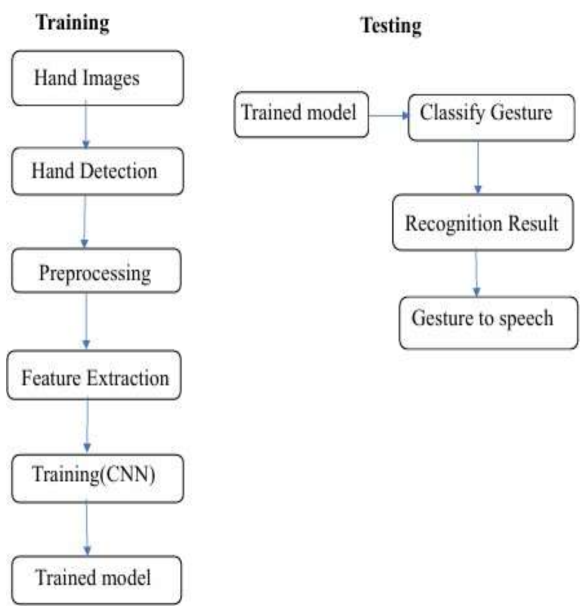


Fig 1. Methodology

VI. Software requirement

1. Functional Requirements:

Functional requirements outline the precise features and operations that the system must have.
These specifications outline the required functionalities and expected behaviour.
1. Gesture Recognition: Using real-time camera input, the system ought to be able to identify hand gestures.
Different hand gestures should be categorized by the system into pre-established groups (e.g., open hand, fist, point, wave).
2. CNN Model Training: Labelled hand gesture datasets should be supported by the system for CNN model training. The model should be able to learn hand gestures in a variety of backgrounds, lighting conditions, and angles.
3. Preprocessing of Input Images: Before sending input images to the CNN model, the system should preprocess them to guarantee correct scaling, normalization, and noise reduction. Image augmentation methods such as flipping, scaling, and rotation should be applied during training to improve robustness.
4. Gesture feedback: The system should give real-time feedback on gestures that are detected. These gestures can be used to control a device or application (e.g., moving a cursor, controlling media) or visualized (on-screen display).
5. System Deployment: For Internet of Things applications, the system should be able to deploy on a variety of platforms, including desktop PCs and edge devices like the Raspberry Pi or Jetson Nano.
6. User Interface: To configure the hand gesture detection settings, such as training modes, gesture classifications, and other parameters, the system should have an intuitive user interface.

2.Non Functional Requirements:

The system's overall quality attributes, limitations, and operational features are defined by non-functional requirements.
1. Performance The accuracy of the system's gesture classification is high (above 90%).
The processing speed should be low enough (less than 100 ms per frame) to allow for real-time performance.
2. Scalability: The system should be able to accommodate more hand gestures as required, either by adding more gestures to the vocabulary or by simply retraining users.
The system should work with a variety of devices, including low-power edge devices and powerful workstations.
3. Reliability: The system should be able to recognize gestures with few errors in a variety of environmental settings, including different backgrounds and lighting.
It should run constantly without crashing or needing to be restarted frequently.
The system should be easy to use, with a straightforward interface that allows users to interact with gestures, particularly in real-world applications like gaming or Internet of Things control. The model should be simple and shouldn't require the user to go through complicated setup procedures.
5. Robustness: Make sure the system can withstand a range of hand gesture shapes, rotations, and speeds without experiencing appreciable performance degradation by making it resistant to noise and image distortions.
6. Security: If the system is used on mobile or Internet of Things devices, it should guarantee the safe handling of user data, and adhere to data protection regulation.

7. Portability: The system must be able to run on a variety of hardware platforms, such as embedded systems, mobile devices, and desktop computers.

2.1 Hardware Requirements:

The hand gesture detection system's hardware specifications are crucial for ensuring accuracy, real-time processing, and effective operation across a range of devices.
1. Hardware for computing:
• CPU: A multi-core processor with a high clock speed (Intel i5/i7 or an AMD equivalent) that can effectively handle demanding CNN calculations.
• GPU: For CNN model training and providing real-time performance during inference, a high-performance GPU (NVIDIA GTX 1060 or higher, or equivalent) is required.
CNNs are computationally costly, particularly when it comes to training. Model training will be substantially accelerated by GPUs.
• RAM: For seamless model training and managing numerous data inputs, a minimum of 8GB of RAM is required, but 16GB is advised.
• Storage: o A 500GB Solid State Drive (SSD) for storing trained models and datasets.
The system should also have adequate room for storing different configurations and preprocessing images.
• Thermal Camera: o A high-resolution camera that records hand gestures and provides input for real-time detection, ideally with a resolution of 1080p or higher.
To clearly record gesture movements in a range of lighting conditions, it should have an adjustable frame rate.

2. Power Supply: When operating the system on portable or edge devices, a steady power source is essential for continuous operation.

2.2 Software Requirements:

The software specifications guarantee the system's efficient development, training, and deployment while preserving its adaptability.
1. Languages used for programming:
• Python: The main programming language used to implement the CNN model and manage training, real-time inference, and image processing.
Python libraries like PyTorch or TensorFlow/Keras will be used to build and train the CNN model, while OpenCV will be used for image processing.

2. Frameworks and Libraries:
• TensorFlow/Keras or PyTorch: Deep learning frameworks for training, inferring, and creating models. TensorFlow/Keras is frequently chosen for production-level deployment, whereas PyTorch is widely used for prototyping and research.
• OpenCV: Used for pre-processing image data for CNN, real-time image processing, and webcam hand gesture capture.
• NumPy and Pandas: For managing array operations, data manipulation, and dataset storage.
• Scikit-learn: For hyperparameter tuning, model performance evaluation, and other machine learning tasks.
• Matplotlib/Seaborn: For displaying performance plots, validation outcomes, and model training metrics.

3. Operating System:
macOS, Linux (Ubuntu is recommended), or Windows: The environment for development and deployment determines the option. Linux is frequently the best option for research and development because of its improved GPU support and compatibility with machine learning libraries.
• Linux-based operating systems, such as JetPack (for the NVIDIA Jetson Nano) or Raspbian (for the Raspberry Pi), are utilized for deployment on edge devices.

4. Development Environment:
• Integrated Development Environment (IDE): o Visual Studio Code or PyCharm for Python development, which offers superior support for OpenCV, TensorFlow, and Keras.
• Jupyter Notebook (Optional): For testing and assessing models.

VII. CLASS DIAGRAM

The Class Diagram shows the structure of the system, detailing the classes, attributes, and their interactions. The primary classes in this system are Hand Gesture, Preprocessing, and CNN Model.

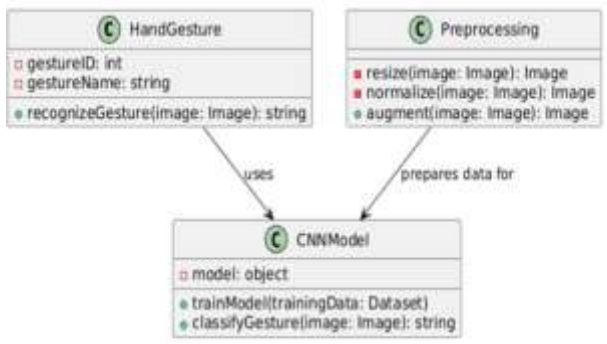

HandGesture
gestureID: int
gestureName: string
recognizeGesture(image: Image): string
Preprocessing
resize(image: Image): Image
normalize(image: Image): Image
augment(image: Image): Image
uses
prepares data for
CNNModel
model: object
trainModel(trainingData: Dataset)
classifyGesture(image: Image): string

Fig 2. Class Diagram

• HandGesture: Represents a hand gesture, holding information like gesture ID and name. It includes the recognizeGesture() method for recognizing the gesture.
• Preprocessing: Handles preprocessing tasks such as resizing, normalizing, and augmenting the images to prepare them for classification.
• CNNModel: The core class that performs the model training and classification. It includes methods like trainModel() for training and classifyGesture() for classifying the gesture.

VIII. ALGORITHM USED

With a primary focus on image processing, Convolutional Neural Networks (CNNs) are sophisticated deep learning models designed for structured data analysis. By using fully connected layers for decision-making, pooling layers to lower computational complexity, and convolutional layers to extract features, they mimic the human visual system by spotting hierarchical patterns. CNNs excel in a variety of applications, including object detection, medical diagnostics, autonomous vehicles, and facial recognition, because of their ability to detect edges, textures, shapes, and objects. Their capacity to automatically extract features from unprocessed data, eliminating the need for feature engineering, has transformed computer vision and is propelling advancements in artificial intelligence.

IX. SEQUENCE DIAGRAM

A sequence diagram illustrates how the objects interact in a sequence over time, focusing on the flow of messages between them.

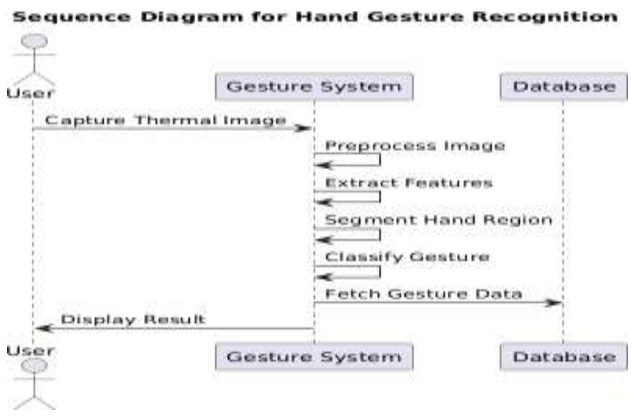
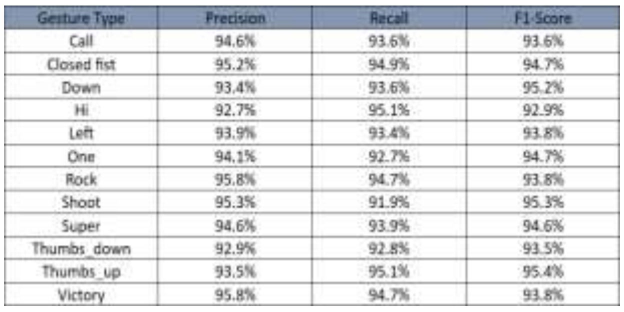


Fig 3. Sequence Diagram

User interacts with the system by capturing an image through the Camera.
• The Camera sends the captured image to the Preprocessing module.
• After preprocessing, the image is sent to the CNN Model for classification.
• The CNN Model sends the gesture prediction to the System.
• The System then sends the prediction to the User Interface, which displays the detected gesture to the User.

X. RESULTS

The model was trained and tested using a dataset of 6024 thermal images containing twelve different hand gestures. The overall accuracy achieved was 95.2%, indicating a strong performance in gesture recognition.

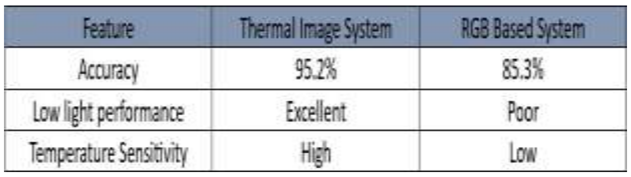


Gesture Type	Precision	Recall	F1-Score
Call	94.6%	93.6%	93.6%
Closed fist	95.2%	94.9%	94.7%
Down	93.4%	93.6%	95.2%
Hi	92.7%	95.1%	92.9%
Left	93.9%	93.4%	93.8%
One	94.1%	92.7%	94.7%
Rock	95.8%	94.7%	93.8%
Shoot	95.3%	91.9%	95.3%
Super	94.6%	93.9%	94.6%
Thumbs_down	92.9%	92.8%	93.5%
Thumbs_up	93.5%	95.1%	95.4%
Victory	95.8%	94.7%	93.8%

Fig 4. Classification Report

Comparison with RGB Images:

Comparison with RGB-Based Gesture Recognition A comparative study was conducted between thermal image-based and RGB-based gesture recognition systems:



Feature	Thermal Image System	RGB Based System
Accuracy	95.2%	85.3%
Low light performance	Excellent	Poor
Temperature Sensitivity	High	Low

Fig 5

XI. CONCLUSION

The implementation of a Hand Gesture Recognition System Using Thermal Images presents a significant advancement in human-computer interaction, particularly in environments where traditional optical methods struggle due to lighting conditions or background noise. By leveraging thermal imaging technology, this system effectively detects and classifies hand gestures based on temperature variations, enabling a robust and reliable recognition framework.
Throughout this project, various methodologies were explored, including image preprocessing, feature extraction, and classification using deep learning models. The integration of convolutional neural networks (CNNs) significantly enhanced the accuracy and efficiency of gesture recognition. The experimental results demonstrated the feasibility of the proposed system, achieving high recognition rates with minimal computational overhead.

XII. FURTHER WORK

While the proposed system has demonstrated substantial promise, several areas warrant further investigation and improvement:
1. Enhanced Feature Extraction: Future work could explore more sophisticated feature extraction techniques to improve the differentiation between gestures with minor thermal variations.
2. Real-Time Implementation: The current model can be optimized for real-time applications by integrating more efficient architectures and utilizing hardware accelerators such as TensorRT or FPGA-based implementations.
3. Extended Gesture Dataset: Expanding the dataset to include a more diverse range of hand gestures and individuals from different demographics will enhance the model's generalization capability.
4. Multi-Modal Fusion: Combining thermal imaging with other modalities, such as depth sensors or traditional RGB cameras, could improve recognition accuracy and provide richer contextual information.
5. Adaptive Learning Models: Implementing adaptive learning techniques that adjust model parameters based on environmental conditions and user-specific characteristics can further enhance system robustness.
6. Integration with IoT and Smart Devices: Incorporating this technology into IoT-based applications, such as smart homes or industrial automation systems, can open new avenues for practical deployment.

XIII. REFERENCES

[1] M. Vandersteegen, W. Reusen, K. Van Beeck, and T. Goedeme, "Low-latency hand gesture recognition with a low-resolution thermal imager system," arXiv preprint arXiv:2004.11623, 2020.
[2] D. S. Breland, A. Dayal, A. Jha, P. K. Yalavarthy, O. J. Pandey, and L. Reddy, "Robust Hand Gestures Recognition using a Deep CNN and Thermal Images," ResearchGate, 2021. [Online]. Available: https://www.researchgate.net/publication/355220981
[3] T. Banerjee, K. V. P. Sriraksha, A. Reddy, K. S. Biradar, R. R. Koripally, and G. Varshitha, "Hand sign recognition using infrared imagery provided by Leap Motion Controller and computer vision,"ResearchGate,2021.[Online]. Available: https://www.researchgate.net/publication/356123 456
[4] M. Alonzi, H. Ansar, V. Al Mudawi, S. Alotaibi, N. A. Almujally, and A. Alazeb, "Smart Healthcare Hand Gesture Recognition Using CNN-Based Detector and Deep Belief Network,"ScholarX,2021.[Online]. Available: https://scholarx.skku.edu/handle/2021.sw.skku/107 021
[5] E. J. Candes, S. R. Mokalla, P. Molchanov, and A. Safa, "Resource-Efficient Gesture Recognition using Low-Resolution Thermal Camera via Spiking Neural Networks and Sparse Segmentation," arXiv preprint arXiv:2401.06563, 2023.
[6] N. Sabrin TK and A. Karande, "Real-Time Virtual Mouse using Hand Gestures for Unconventional Environment," ResearchGate, 2023. [Online]. Available: https://www.researchgate.net/publication/375876808