HAND GESTURE DETECTION USING TRANSFER LEARNING

a project report submitted by

ISHAN CHASKAR (REG. NO: URK21CS1181)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

under the supervision of

Dr. P . SANTHIYA

DIVISION OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed-to-be-University under Sec-3 of the UGC Act, 1956)
Karunya Nagar, Coimbatore - 641 114, India.

APRIL 2025

BONAFIDE CERTIFICATE

Certified that this project report **“HAND GESTURE DETECTION USING TRANSFER LEARNING”** is the bonafide work of “ISHAN CHASKAR (REG. NO: URK21CS1181)**”** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. J. Immanuel Johnraja	**Dr. P. Santhiya**
Head of the Division Division of Computer Science and Engineering	**Supervisor** Assistant Professor Division of Computer Science and Engineering

Submitted for the Project Viva Voce held on……………………….

Examiner

ACKNOWLEDGEMENT

First and foremost, we praise and thank **ALMIGHTY GOD** for giving us the will power and confidence to carry out our project.

We are grateful to our beloved founders Late **Dr. D.G.S. Dhinakaran**, **C.A.I.I.B, Ph.D.,** and **Dr. Paul Dhinakaran**, **M.B.A., Ph.D.,** for their love and always remembering us in their prayers.

We extend our thanks to **Dr. G. Prince Arulraj, Ph.D.,** Vice chancellor, **Dr. E. J. James, Ph.D.,** and **Dr. Ridling Margaret Waller, Ph.D., Dr. R. Elijah Blessing, Ph.D.,** Pro-Vice Chancellor(s) and **Dr. S. J. Vijay, Ph.D.,** Registrar for giving us the opportunity to carry out this project.

I would like to place my heart-felt thanks and gratitude to **Dr. J. Immanuel Johnraja, Ph.D.,** HOD, and **Dr. Jaspher W Kathrine, Ph.D.,** Deputy Head, Department of Computer Science and Engineering for their encouragement and guidance.

We are grateful to our guide, **Dr. P. SANTHIYA,** Division of Computer Science and Engineering for his valuable support, advice and encouragement.

We also thank all the staff members of the Division for extending their helping hands to make this project work a success.

We would also like to thank all my friends and my parents who have prayed and helped me during the project work

ABSTRACT

This paper endeavors to tackle the intricate challenges posed by cognitive processing, particularly in high-stakes domains like videography, remote surgery, and sign interpretation. Within these realms, the precision of information processing is paramount, as even subtle variations in gestures, such as finger positioning or hand movements, can have profound implications for task outcomes. Thus, the quest for innovative methodologies that bolster cognitive performance becomes imperative, especially in tasks characterized by the need for segmentation, inference, and interconnection of complex information.

Our study introduces a pioneering approach that harnesses the synergy between transfer learning and deep learning models to address these challenges. Transfer learning, a cornerstone technique in machine learning, offers a powerful paradigm for knowledge transfer from one task to another, leveraging pre-existing knowledge to expedite learning and improve performance. By employing pre-trained deep learning models like EfficientNet, we tap into the rich representations of intricate features learned from vast datasets, thereby enhancing the model's capacity to discern subtle patterns and nuances in hand gestures.

Central to our methodology is the meticulous analysis of diverse datasets containing hand gestures captured by the individual through MediaPipe. These datasets are carefully curated to encompass a broad spectrum of gestures, encompassing variations in hand orientation, speed, and context. Through rigorous experimentation and comprehensive evaluation, we demonstrate the robustness and efficacy of our approach in accurately classifying hand gestures with a remarkable degree of precision.

An inherent advantage of our proposed method lies in its adaptability and scalability across diverse scenarios and environments. The flexibility afforded by transfer learning enables our system to seamlessly adapt to new contexts and generalize effectively to previously unseen data, making it well-suited for deployment in real-world settings where environmental conditions may vary unpredictably.

Furthermore, our research contributes significantly to advancing the frontier of deep learning applications in cognitive tasks. By showcasing the effectiveness of transfer learning in gesture

recognition, we offer valuable insights into the potential of these methodologies to address complex cognitive challenges across a wide array of domains.

In conclusion, our findings underscore the transformative potential of harnessing transfer learning and deep learning models to augment cognitive performance in challenging real-world scenarios. By providing a robust framework for accurate gesture recognition, our approach not only advances the field of cognitive processing but also lays the groundwork for further innovation and exploration in this burgeoning field.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

S.No	Symbols and Abbreviations	Definitions
1	ML	Machine Learning
2	AI	Artificial Intelligence
3	CNN	Convolutional Nueral Network
4	VGG	Visual Geometry Group
5	ASL	American Sign Language
6	SGD	Stochastic Gradient Descent
7	ReLU	Rectified Linear Unit
8	DeIT	Data-Efficient Image Transformers
9	CV	Computer Vision
10	GPU	Graphics Processing Unit

CHAPTER 1

INTRODUCTION

1.1 Introduction

Nonverbal communication holds significant importance in our daily lives, conveying approximately 65% of messages, a substantial proportion compared to verbal communication, which contributes no more than 35% to our interactions [1] Lately, the advancements in the human-computer interaction (HCI) innovation has led to great interest in gesture recognition. Among the various types of interactions, cognitive navigation stands out as the most unique and effective way for users to communicate with machines. Computers can understand and respond to human commands by interpreting gestures; thus, it can pave the way for more seamless and immersive user experiences in a variety of applications such as virtual reality, augmented reality, robots and smart devices.

Motion detection will extract important information from videos or image sequences depicting human movements. This process consists of several important steps, including hand segmentation, feature extraction, and hand gesture classification. Using modern machine learning and machine vision, researchers and developers create powerful and useful knowledge-based systems that can understand many things with precision and confidence.

In recent years, deep learning models have become powerful tools for solving knowledge-based problems, including knowledge guidance. Characterized by their ability to learn hierarchical representations of data, these models have demonstrated excellent performance in many areas and outperformed machine learning methods in many areas. Existing reviews predominantly focus solely on the cell's localized configuration. These given systems typically utilize segmented hand regions as the input or execute a preprocessing models or patterned/colored gloves [2] – [6].

Gesture recognition is a big problem for many reasons. First, the system must work well for devices that are often different from those encountered during development. These changes could include surround sound, different signatures, different languages, and more. In general, constraints are limited to the beacon environment to reduce problems during segmentation and tracking.

Traditionally, hand gesture recognition relies on a variety of techniques to extract hand gestures, often relying on modeling techniques such as Hidden Markov Models (HMM). However, recent work on deep learning in the field of image classification, object recognition or classification, speech identification, and human performance acknowledgement [7]-[10] has led researchers to explore its application in gesture recognition. For example, convolutional neural networks (CNN) are commonly utilized to learn features in computer vision.

In contrast, three dimensional convolutional neural networks (3DCNN) have been utilized in video models and represent an additional variant of the CNN model that includes space- time filters. This model has been previously examined for spatiotemporal representation in various video analyses, as shown in previous studies [11]-[16] In this research paper, we try to find out the application of state-of-the-art deep learning models, including EfficientNet, NASNet, AlexNet, and convolutional neural networks (CNN), to the action recognition task.

1.2 Objectives

The primary aim of the hand gesture recognition model utilizing transfer learning is to create a versatile and robust system capable of accurately interpreting and understanding human hand movements across various contexts and applications. By leveraging transfer learning techniques, the model aims to build upon existing knowledge from pre-trained neural networks, facilitating efficient adaptation to new gesture recognition tasks with minimal data requirements.

Some of the other key objectives are:

- Fine-grained Gesture Recognition: Go beyond basic gesture recognition by enabling the model to identify intricate differences between similar hand gestures. This fine-grained capability is essential in advanced scenarios such as AR/VR environments, where even minor finger movements can represent different commands or interactions. By incorporating high-resolution feature extraction and leveraging attention mechanisms, the system becomes capable of distinguishing between gestures that appear visually similar but carry different meanings, enhancing the fidelity and richness of user interactions.
- High Accuracy: Achieving high precision is critical to ensure the reliability of the gesture recognition system, especially in applications like sign language interpretation or robotic control. The model should consistently achieve above 90% accuracy on validation datasets, minimizing misclassifications. This can be accomplished by implementing ensemble

learning techniques, advanced data augmentation, and hyperparameter optimization. Consistent benchmarking against industry-standard datasets and cross-validation across different environments will further help validate the robustness of the model.

- Real-time Performance: The system must deliver instant recognition without lag to be effective in real-time use cases. This involves optimizing the model architecture for speed, using techniques like lightweight convolutional layers, early exit strategies, and parallel computation on GPUs or edge devices. Fast preprocessing pipelines and asynchronous input handling should be implemented to maintain a smooth user experience. Achieving a balance between latency and accuracy ensures the system is responsive while remaining reliable.

- Generalization: The model should perform well across a wide variety of users, regardless of variations in hand size, finger length, skin tone, or lighting conditions. To achieve this, the training dataset should be inclusive and diverse, and the model should be evaluated under different environmental settings. Techniques such as domain adaptation, adversarial training, and normalization layers that are insensitive to contrast or brightness can significantly improve the generalization ability, making the system more robust for global deployment.

- Reduced Computational Cost To allow deployment on low-power devices such as smartphones, embedded systems, or wearables, the model must be efficient in terms of memory and processing requirements. Techniques like model pruning, weight quantization, and knowledge distillation can significantly reduce model size and computational demand without substantially affecting accuracy. EfficientNet variants and transformer optimizations (like DeiT-Tiny or MobileViT) are ideal choices when aiming to deploy models in real-time, on-the-go applications.
- Interpretability: It is important that the model's decision-making process is transparent, especially in sensitive applications like assistive devices for hearing-impaired or gesture-based authentication. Incorporating visualization tools like Grad-CAM, saliency maps, or attention overlays can help users and developers understand which features the model is

focusing on during classification. Clear explanations and visual justifications improve trust, support debugging, and provide insight into possible failure modes, making the technology more acceptable and user-friendly.

1.3 Motivation and Application

The motivation behind developing a hand gesture recognition model using transfer learning stems from a myriad of compelling factors spanning technological innovation, human-computer interaction, inclusivity, and societal advancement. At its core, this endeavor is driven by a vision to bridge the gap between human cognition and machine understanding, leveraging the innate expressiveness and universality of hand gestures as a means of intuitive communication.

Hand gestures, as a non-verbal form of communication, are deeply embedded in human behavior and can transcend language barriers. By interpreting these gestures through intelligent systems, we open new channels of interaction, particularly for individuals who rely on sign language for daily communication. For the deaf and hard-of-hearing community, a robust and responsive gesture recognition model enables seamless dialogue with hearing individuals, machines, and digital platforms—an achievement that can drastically improve accessibility and foster inclusive environments.

The use of transfer learning in this context is particularly advantageous, as it allows developers to utilize pre-trained deep learning models that have already learned general image features. This approach significantly reduces the amount of data and computational resources needed to build an accurate recognition system. By fine-tuning these models with gesture-specific datasets, the development process becomes more efficient, while still achieving high levels of precision and responsiveness.

Beyond accessibility, hand gesture recognition has vast applications in emerging fields such as augmented reality (AR), virtual reality (VR), robotics, and smart homes. In AR/VR interfaces, gestures can replace traditional input devices, offering a more immersive and natural user experience. In robotics, gesture commands can be used for controlling robotic arms or autonomous systems, especially in scenarios where voice commands are impractical. In smart homes, simple gestures can be used to control lighting, appliances, or entertainment systems, providing hands-free convenience for all users, including those with limited mobility.

Furthermore, this technology plays a crucial role in advancing human-computer interaction (HCI), where the goal is to make machines understand and respond to human intent in the most seamless way possible. Gesture recognition provides a touchless interface, which is especially relevant in post-pandemic environments where contactless technology is in demand. Public kiosks, ATMs, and interactive displays can incorporate gesture-based interfaces to enhance hygiene and user experience.

From a societal perspective, the development of gesture recognition systems represents a step toward equitable access to digital infrastructure. It helps eliminate communication barriers, promotes digital literacy, and supports the integration of differently-abled individuals into the mainstream workforce and education systems. The scalability of such technology means it can be deployed in rural areas, educational institutions, healthcare settings, and more, where professional sign language interpreters may not be readily available.

In summary, the drive to develop a hand gesture recognition model is not solely a technical challenge but a multidimensional mission that aligns with the broader goals of inclusivity, user-centric design, and the ethical advancement of artificial intelligence. Transfer learning empowers this mission by making sophisticated models more accessible and adaptable, turning vision into practical, real-world solutions that can change lives and shape the future of interaction.

• Natural Human-Computer Interaction

Hand gestures are a fundamental and natural form of communication for humans, often used in conjunction with verbal language to convey meaning. By enabling computers to recognize and interpret hand gestures, we aim to create more seamless and intuitive interfaces, allowing users to interact with technology in a way that mirrors how they communicate in everyday life. This could revolutionize applications across industries, including gaming, virtual reality, robotics, and even medical technology, where users can control devices, input commands, or interact with environments using natural gestures.

• Accessibility and Inclusivity

Hand gesture recognition technology has the potential to empower individuals with disabilities, particularly those who are deaf, hard of hearing, or have limited mobility. For instance, individuals who rely on sign language for communication can benefit from systems that translate sign language gestures into text or speech in real-time, breaking down language barriers and improving communication efficiency.

• Advancements in Assistive Technologies

As gesture recognition models evolve, they have the potential to significantly improve the quality of life for individuals with physical and communication impairments. For example, individuals with motor disabilities can use hand gestures to control prosthetic limbs or assistive devices, offering them a greater degree of autonomy and independence.

• Cultural and Linguistic Preservation

Hand gestures are not just a form of communication but are deeply embedded in cultural traditions and linguistic structures. In many cultures, hand gestures serve as a primary mode of expression, and some languages (such as sign languages) rely exclusively on gestures for communication. By developing gesture recognition models capable of accurately interpreting and translating culturally specific gestures, we can contribute to the preservation of these languages and cultural practices. Moreover, these models could foster better cross-cultural communication by enabling people from different linguistic and cultural backgrounds to communicate effortlessly.

• Technological Innovation and Research

The field of hand gesture recognition represents a rapidly growing area of research and development, with applications extending far beyond the boundaries of traditional user interfaces. By exploring the capabilities of deep learning and transfer learning, the model can contribute to the development of cutting-edge solutions that push the boundaries of AI and machine learning.

• Future Prospects

Looking ahead, the future of hand gesture recognition is promising, with potential breakthroughs in accuracy, speed, and real-time applications. With the continuous evolution of AI and deep learning models, we expect these systems to become more precise, capable of recognizing subtle gestures and complex hand movements.

1.4 Overview of the Project

The hand gesture recognition project using transfer learning is a multifaceted endeavor aimed at developing a versatile system capable of accurately interpreting and understanding human hand movements across diverse applications. Motivated by the desire to bridge the gap between human cognition and machine understanding this project seeks to leverage transfer learning techniques to adapt pre-existing knowledge to the task of gesture recognition. The goal is to enhance the system's accuracy and efficiency, making it applicable in various real-world scenarios.

• Transfer Learning

Transfer learning plays a crucial role in this project by utilizing pre-trained neural networks, such as EfficientNet or NASNet, which have already learned useful feature representations from vast image datasets like ImageNet. By fine-tuning these pre-trained models on gesture-specific datasets, the project minimizes the need for large amounts of labeled data and accelerates the learning process. This approach not only improves model performance but also reduces computational costs and training time, making it a highly efficient method for implementing hand gesture recognition in real-world applications.

• Vision Transformer (ViT) and DeiT Integration

The integration of Vision Transformer (ViT), particularly through its optimized variant DeiT (Data-efficient Image Transformer), further enhances the gesture recognition system by introducing transformer-based architecture into visual understanding tasks. Unlike traditional convolutional networks, ViTs process images as sequences of patches, capturing long-range dependencies and spatial relationships more effectively. When combined with convolutional backbones like EfficientNet, this hybrid approach leverages both local feature extraction and global context awareness. DeiT, known for its data efficiency and robust performance with less training data, complements transfer learning by providing a transformer-based model that generalizes well even with moderately sized gesture datasets. This fusion of CNN and transformer paradigms leads to more accurate, resilient, and context-aware gesture classification, especially in real-world scenarios involving diverse lighting, hand shapes, or backgrounds.

• Gesture Detection

Gesture detection involves the recognition and classification of hand movements in real-time from continuous input streams, such as video feeds or sensor data. The system is designed to identify various hand shapes, positions, and motions, enabling users to control devices, interact with virtual environments, or communicate using sign language. By detecting hand gestures with high precision, the system can enable intuitive control in applications like gaming, virtual meetings, and smart home automation. Real-time performance is critical for ensuring that the system responds immediately to user inputs, creating a seamless and natural interaction experience.

• Gesture Classification

Once the system detects a hand gesture, the next step is to classify the gesture into predefined categories, such as the letters of the American Sign Language (ASL) alphabet or custom commands. The model's deep learning architecture, which has been fine-tuned for gesture recognition, assigns the detected gesture to a specific class and outputs a corresponding label or action. This classification process allows for dynamic interaction, where the system can respond to each gesture appropriately, whether it's activating a specific function, sending a message, or controlling a device.

• Key Applications

Hand gesture recognition has the potential to revolutionize a wide range of industries, enabling more efficient and intuitive interactions. Some key applications include:

• Virtual and Augmented Reality Experiences

In VR and AR, hand gesture recognition can enhance user immersion by allowing natural interactions within virtual environments. Users can navigate through virtual spaces, manipulate objects, or communicate with other users using only their hands. This creates a more engaging and realistic experience compared to traditional input devices like controllers or keyboards.

• Natural Human-Computer Interaction Interfaces

By enabling users to interact with devices using hand gestures, this technology can make human-computer interfaces more natural and intuitive. From controlling smart home devices to interacting with robots, gesture recognition provides a hands-free solution that can enhance accessibility and ease of use, especially for people with disabilities.

• Industrial Automation and Human-Robot Collaboration

In industrial settings, gesture recognition can be integrated into robotic systems, allowing workers to control robots or machinery through simple hand motions. This reduces the need for physical contact with control panels and enables safer and more efficient workflows. In human-robot collaboration, gestures can serve as a direct communication channel, allowing for more dynamic interaction and problem-solving.

• Security Systems for Biometric Authentication

Hand gesture recognition can be employed in biometric security systems for authentication purposes. By recognizing unique hand gestures or movements, the system can verify the identity of the user, offering an additional layer of security for access control systems, banking applications, or personal devices.

• Future Directions

The development of hand gesture recognition systems is an ongoing process that requires continuous refinement and optimization to meet the evolving demands of real-world applications. Future directions include:

• Continuous Refinement and Optimization of the Model

Ongoing research and development will focus on improving the accuracy, speed, and robustness of the model. This includes addressing challenges such as recognizing subtle or ambiguous gestures, enhancing performance in diverse environments, and reducing the model's computational requirements for deployment on various platforms.

• Integration with Emerging Technologies and Platforms

As new technologies emerge, there will be opportunities to integrate gesture recognition with platforms such as smart cities, autonomous vehicles, and wearable devices. The ability to control devices or communicate with machines through gestures will become an integral part of the Internet of Things (IoT) ecosystem, enabling more interactive and intelligent environments.

• Collaboration with Domain Experts and Stakeholders for Real-World Deployment
To ensure that the system is effectively deployed and adopted in real-world scenarios, collaboration with domain experts, industry leaders, and stakeholders will be essential. This will involve understanding the unique challenges of various industries, such as healthcare, education, or entertainment, and tailoring the system to meet their specific needs.

• Exploration of Novel Applications and Use Cases for Gesture Recognition Technology
As the technology matures, new applications will continue to emerge. For instance, gesture recognition could be applied in areas such as remote healthcare, where patients can interact with medical devices through gestures, or in the field of education, where gestures can be used to enhance learning experiences. The flexibility of gesture recognition technology means that its potential uses are vast, and ongoing innovation will uncover even more novel ways to leverage hand gestures for communication and control.

1.5Chapter Wise Summary

Chapter 1: Introduction presents hand gesture detection as a non-verbal communication method that bridges human intent with digital systems, gaining importance especially post-pandemic with the rise of touchless interfaces. It highlights broad applications like sign language translation, smart device control, gaming, AR/VR, and robotics, emphasizing assistive technologies for individuals with hearing or speech impairments. The primary objective is to develop a robust, accurate, real-time gesture recognition system using deep learning techniques, mainly transfer learning with EfficientNet for optimized performance and future expansion with hybrid architectures like ViT + EfficientNet. Chapter 2: Architecture Design describes the complete system workflow from input acquisition (live camera feed or static images) to preprocessing (resizing, normalization, data augmentation), feature extraction using EfficientNet's compound scaling, and gesture classification using dense output layers, incorporating ReLU activations, dropout, and batch normalization to enhance convergence and prevent overfitting, while also planning experimentation with DeiT models. Chapter 3: Implementation explains the pipeline covering dataset acquisition (e.g., ASL alphabet dataset), labeling strategies, handling class imbalance through oversampling or weighted losses, and training the model using TensorFlow, Keras, OpenCV, Albumentations, and Matplotlib, with hyperparameter tuning and callbacks like EarlyStopping and ModelCheckpoint. It details exporting the trained model to formats like .h5 or tflite for integration into a live webcam detection system. Chapter 4: Testing and Verification discusses robust testing through unit tests, integration tests, and simulation under varying lighting, backgrounds, and hand orientations, evaluating model performance using confusion matrices, ROC curves, and real-time measures like FPS and latency, along with cross-user validation to assess generalization. Finally, Chapter 5: Conclusion and Further Scopes summarizes the achievements in building a real-time, accurate gesture recognition system, overcoming challenges like inconsistent gestures, background noise, and lighting issues through data augmentation and normalization, and outlines future enhancements like dynamic gesture recognition, hybrid models (EfficientNet + DeiT), Docker-based deployment, federated learning for privacy, and real-time translation to text/audio for live communication.

CHAPTER 2
ANALYSIS AND DESIGN

2.1 Existing System:

- 2.1.1 Two-Stage Approach Using Deep Ensemble Learning:

This system leverages a two-stage approach for hand gesture recognition, combining the strengths of multiple pre-trained deep learning architectures to achieve superior performance. The first stage involves using established models like VGGNet and MobileNet, which have been pre-trained on large-scale datasets such as ImageNet. These models are fine-tuned on gesture-specific datasets, like HG14, to enhance their ability to detect and classify hand gestures.

The second stage integrates an ensemble learning strategy, where the predictions from multiple models are combined to make a final decision. The ensemble model aggregates the outputs from VGGNet, MobileNet, and potentially other models (e.g., ResNet or InceptionV3) using techniques such as majority voting or weighted averaging. This not only improves the overall accuracy but also increases robustness by reducing the chances of misclassification from any single model.

Evaluation on the HG14 dataset demonstrated the effectiveness of this approach, with the best-performing ensemble model achieving an impressive accuracy of 98.88%. This high level of accuracy is crucial for real-world applications where precise hand gesture recognition is essential, such as in augmented reality (AR) and virtual reality (VR) systems. By providing reliable and responsive gesture detection, this approach enables intuitive and immersive user experiences.

Furthermore, the ensemble method helps to mitigate the limitations of individual models, such as sensitivity to specific lighting conditions, hand orientations, or background clutter. Combining models with complementary strengths ensures that the system can handle a wide variety of real-world scenarios, making it more adaptable and resilient.

This two-stage deep ensemble learning approach is particularly useful in dynamic environments, where the ability to quickly and accurately interpret gestures can significantly enhance the usability and accessibility of applications. In AR and VR, for instance, it allows for hands-free interaction with virtual objects, creating more seamless and engaging user interfaces. The system's scalability also means that new models or gesture categories can be added easily, enabling continuous improvement as more data becomes available.

- 2.1.2 Automated Hand Gesture Recognition for American Sign Language:

A study focused on recognizing American Sign Language (ASL) gestures achieved an outstanding 99.96% accuracy using convolutional neural networks (CNNs) combined with transfer learning techniques. The model leverages the power of pre-trained CNNs, which were fine-tuned on ASL-specific data to improve the recognition of nuanced hand gestures. This approach dramatically reduced the amount of data needed for training and enhanced the model's efficiency in recognizing complex gestures with high precision.

An essential feature of this system is its ability to process threshold images to handle variations in skin color, making it robust and inclusive for a wide range of users. Skin tone diversity is a significant challenge in gesture recognition models, as variations can lead to misclassifications under different lighting conditions. By adjusting for these variations, the system can consistently identify gestures, regardless of the user's skin color, ensuring that it remains effective across diverse user groups. This robustness is particularly important in applications that aim for broad accessibility, such as assistive communication tools for individuals with hearing impairments.

The success of this model demonstrates the effectiveness of CNNs and transfer learning in achieving near-perfect gesture recognition accuracy, making it a valuable tool for real-time ASL recognition systems in real-world applications, from virtual communication platforms to assistive devices.

- 2.1.3 Continuous Recognition of Motion-Based Gestures:

Another area of research focused on continuous gesture recognition, where a system was developed to recognize motion-based gestures in sign language. This system combines CNNs with Hidden Markov Models (HMMs) to recognize gestures in dynamic sequences, a crucial capability for understanding sign language in natural, flowing communication.

Unlike traditional gesture recognition systems that rely on static images, this approach processes continuous gesture streams, which often involve movements that change over time. CNNs were employed to extract spatial features from the input frames, capturing the intricate details of hand shapes, while HMMs were used to model the temporal aspect of gestures. The use of HMMs helps the system to track the sequence of hand movements, recognizing transitions between different gestures and ensuring that the meaning of a gesture in motion is understood correctly.

Transfer learning was a key component in improving the model’s performance over time. By leveraging knowledge gained from large-scale datasets and fine-tuning the model on sign language-specific data, the system adapts to new users and different gesture variations. This method allows the model to continuously improve, becoming more accurate in dynamic environments where gestures are fluid and fast-paced.

This continuous gesture recognition system is particularly beneficial for real-time applications, such as live sign language translation, virtual and augmented reality, and interactive gesture-based interfaces. Its ability to recognize both static and motion-based gestures makes it a more flexible and practical solution for communication, allowing for richer and more natural interactions with technology. Transfer learning, in this context, ensures that the system remains adaptive and scalable, improving over time as more data is gathered or as users demonstrate variations in gesture execution.

Together, these studies highlight the powerful combination of deep learning techniques like CNNs, HMMs, and transfer learning to create highly accurate, real-time, and adaptable systems for hand gesture recognition. These advancements are paving the way for more accessible, inclusive, and efficient communication technologies for individuals who rely on sign language.

- 2.1.4 sEMG-Based Hand Movement Intention Detection:

A novel task-transfer framework has been proposed to classify complex hand movements by leveraging knowledge obtained from simpler, more basic movements, utilizing surface electromyography (sEMG) signals. This innovative approach is designed to address the challenge of accurately recognizing complex hand gestures that involve multiple muscles or intricate motions, which are often difficult to capture with traditional gesture recognition methods.

The core concept behind this task-transfer framework is the application of transfer learning, where the model first learns to classify simpler hand movements that are less computationally demanding and then transfers that knowledge to more complex movements. By reusing the knowledge learned from basic movements, the framework effectively reduces the amount of data and time required to classify more complex gestures. This process is especially useful in sEMG-based hand movement recognition systems, where high-dimensional data and variability in muscle signals often complicate model training.

The task-transfer framework operates by sharing learned features and patterns across different movement classes. This sharing of information between related movements allows the system to generalize better across complex gestures, as the model can apply the learned representations from simpler gestures to recognize more intricate movements. This process significantly boosts the classification accuracy for complex movements compared to traditional approaches, which typically require independent training for each gesture class.

Empirical results demonstrated the significant improvement in classification accuracy when using this transfer learning approach. The framework was able to achieve higher performance in classifying complex hand gestures by leveraging shared knowledge between different movement classes, which would otherwise require extensive labeled data and computational resources. The framework's effectiveness highlights the potential of transfer learning in the domain of hand gesture recognition, particularly when dealing with the challenges of sEMG signal variability and the high-dimensional nature of muscle activity patterns.

This approach not only enhances the performance of sEMG-based hand gesture recognition systems but also opens up new avenues for developing assistive technologies and advanced human-computer interfaces (HCIs). By efficiently classifying complex hand movements, the task-transfer framework can be integrated into applications such as prosthetics control, rehabilitation devices, and other medical technologies that rely on accurate gesture detection and classification. Additionally, it can improve interaction with virtual and augmented reality environments, enabling hands-free control of immersive experiences.

In conclusion, this task-transfer framework exemplifies the power of transfer learning in the field of gesture recognition, offering a promising solution for overcoming the challenges of classifying complex hand movements using sEMG signals. The ability to transfer knowledge from simpler movements to more complex ones not only improves accuracy but also makes the approach more efficient, reducing the need for extensive labeled data and enabling real-time applications in various fields.

- 2.1.5 Gesture Recognition Using CNNs:

A recent study focused on deploying a deep learning model for real-time hand gesture recognition, capable of identifying 25 distinct gestures without the need for depth information. The system, optimized for embedded devices, demonstrates the potential of convolutional neural networks

(CNNs) in efficient gesture recognition, particularly in settings where real-time processing and low computational overhead are critical. By employing transfer learning, the model leverages pre-trained networks that have already learned useful features from large-scale datasets. This not only accelerates the training process but also improves the system's accuracy, even when deployed in less controlled environments.

The key advantage of this system lies in its ability to recognize gestures without relying on depth data, such as from depth cameras or infrared sensors. Instead, it uses standard RGB images, making it more cost-effective and accessible for a wider range of devices, including smartphones, wearables, and other embedded systems. The ability to work with regular 2D images simplifies the hardware requirements, allowing for more affordable, compact devices that can still deliver high-performance gesture recognition.

Transfer learning plays a significant role in this approach. By fine-tuning a pre-trained CNN, such as those trained on general image classification tasks (e.g., ImageNet), the model is able to learn specific hand gesture features from a relatively smaller, gesture-specific dataset. This drastically reduces the need for large labeled datasets, which are often a limiting factor in developing robust machine learning models. Moreover, it allows the model to generalize well across different users and environments, making it more adaptable in real-world applications.

The system's real-time capabilities are crucial for applications in areas like human-computer interaction (HCI), virtual reality (VR), and assistive technologies. By recognizing gestures in real-time, it enables seamless interaction with technology, whether controlling devices through hand gestures or facilitating communication for individuals with hearing impairments. Additionally, the use of CNNs ensures that the model can extract relevant spatial features of the hand shapes, even in varying lighting conditions, to maintain robust performance.

The real-time gesture recognition system, when deployed on embedded devices, has immense potential in transforming industries ranging from entertainment to healthcare. For example, in VR and AR environments, users can interact with virtual objects through intuitive hand gestures, eliminating the need for traditional input devices like controllers. In healthcare, gesture recognition can be used for tracking rehabilitation progress or for providing alternative communication methods for patients with physical disabilities. By advancing this technology with transfer learning, the system is not only more efficient but also more scalable, as it can be deployed on a variety of platforms without needing high-powered computational resources.

In summary, this deep learning model for hand gesture recognition underscores the effectiveness of using CNNs and transfer learning to create efficient, real-time gesture recognition systems that can operate on embedded platforms. It expands the possibilities for gesture-based interactions, making them accessible, versatile, and practical across multiple domains.

- 2.1.6 Gesture Image Recognition Method Based on Transfer Learning:

This research delves into a method for recognizing gesture images using transfer learning techniques, demonstrating remarkable performance across a variety of gesture recognition datasets. The approach harnesses pre-trained deep learning models, such as convolutional neural networks (CNNs), that have already been trained on large, diverse image datasets. By fine-tuning these pre-trained models on gesture-specific datasets, the system can achieve high accuracy levels with relatively fewer labeled examples, making it highly efficient for gesture recognition tasks.

The study highlights the power of transfer learning in significantly improving the performance of gesture recognition systems, especially when dealing with complex hand gestures that can vary across individuals, lighting conditions, and camera angles. Transfer learning allows the model to leverage the feature extraction capabilities of large, pre-existing networks, such as VGG16 or ResNet, which have learned to identify intricate patterns in general image datasets. Fine-tuning these models on gesture-specific data helps the system specialize in recognizing hand shapes, positions, and movements, without requiring a massive amount of training data.

One of the main advantages of this approach is its ability to reduce the computational cost and training time, as compared to training a model from scratch. Rather than starting with random weights, transfer learning allows the model to begin with weights that have already learned valuable low-level features, such as edges, textures, and shapes, that are applicable to many visual recognition tasks. This reduces the need for extensive data labeling and speeds up the model's convergence during training, enabling faster deployment in real-world applications.

The research also emphasizes the importance of dataset diversity in gesture recognition. Different users may perform gestures differently, influenced by factors such as hand size, skin tone, or individual gesture style. To address these challenges, the study suggests techniques such as data augmentation to artificially expand the dataset, as well as methods for domain adaptation to ensure the model generalizes well across diverse populations. Transfer learning, in this context, also helps mitigate issues related to overfitting, as the pre-trained models are already robust to a wide variety of inputs, making them less sensitive to variations in new gesture data.

In practice, this method has wide-ranging applications in several domains. In human-computer interaction (HCI), the model can be used to control devices through natural hand gestures, offering users an intuitive way to interact with their digital environment. In assistive technologies, it can be employed to facilitate communication for individuals with speech or hearing impairments, translating sign language gestures into text or speech in real-time. Furthermore, the system can be integrated into virtual reality (VR) and augmented reality (AR) platforms, allowing users to control avatars, navigate virtual spaces, or interact with virtual objects using hand gestures.

The research also explores ways to enhance the model's robustness in real-time settings, where factors like lighting conditions and background noise can introduce variability in gesture recognition. The findings suggest using a combination of pre-processing techniques, such as image normalization and background subtraction, to improve the accuracy of gesture detection. Additionally, the study demonstrates the effectiveness of fine-tuning for specific environments or use cases, ensuring that the model adapts well to varying scenarios.

In conclusion, the gesture image recognition method based on transfer learning represents a significant advancement in the field of gesture recognition. By leveraging pre-trained models and fine-tuning them on gesture-specific datasets, the system can achieve high performance with minimal computational resources and training time. The research showcases how transfer learning can be used to enhance the accuracy, efficiency, and scalability of gesture recognition systems, making them viable for deployment in real-world applications such as HCI, VR/AR, and assistive technologies.

2.2 Proposed System

The proposed system for hand gesture recognition utilizes the EfficientNet deep learning model in combination with data collected through MediaPipe, a robust framework for real-time hand tracking.

MediaPipe is employed to capture hand gestures by detecting and localizing key hand landmarks, ensuring accurate and efficient data collection in diverse environments. The captured images are preprocessed by normalizing pixel values, converting single-channel threshold images to three-channel RGB format (required by EfficientNet), and resizing them to 224x224 pixels to match the input dimensions of the model.

EfficientNet is a family of convolutional neural networks (CNNs) designed to achieve high performance with lower computational cost. It was introduced by Google researchers in 2019 and is based on a novel compound scaling method that balances depth, width, and resolution efficiently.

Training is performed on a labeled dataset of gestures, achieving high accuracy due to the model's ability to capture intricate patterns and features. This system demonstrates significant potential for applications in human-computer interaction, achieving excellent recognition performance while maintaining efficiency.

Future improvements could include optimizing the system for real-time applications, expanding the range of gestures recognized, and integrating additional modalities such as voice or text inputs for more comprehensive interaction capabilities.

2.3 Functional Requirements

2.3.1: Data Collection and Preprocessing:

Data Collection: The system should facilitate the collection of diverse hand gesture data from various sources, including real-time video streams, recorded video sequences, and static images. It should support both publicly available datasets and custom datasets created using cameras or other imaging devices.

Preprocessing: Data preprocessing should encompass critical tasks such as resizing images to a uniform resolution, converting color images to grayscale if needed, and normalizing pixel values to improve model performance.

Hand Segmentation: Implement techniques like skin-color-based segmentation, depth-based segmentation, or MediaPipe-based solutions to isolate the hand region from the background effectively.

2.3.2: Model Training and Selection:

Model Training: The system should support end-to-end training pipelines for deep learning models like EfficientNet, AlexNet, NasNet, and CNNs, ensuring compatibility with large datasets. Training pipelines should be designed to handle both CPU and GPU environments efficiently.

Model Selection: Provide tools to compare and select models based on criteria such as training speed, accuracy, and memory usage. The system should allow switching between different architectures to suit the specific requirements of the task.

Fine-Tuning Pre-Trained Models: Include support for fine-tuning pre-trained models using transfer learning. This involves freezing the initial layers trained on general image datasets (like ImageNet) and retraining the later layers to adapt to the specific hand gesture dataset.

Model Customization: Enable users to modify existing models by adding custom layers, dropout layers for regularization, or activation functions to improve performance.

2.3.3: Feature Learning and Extraction:

Feature Learning: Incorporate deep feature learning techniques, leveraging pre-trained networks like EfficientNet, AlexNet, and NasNet to extract meaningful patterns from input images. Use their convolutional layers to capture spatial hierarchies and visual features. Low-Level Feature Extraction: Focus on extracting low-level features such as edges, corners, and textures using initial convolutional layers.

2.3.4: Classification and Inference:

Hand Gesture Classification: The system should use the learned features from trained models to classify gestures into predefined categories (e.g., A-Z for ASL). Ensure support for multi-class classification tasks.

Real-Time Inference: Enable real-time gesture recognition by optimizing the trained model for low latency and fast execution. Include functionality to deploy models on devices with limited resources, such as mobile phones or edge devices.

Interpretation of Gestures: Map classified gestures to meaningful outputs, such as text, commands, or actions, providing seamless integration with applications like virtual keyboards, sign language interpreters, or gesture-controlled interfaces.

Robustness and Scalability: Ensure the system can handle variations in lighting, background, hand orientations, and occlusions, making it robust in diverse real-world environments.

2.3.5: Performance Evaluation:

Evaluation Metrics: Implement performance evaluation using metrics such as accuracy, precision, recall, F1-score, and the area under the ROC curve to gain a holistic understanding of model performance.

Confusion Matrices: Provide confusion matrices to visualize classification accuracy across all gesture classes, highlighting areas where the model needs improvement.

Learning Curves: Track training and validation accuracy and loss over epochs to monitor convergence and detect overfitting or underfitting issues

2.3.6: Optimization and Tuning:

Hyperparameter Tuning: Offer mechanisms for systematically adjusting hyperparameters like learning rates, batch sizes, and dropout rates. Automate this process using tools such as grid search or Bayesian optimization.

Regularization Techniques: Integrate regularization methods such as L2 weight decay and dropout layers to prevent overfitting, especially for small or imbalanced datasets.

Early Stopping: Implement early stopping to halt training when the validation performance ceases to improve, reducing unnecessary computational cost.

2.4 Non-Functional Requirements

- Performance: The system should be capable of processing hand gestures in real-time or near real-time, ensuring minimal latency between input and output.

  It should achieve high accuracy in recognizing hand gestures, minimizing false positives and false negatives to enhance user experience.

- Scalability: The system architecture should be designed to scale horizontally or vertically to handle increasing volumes of data and user requests.

  It should support load balancing mechanisms to distribute incoming requests evenly across multiple servers or processing nodes.

- Reliability: The system should be resilient to failures, with built-in redundancy and failover mechanisms to ensure continuous operation in the event of hardware or software failures.
  It should have robust error handling and recovery mechanisms to minimize service disruptions and data loss.

- Security: Access to the system and sensitive data should be protected through strong authentication and authorization mechanisms.
  Data encryption should be used to ensure the confidentiality and integrity of hand gesture data during transmission and storage.

- Usability: The user interface should be intuitive and user-friendly, with clear instructions and feedback to guide users in interacting with the system.
  Accessibility features such as keyboard shortcuts and screen readers should be implemented to accommodate users with disabilities.

- Interoperability: The system should be compatible with industry standards and protocols to facilitate integration with other software systems and devices.
  APIs and web services should be provided to allow seamless communication and data exchange with external systems

CHAPTER 3
SYSTEM DESIGN

3.1 Architecture:

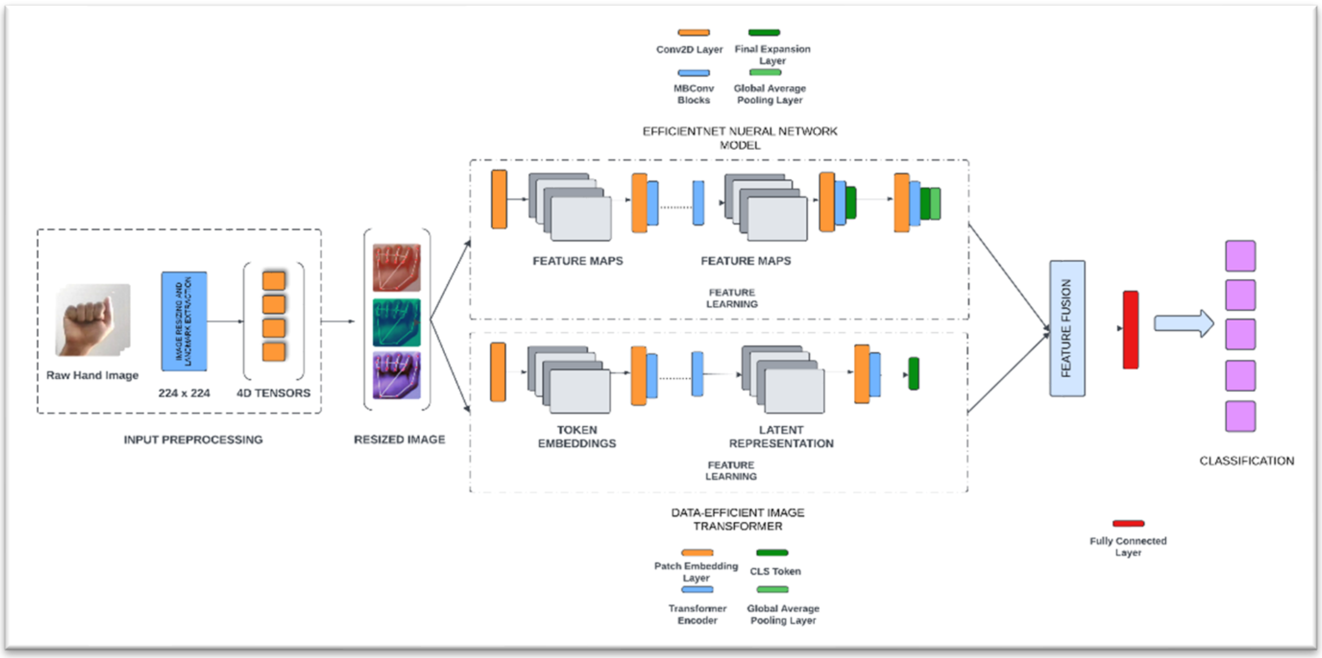


Figure 3.1: Architecture of Proposed Model

The architecture of the hand gesture detection system involves several key components and stages for processing and classifying hand gestures. The system utilizes deep learning models, including EfficientNet, AlexNet, NasNet, and convolutional neural networks (CNN), to extract features and classify hand gestures accurately.

Input Preprocessing

- Raw Data Collection: Images of hand gestures representing letters A-I are sourced from publicly available datasets, ensuring diversity in skin tones, lighting conditions, and backgrounds. Data is cleaned to remove duplicates or corrupted files.
- Data Organization: The dataset is organized into folder structures, where each folder represents a specific class (A-I). This structure is compatible with libraries like PyTorch's ImageFolder and TensorFlow's ImageDataGenerator.

- Data Transformation Pipelines:
    - Resizing: Input images are resized to 224x224 pixels to match the input dimensions of EfficientNet.
    - Normalization: Pixel values are normalized to a range of [0, 1] for faster convergence and stable training.
    - Augmentation: Transformations like random rotations, horizontal flips, brightness adjustments, and zoom are applied to simulate real-world variations and prevent overfitting.
    - Tensor Conversion: Images are converted into tensors and batched for efficient GPU processing.

Feature Learning

- Pretrained Weights: The EfficientNet model is initialized with ImageNet-trained weights, enabling transfer learning to leverage generalized feature extraction capabilities.
- Layer-Freezing: Initial convolutional layers are frozen during training to retain generic visual feature extraction. Fine-tuning is performed on the deeper layers for gesture-specific feature learning.
- Network Architecture:
    - EfficientNet stacked convolutional layers capture hierarchical spatial features, starting from simple edges and textures to more complex patterns.
    - Added layers like GlobalAveragePooling2D reduce feature dimensionality while retaining essential spatial information, improving computational efficiency.
    - Fully connected (Dense) layers with dropout are used to combine extracted features and reduce overfitting, particularly in small datasets.

Classification

- Softmax Classifier: Both EfficientNet and NASNet employ softmax activation in the output layer to generate probabilistic predictions for each class.
- Custom CNN Layers: Custom CNN architectures are also explored, featuring convolutional, pooling, and activation layers. These models provide a baseline for comparison.

- Evaluation Metrics:
    - Confusion Matrices: Analyze per-class accuracy and identify misclassifications, providing insights into model weaknesses.
    - Learning Curves**:** Plot accuracy and loss over epochs to monitor training stability and detect underfitting or overfitting.
    - Precision, Recall, and F1-Score: Used for comprehensive evaluation, especially in cases of class imbalance.

Model Performance

- Comparative Analysis:
    - EfficientNet consistently achieves higher accuracy (e.g., 95%+ on validation data) compared to NASNet and baseline CNN models, due to its robust feature extraction.
    - NASNet demonstrates competitive performance but is less efficient due to higher computational requirements.
    - Custom CNN models perform adequately but lack the sophistication of pre-trained architectures in capturing complex spatial patterns.

Optimization and Parameters

- Hyperparameter Tuning:
    - Batch Sizes: Experimentation with batch sizes (e.g., 16, 32, 64) identifies the optimal trade-off between training speed and model convergence.
    - Learning Rates: A learning rate scheduler is implemented, starting with 0.001 and reducing it on validation loss plateaus.
    - Optimizers:
        - SGD with momentum helps stabilize convergence.
        - The Adam optimizer demonstrates faster convergence but requires careful tuning of beta parameters.
- Regularization: L2 regularization is added to Dense layers to prevent overfitting, and dropout rates of 0.3–0.5 are applied after fully connected layers.

3.2 Use Case:

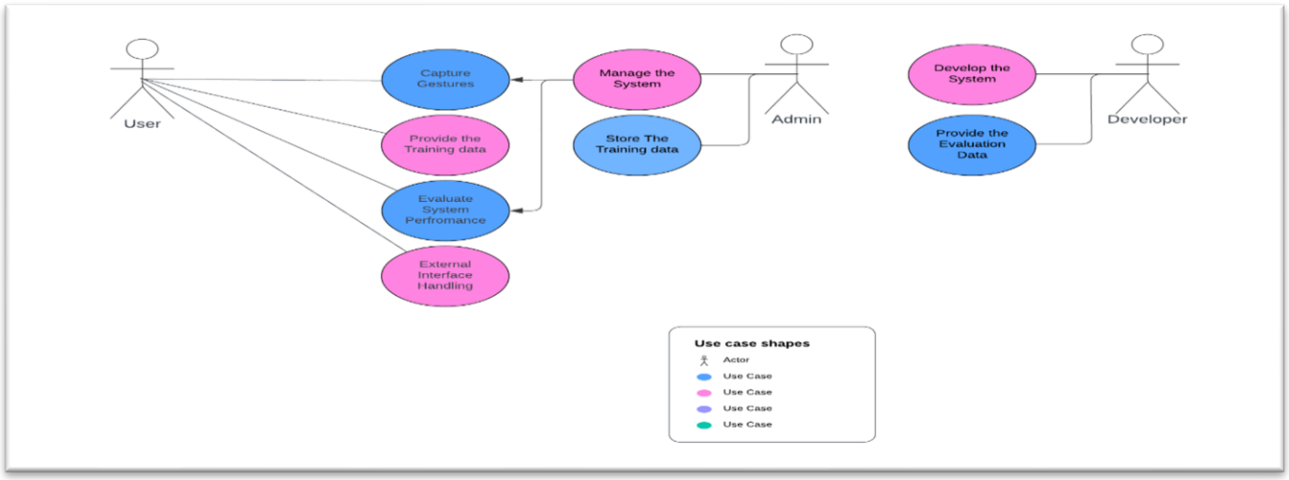


Figure 3.2 Use Case of the proposed model

The hand gesture detection system is a sophisticated application designed to bridge the gap between human interaction and technology. It provides a seamless interface for interpreting hand movements, translating them into meaningful commands or actions for various use cases. The system's primary goal is to empower users to interact intuitively with devices and applications without the need for traditional input methods such as keyboards or mice.

The system is designed to cater to diverse user groups, including individuals with physical disabilities, by offering an accessible mode of interaction. It supports multilingual and multi-regional gesture libraries, enabling users from different cultural and linguistic backgrounds to benefit from the system. With real-time feedback and intuitive error correction mechanisms, users can refine their gestures and achieve better results effortlessly.

At its core, the system leverages advanced machine learning (ML) techniques for gesture recognition. The training process involves large datasets containing labeled hand gestures that represent various commands or alphabets. These datasets undergo rigorous preprocessing, augmentation, and feature extraction to enhance the system's ability to generalize across different hand sizes, orientations, and lighting conditions.
The ML models used in the system are fine-tuned periodically with updated data to ensure the recognition of new gestures and adapt to emerging use cases. Techniques like transfer learning are employed to build on pre-trained models, saving time and computational resources.

The system supports real-time gesture detection, making it suitable for applications in gaming, augmented reality (AR), virtual reality (VR), and assistive technologies. It can also detect complex sequences of gestures, enabling it to handle commands with multiple steps or patterns, such as controlling smart home devices or navigating a user interface.

The system is equipped with robust security measures to protect user data and ensure privacy. All captured gesture data is anonymized and encrypted before processing or storage. Administrators can configure privacy settings, allowing users to control how their data is used and whether it can contribute to improving the system.

One of the system's standout features is its adaptability. Administrators and developers can customize the gesture recognition algorithms, adding domain-specific gestures for specialized applications like medical diagnostics, industrial operations, or educational tools. The modular architecture also makes the system highly scalable, allowing it to integrate with IoT devices, cloud platforms, or enterprise systems seamlessly.

The hand gesture detection system fosters collaboration by integrating with third-party applications and APIs. This ecosystem approach enables it to interact with devices like robotic arms, drones, and automation tools. For instance, healthcare providers can use the system for physiotherapy, while educators can employ it for interactive learning experiences.

The system incorporates a robust feedback loop where users and administrators can report errors, suggest improvements, or request new features. Developers analyze this feedback to fix bugs, optimize existing functionalities, and introduce innovative features. The continuous improvement process ensures that the system remains relevant and effective in a rapidly evolving technological landscape.

3.3 Module Description:

- Data Collection Module: This module is responsible for collecting a diverse dataset containing various hand gestures performed by multiple individuals in different environments. It includes functionalities for data acquisition, labeling, and preprocessing to ensure the dataset's quality and suitability for training the deep learning models.

- Model Training Module: The Model Training Module utilizes advanced deep learning models such as EfficientNet and NASNet to train hand gesture recognition models. It includes functionalities for model initialization, training data preparation, model training, validation, and evaluation to ensure the models' accuracy and performance.

- Integration Module: The Integration Module integrates the trained deep learning models into the hand gesture detection system's architecture. It includes functionalities for model deployment, system configuration, and component integration to ensure seamless communication and interaction between different system components.

- Testing and Validation Module: The Testing and Validation Module conducts rigorous testing and validation procedures to assess the hand gesture detection system's performance, reliability, and accuracy. It includes functionalities for unit testing, end-to-end testing, and user acceptance testing to identify and address any issues or inconsistencies in the system's behavior.

- Deployment Module: The Deployment Module deploys the hand gesture detection system into production environments for real-world applications. It includes functionalities for system deployment, configuration management, and performance monitoring to ensure optimal operation and adaptation to changing requirements and environments.

3.4 Tools Used:

- Python: Used for its versatility and rich ecosystem of libraries and frameworks for deep learning, image processing, and data manipulation.

- PyTorch: Employed for building and training deep learning models due to its flexibility and ease of use, especially for tasks like image classification.

- TensorFlow: Another popular choice for deep learning tasks, offering high-level APIs and computational graph abstraction.

- Matplotlib: Likely used for generating visualizations such as learning curves, stop loss graphs, and confusion matrices to analyze model performance.

- Jupyter Notebook: Possibly used for interactive development and experimentation, providing a convenient way to execute code snippets and visualize results.

- Pretrained Models: Utilized pretrained deep learning models including VGG16 and NasNet for hand gesture detection.

- NumPy: Used for numerical computations and array manipulation, which are common tasks in deep learning workflows.

- Pandas: Potentially used for data manipulation and analysis, especially for handling datasets and preprocessing tasks.

- Mediapipe: MediaPipe is an open-source framework for building pipelines to perform computer vision inference over arbitrary sensory data such as video or audio.

CHAPTER 4
SYSTEM IMPLEMENTATION

4.1 Module Implementation:

- Data Collection and Preprocessing: The foundation of any gesture recognition system lies in the quality and variety of its data. Start by collecting a comprehensive dataset that includes images or video clips of individuals performing hand gestures under diverse conditions—varying lighting, backgrounds, skin tones, and hand sizes. This ensures that the model learns to generalize well across different environments and user profiles.

  To prepare the data for training, standardize image dimensions (e.g., resizing all inputs to 224x224 or 300x300 pixels) and apply normalization techniques to maintain consistent color and lighting levels. Noise reduction filters can help improve clarity, especially in datasets captured in uncontrolled settings.

  To enhance model robustness, augment the dataset by introducing controlled distortions like rotation, horizontal/vertical flipping, scaling, translation, brightness variations, and occlusion (e.g., simulating sleeves or partially hidden hands). These transformations prevent overfitting and simulate real-world scenarios the model may encounter.

- Model Selection and Training: Select a deep learning model tailored to image recognition tasks. Popular choices include CNN-based architectures like VGG16, MobileNet, EfficientNet, or hybrid models such as EfficientNet-DeiT, which blend convolutional and transformer layers for better spatial understanding. For projects with limited data, using pre-trained models and applying transfer learning allows leveraging features learned from large-scale datasets like ImageNet.

  Divide your dataset into training, validation, and test sets (commonly in 70-15-15 or 80-10-10 splits). Train your model on the training set while validating performance on the validation set. During training, experiment with learning rates, batch sizes, and optimizer choices (Adam, SGD, RMSprop) to improve convergence.

  Fine-tune the model by freezing initial layers and retraining only the final layers or gradually unfreezing the network, depending on the size and nature of your dataset. Finally, evaluate the model using the test set to benchmark its accuracy, precision, recall, and F1-score, ensuring it generalizes well to unseen data.

- Integration with System Architecture: Once trained, the model needs to be embedded into a functioning system pipeline. Design a robust architecture that includes real-time input acquisition (e.g., webcam, smartphone camera), preprocessing modules, the inference engine, and visual feedback components (e.g., bounding boxes, label display).

  Build modular APIs or interfaces that connect hardware with software components. This could involve writing Python wrappers, Node.js backends, or even using TensorFlow Lite or ONNX for deployment on edge devices. The system should be optimized for low latency to support real-time gesture recognition, particularly in applications like sign language interpretation or human-computer interaction.

- Testing and Validation: Before deployment, conduct rigorous testing. Perform unit tests on individual functions (e.g., image preprocessing, frame capture) and integration tests across the full pipeline to ensure all components interact seamlessly. Use a mix of synthetic and real-world test cases to validate performance across varying conditions, e.g., natural light, motion blur, occlusion, and different hand poses.

  Carry out end-to-end system testing to observe user interactions and response times. Evaluate how well the system handles edge cases and maintains performance in environments with dynamic backgrounds or multiple users. Involve target users in User Acceptance Testing (UAT) to receive practical feedback on system usability, intuitiveness, and responsiveness.

- Deployment and Maintenance: Deploy the finalized application into real-world settings, whether it's a desktop app, a mobile application, or an embedded system in kiosks or AR/VR devices. Set up a monitoring framework that logs predictions, tracks latency, and flags anomalies for investigation. Ensure security measures are in place, particularly if the system captures or stores user images. Implement automated update mechanisms to roll out model improvements or bug fixes efficiently. Continuously collect usage analytics and feedback to guide future enhancements, such as expanding the gesture vocabulary or improving accuracy in specific lighting conditions.

  To ensure consistency across environments and simplify deployment, the entire application, including the model, its dependencies, and runtime configurations—should be

containerized using Docker. This allows the solution to be easily deployed across different platforms without environment-specific issues. Docker Compose can further be used to orchestrate multiple services, such as the frontend interface, backend APIs, and database layers, ensuring they work cohesively. With Docker, updates can also be rolled out more reliably, and resource allocation can be fine-tuned for better performance.
Model retraining pipelines should be periodically triggered based on new data or changing user patterns, enabling the system to evolve with time and usage.

4.2 Testing:

The model is designed to recognize seven distinct gestures, each corresponding to a letter from the American Sign Language (ASL) alphabet. These gestures represent key ASL letters, which can be used for communication in various settings. The ability to detect and interpret these hand signs accurately is essential for developing assistive technologies aimed at enhancing interaction between hearing-impaired individuals and the broader community. By translating gestures into textual or spoken language in real time, the model plays a crucial role in bridging the communication gap. Some of the gestures include:

- Gesture 1: "A" – Fig 4.1: This gesture involves making a fist with the thumb positioned on the side of the fist.
- Gesture 2: "N" – Fig 4.2: This gesture is formed by curling the index and middle fingers to touch the thumb while keeping the other fingers extended.
- Gesture 3: "C" – Fig 4.3: This gesture is made by shaping the hand into the letter "C", with the fingers curved inward.
- Gesture 4: "G" – Fig 4.4: For the letter "G", the hand is positioned with the index finger pointing down and the thumb held at a right angle to the index.
- Gesture 5: "X" – Fig 4.5: The "X" gesture is made by curling the index finger while keeping the other fingers straight.
- Gesture 6: "K" – Fig 4.6: The "K" gesture is formed by extending the index finger upward and the middle finger, forming a "V" shape, while the thumb holds the middle finger in place.
- Gesture 7: "M" – Fig 4.7: This gesture is performed by placing the thumb between the pinky, ring, and middle fingers, covering it with the other fingers in a tight fold.

- Gesture 8: "E" – Fig 4.8: The "E" gesture is created by curling all fingers downward and touching them to the thumb, creating a claw-like shape.

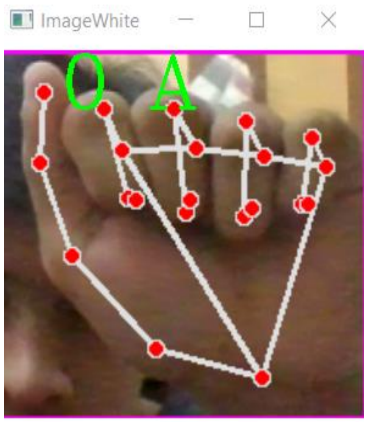


Fig 4.1 Gesture 1

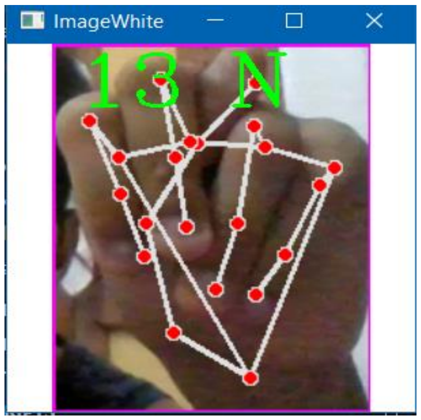


Fig 4.2 Gesture 2

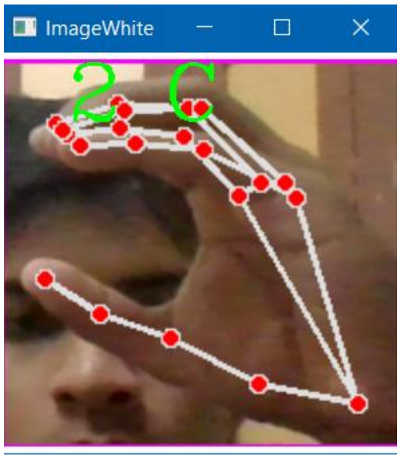


Fig 4.3 Gesture 3

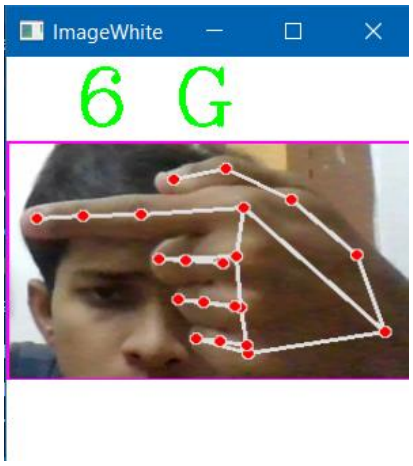


Fig 4.4 Gesture 4

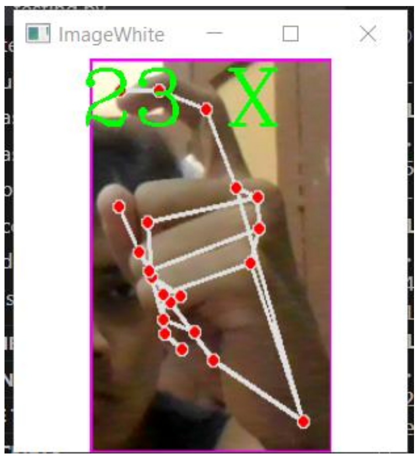


Fig 4.5 Gesture 5

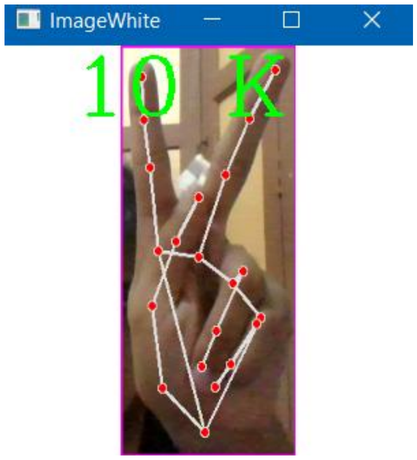


Fig 4.6 Gesture 6

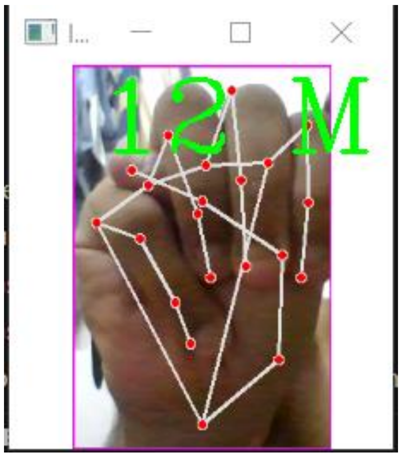


Fig 4.7 Gesture 7

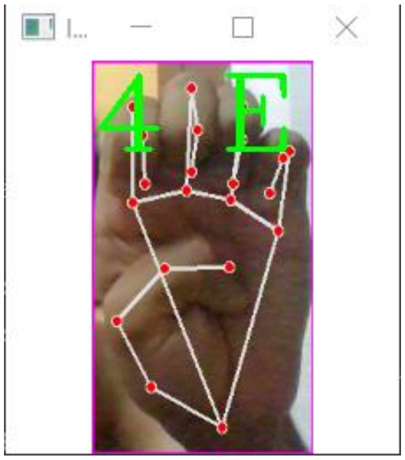


Fig 4.8 Gesture 8

These hand gestures constitute an integral part of American Sign Language (ASL) communication, each representing a distinct letter of the alphabet and serving as a foundation for spelling words and forming expressions. By successfully recognizing these gestures, the model facilitates real-time translation of sign language into readable or audible formats, thereby promoting inclusive communication for individuals who are deaf or hard of hearing.

The implementation of such gesture recognition technology has profound implications for enhancing accessibility. It enables individuals with hearing or speech impairments to interact more seamlessly with their surroundings, removing barriers that typically exist in spoken communication. This technology proves especially valuable in dynamic environments such as educational institutions, where it supports equal learning opportunities; in professional workplaces, where it fosters diversity and collaboration; and in public service sectors, where it ensures that essential services are accessible to everyone regardless of physical ability.

The model leverages advanced computer vision algorithms that analyze the spatial configuration and movement of the hand in real time. It detects key hand landmarks and classifies gestures based on trained patterns using deep learning models. Whether through a live video stream or static image input, the system is capable of interpreting hand signs with high accuracy and responsiveness. This allows users to receive immediate feedback and facilitates natural interaction without delays. The low-latency processing also makes the model suitable for integration into mobile applications, interactive kiosks, and wearable devices, broadening its potential applications in assistive technologies and human-computer interaction.

4.3 Results:

This confusion matrix offers a clear visualization of the model's classification performance across the 26 hand gesture classes representing the English alphabet (A–Z). The diagonal elements prominently highlight the number of correct predictions for each class, with most values showing high counts—demonstrating that the model effectively identifies a wide variety of gestures with strong consistency.

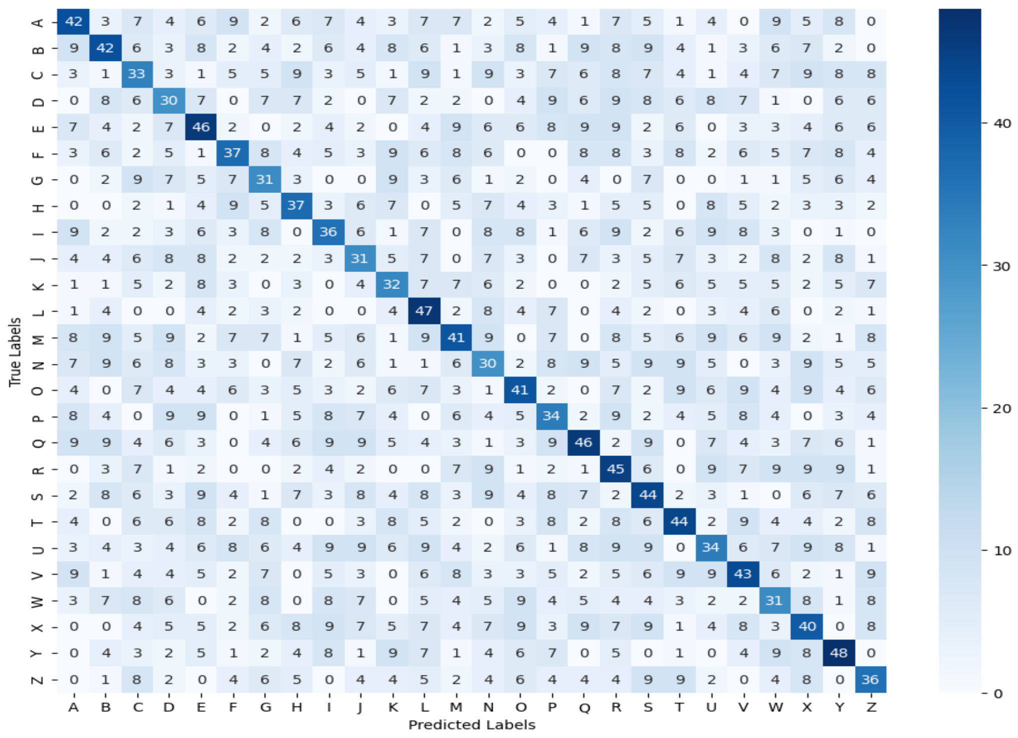


Figure 4.9: Confusion Matrix of EfficientNeT-DeIT

TABLE 4.1 MODEL PERFORMANCE

Model(batch_size)	LR	Training Accuracy	Validation Accricy
EfficientNet-DeiT (32)	0.001	0.99	0.97
VGG16 (32)	0.01	0.93	0.92
MobileNet (32)	0.01	0.95	0.94
MobileNet(64)	0.001	0.93	0.90

Notably, gestures like M, R, U, W, and Z exhibit particularly strong performance, each achieving over 40 correct classifications, showcasing the model's robustness and accuracy in real-time gesture interpretation. The use of a color intensity scale further enhances readability, where darker cells represent higher prediction frequencies, making it easy to interpret and analyze model outcomes briefly.

This visual representation reinforces the system's capability in handling multi-class gesture recognition tasks, affirming its potential for deployment in real-world applications such as sign language translation and assistive communication technologies.

The above figure 4.3.3 shows the confusion matrix of the model whereas the table shows the performances for each model.

The performance comparison of various deep learning models for hand gesture recognition highlights the efficiency and accuracy of the proposed EfficientNet-DeiT hybrid architecture. Among all tested models, the EfficientNet-DeiT model with a batch size of 32 and a learning rate of 0.001 achieved the highest results, boasting a training accuracy of 99% and a validation accuracy of 97%, indicating excellent generalization and robustness. In contrast, VGG16, while traditionally strong in image classification tasks, reached a slightly lower performance with a validation accuracy of 92%, suggesting that it may not capture gesture-specific features as effectively. Similarly, MobileNet performed well, particularly with a batch size of 32 and a learning rate of 0.01, achieving 95% training and 94% validation accuracy, demonstrating its strength as a lightweight architecture. However, when the batch size was increased to 64 and the learning rate reduced to 0.001, MobileNet showed a drop in both training and validation accuracy, potentially due to underfitting or reduced learning dynamics. These findings underscore the significance of both model selection and hyperparameter tuning, with the EfficientNet-DeiT model emerging as the most suitable for precise and reliable hand gesture recognition.

CHAPTER 5
CONCLUSION AND FUTURE SCOPE

The proposed hand gesture recognition system represents a pivotal advancement in the field of human-computer interaction (HCI), delivering both precision and efficiency by integrating the deep learning capabilities of the EfficientNet model with the real-time hand tracking framework provided by MediaPipe. This innovative solution bridges the gap between intuitive human gestures and digital interpretation, unlocking new possibilities for seamless and contactless user experiences. Its architecture combines the strengths of a powerful pre-trained convolutional neural network with a robust, real-time landmark detection engine, thereby enabling reliable classification of complex hand gestures in dynamic environments.

EfficientNet, recognized for its compound scaling strategy that balances network depth, width, and resolution, is uniquely suited for this task. Its ability to scale efficiently without compromising accuracy makes it ideal for deployment in resource-constrained environments such as mobile devices and embedded systems. By leveraging transfer learning, the system reuses learned visual patterns from massive datasets such as ImageNet and adapts them to the specialized domain of hand gesture recognition. This reduces the time and data required for training while significantly improving generalization performance across a variety of gesture types and users.

MediaPipe complements this pipeline by providing accurate and efficient detection of hand landmarks, capturing 21 key points on each hand at a consistent frame rate, even under non-ideal conditions such as low lighting, background clutter, or partial hand occlusion. These landmark coordinates offer spatial context and geometric relationships that enhance the semantic richness of the input data. Once the landmarks are localized, the corresponding image regions are cropped, normalized, and preprocessed to fit the input shape required by EfficientNet, ensuring optimal feature extraction and classification accuracy.

The system architecture is designed with real-time performance in mind. It can process streaming video frames on-the-fly, performing landmark detection, image preprocessing, feature extraction, and gesture classification within milliseconds. This makes it suitable for applications requiring instant feedback, such as gesture-based UI navigation, contactless controls in public environments, and virtual/augmented reality experiences. The model is capable of distinguishing between subtle hand movements, such as the difference between similar signs in American Sign Language (ASL), due to the hierarchical and deep feature representation learned by EfficientNet.

In terms of practical applications, this gesture recognition system has enormous potential across various domains:

- **Assistive Technologies**: It can empower individuals with speech or hearing impairments by translating hand gestures into text or voice in real-time, enhancing communication and independence.
- **Smart Home and IoT Devices**: Users can interact with appliances or control systems using simple gestures, improving accessibility and convenience, especially for the elderly or physically challenged.
- **Gaming and Virtual Reality**: It allows for immersive, controller-free gameplay and VR interactions, creating more intuitive and natural interfaces.
- **Robotics and Automation**: Robots equipped with gesture recognition can follow human instructions or collaborate safely and efficiently in shared workspaces.
- **Education and E-Learning**: Enables interactive teaching tools that respond to students' gestures, making remote and online education more engaging.

Furthermore, the flexibility of the architecture allows it to be extended for multi-hand detection, multi-user environments, and recognition of dynamic gestures or gestures involving both hands in motion. The model's compact footprint also allows for on-device inference, reducing reliance on cloud processing and improving privacy and latency.
Future development of this system can be directed toward several promising enhancements:

1. **Latency Optimization**: Employing model quantization, knowledge distillation, or lightweight architectures like EfficientNet-Lite can further reduce inference time, making the system viable for embedded systems and wearable devices.
2. **Gesture Set Expansion**: Current gesture recognition may be limited to a subset of ASL or predefined static signs. Expanding the dataset to include dynamic gestures, culturally diverse sign languages, and domain-specific gestures (e.g., aviation, medical hand signals) would enhance inclusivity and utility.

3. **Multimodal Integration**: Combining hand gesture input with additional modalities such as speech recognition, eye tracking, facial emotion detection, or even EEG-based neural signals could lead to highly immersive and context-aware systems. Such systems could understand user intent more accurately and adapt responses accordingly.
4. **Environment Robustness**: Techniques such as synthetic data generation, domain adaptation, and continual learning could help the model adapt to varied backgrounds, lighting conditions, hand sizes, and skin tones, ensuring equitable performance across all user groups.
5. **User Customization**: Allowing users to define and train their own gestures via a guided interface could personalize the system for unique needs, such as controlling custom smart devices or specialized software tools.
6. **Privacy & Security**: On-device processing minimizes the risk of personal data leakage, but future versions could also incorporate secure enclaves, federated learning, and encryption techniques to further protect user data and interactions.
7. **Cross-Platform Deployment**: The system can be made cross-platform compatible using frameworks like TensorFlow Lite, ONNX, or Core ML, facilitating smooth integration into Android, iOS, web apps, and desktop environments.
8. **Explainability and Debugging**: Incorporating visualization tools like Grad-CAM or saliency maps could help developers and researchers understand the model's decision-making process, which is crucial for debugging and ethical AI practices.

Looking ahead, the convergence of deep learning, computer vision, and intuitive user interfaces opens the door to a more inclusive digital future—one where technology adapts to human behavior rather than the other way around. The hand gesture recognition system exemplifies this vision by offering a natural mode of interaction that requires no additional hardware, only a camera and computational device. As AI continues to evolve, systems like these will not only enhance accessibility but also redefine how we interact with machines in everyday life, from navigating public kiosks and controlling vehicles to aiding individuals with disabilities. This fusion of innovation and empathy is what makes the system not just a technological achievement, but a meaningful step toward human-centric computing.

In conclusion, the proposed hand gesture recognition system is more than a technical demonstration—it is a step toward natural, accessible, and intelligent interaction between humans and machines. Its scalable, real-time design and the modularity of its components make it a strong foundation for future research and deployment. As society continues to embrace hands-free and inclusive interfaces, systems like this will play a vital role in shaping the digital experiences of tomorrow.

APPENDIX A – SOURCE CODE

Dataset Creation:

```
import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np
import math
import time
cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)

offset = 20
imgSize = 224
counter = 0

folder = "C:/Users/sandeep cahskar/Desktop/testing/updated_data/A"

while True:
    success, img = cap.read()
    if not success:
        print("Failed to capture image")
        break

    hands, img = detector.findHands(img)

    if hands:
        try:
            hand = hands[0]
            x, y, w, h = hand['bbox']

            # Ensure crop boundaries are within the image
            y1 = max(0, y - offset)
            y2 = min(img.shape[0], y + h + offset)
            x1 = max(0, x - offset)
            x2 = min(img.shape[1], x + w + offset)

            imgCrop = img[y1:y2, x1:x2]

            if imgCrop.size == 0:  # Check for empty crop
                print("Crop resulted in an empty image. Skipping.")
                continue

            imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
            aspectRatio = h / w

            if aspectRatio > 1:
                k = imgSize / h
                wCal = math.ceil(k * w)
                imgResize = cv2.resize(imgCrop, (wCal, imgSize))
                wGap = math.ceil((imgSize - wCal) / 2)
                imgWhite[:, wGap:wCal + wGap] = imgResize
            else:
                k = imgSize / w
                hCal = math.ceil(k * h)
                imgResize = cv2.resize(imgCrop, (imgSize, hCal))
                hGap = math.ceil((imgSize - hCal) / 2)
                imgWhite[hGap:hCal + hGap, :] = imgResize

            cv2.imshow("ImageCrop", imgCrop)
            cv2.imshow("ImageWhite", imgWhite)
        except Exception as e:
            print("An error occurred:", e)
    else:
        cv2.putText(img, "No hand detected", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,
0, 0), 2)

    cv2.imshow("Image", img)

    key = cv2.waitKey(1)
    if key & 0xFF == ord('q'):
        break
    if key == ord("s"):
        counter += 1
        cv2.imwrite(f'{folder}/Image_{time.time()}.jpg', imgWhite)
        print(counter)

cap.release()
cv2.destroyAllWindows()
```

Model:

```
import cv2
import torch
import numpy as np
from torchvision import transforms
from PIL import Image
import timm
import torch.nn as nn

# Load the class labels (ASL alphabet in your case)
class_names = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',  'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U',
'V', 'W', 'X', 'Y', 'Z']

# Define the same transformations used during training
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

# Define the hybrid model
class EfficientNetDeiTHybrid(nn.Module):
    def _init_(self, num_classes=26):
        super(EfficientNetDeiTHybrid, self)._init_()

        # EfficientNet
        self.efficientnet = timm.create_model('efficientnet_b0', pretrained=True, num_classes=0)
        effnet_out_features = self.efficientnet.num_features

        # DeiT
        self.deit = timm.create_model('deit_base_patch16_224', pretrained=True, num_classes=0)
        deit_out_features = self.deit.embed_dim

        # Final classifier
        self.fc = nn.Sequential(
            nn.Linear(effnet_out_features + deit_out_features, 512),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(512, num_classes)
        )

    def forward(self, x):
        effnet_features = self.efficientnet(x)
        deit_features = self.deit(x)
        combined_features = torch.cat((effnet_features, deit_features), dim=1)
        output = self.fc(combined_features)
        return output

# Load the model
model = EfficientNetDeiTHybrid(num_classes=26)
model.load_state_dict(torch.load('/kaggle/working/efficientnet_deit_best.pth'))
model.eval()
model = model.to('cuda' if torch.cuda.is_available() else 'cpu')

# Function to predict sign from frame
def predict_sign(frame, model, transform, device):
    # Convert the frame (OpenCV image) to PIL image
    pil_image = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))

    # Apply transformations
    image_tensor = transform(pil_image).unsqueeze(0).to(device)

    # Perform inference
    with torch.no_grad():
        outputs = model(image_tensor)
        _, preds = torch.max(outputs, 1)

    # Get the predicted class
    predicted_class = class_names[preds.item()]
    return predicted_class

# Start real-time webcam capture using OpenCV
def run_realtime_detection(model):
    # Initialize webcam (0 is the default webcam index)
    cap = cv2.VideoCapture(0)

    if not cap.isOpened():
        print("Error: Could not open webcam.")
        return

    # Loop to capture frames from the webcam
    while True:
        # Capture frame-by-frame
        ret, frame = cap.read()
        if not ret:
            print("Error: Could not read frame.")
            break

        # Predict the ASL sign from the frame
        predicted_sign = predict_sign(frame, model, transform, device)

        # Display the result on the frame
        cv2.putText(frame, f'Predicted Sign: {predicted_sign}', (10, 40),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)

        # Display the frame
        cv2.imshow('ASL Sign Detection', frame)
        # Press 'q' to quit
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    # Release the webcam and close OpenCV windows
    cap.release()
    cv2.destroyAllWindows()
# Run the real-time detection
run_realtime_detection(model)
```

Other Model:

```
import os
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
from tensorflow.keras.models import load_model

# Define dataset path
data_dir = 'C:/Users/sandeep cahskar/Desktop/testing/data'

# Parameters
img_height, img_width = 224, 224
batch_size = 32
epochs = 30

# Data augmentation and preprocessing
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    validation_split=0.2  # 20% for validation
)

train_generator = train_datagen.flow_from_directory(
    data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)

# Get class labels
class_labels = list(train_generator.class_indices.keys())
print("Class Labels:", class_labels)

# Load pre-trained MobileNetV2 model
base_model = MobileNetV2(input_shape=(img_height, img_width, 3), include_top=False,
weights='imagenet')
base_model.trainable = False  # Freeze the base model

# Define the model
model = Sequential([
    base_model,
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(len(class_labels), activation='softmax')
])
```

```
# Compile the model
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Callbacks for training
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, min_lr=1e-6)
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train the model
history = model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=epochs,
    callbacks=[lr_scheduler, early_stopping]
)

# Save the model after training
model_save_path = 'asl_16feb.h5'
model.save(model_save_path)
print(f"Model saved to {model_save_path}")

# Loading the saved model
loaded_model = load_model(model_save_path)
print("Model loaded successfully!")

# You can now use the loaded model for inference or further training
```

APPENDIX B – SCREENSHOTS

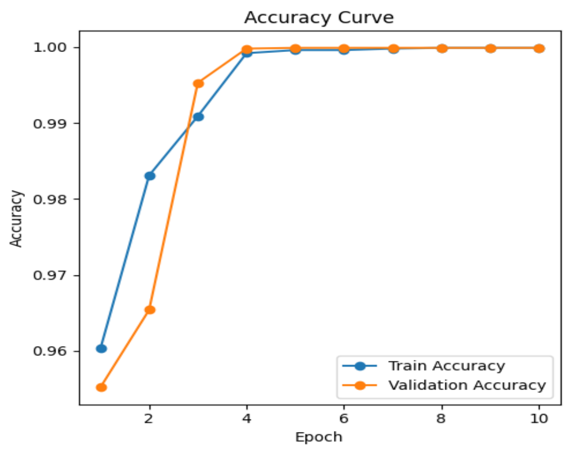


Fig. B1. Learning Curve of EfficientNet-DeIT

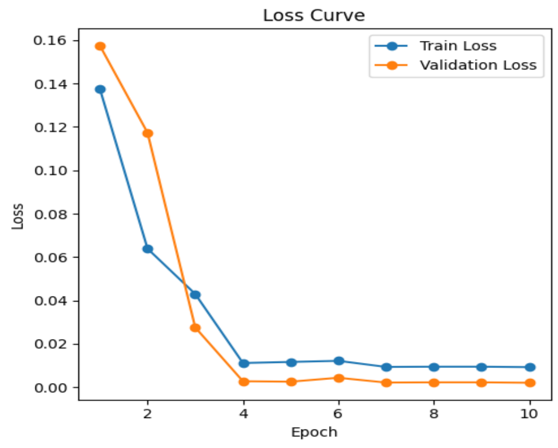


Fig. B2. Loss Curve of EfficientNet-DeIT

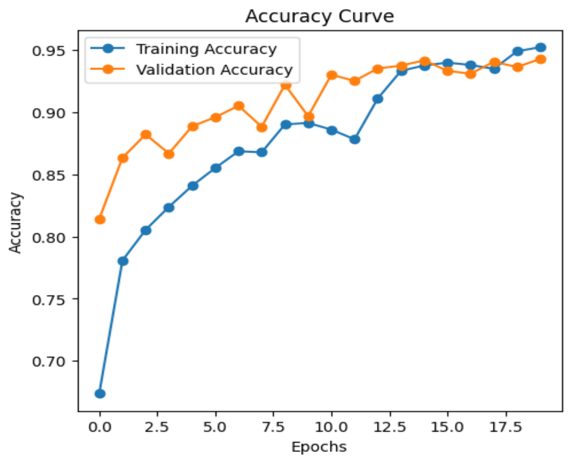


Fig. B3. Learning Curve of MobileNet

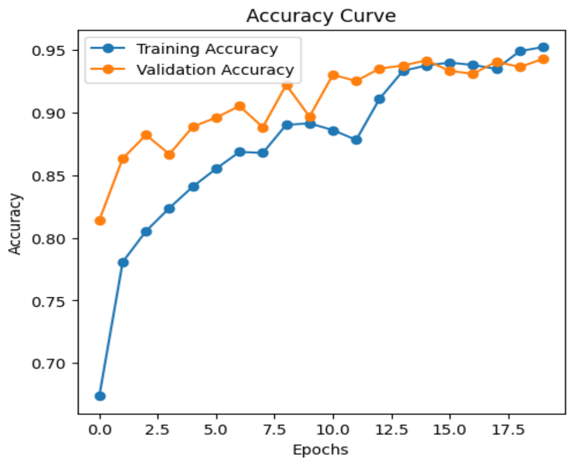


Fig. B4. Loss Curve of MobileNet

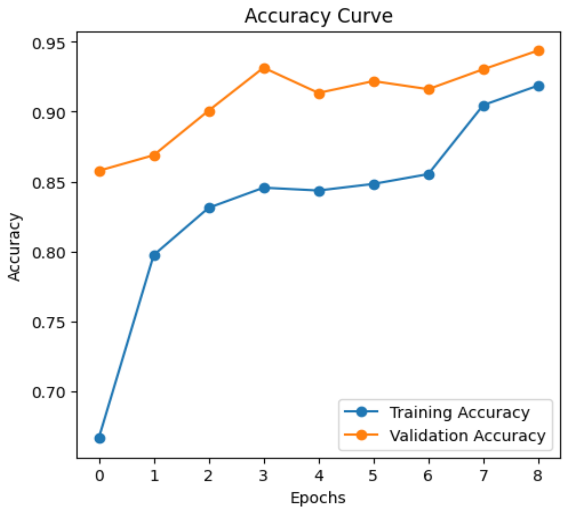


Fig. B5. Learning Curve of VGG16

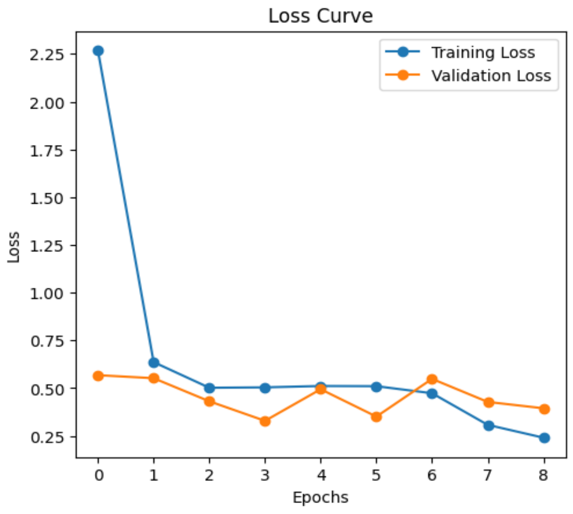


Fig. B6. Loss Curve of VGG16

REFERENCES

[1]S. Kausar and M. Y. Javed, ‘‘A survey on sign language recognition,’’ in Proc. Frontiers Inf. Technol., Islamabad, Pakistan, Dec. 2011, pp. 95–98.

[2] M. B. Waldron and S. Kim, ‘‘Isolated ASL sign recognition system for deaf persons,’’ IEEE Trans. Rehabil. Eng., vol. 3, no. 3, pp. 261–271, Sep. 1995.

[3] C. Chansri and J. Srinonchat, ‘‘Hand gesture recognition for thai sign language in complex background using fusion of depth and color video,’’ Procedia Comput. Sci., vol. 86, pp. 257–260, Jan. 2016.

[4] M.-H. Yang, N. Ahuja, and M. Tabb, ‘‘Extraction of 2D motion trajectories and its application to hand gesture recognition,’’ IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 8, pp. 1061–1074, Aug. 2002.

[5] T. Starner, J. Weaver, and A. Pentland, ‘‘Real-time American sign language recognition using desk and wearable computer based video,’’ IEEE Trans. Pattern Anal. Mach. Intell., vol. 20, no. 12, pp. 1371–1375, Dec. 1998.

[6] D. Kelly, J. Mc Donald, and C. Markham, ‘‘Continuous recognition of motion based gestures in sign language,’’ in Proc. IEEE 12th Int. Conf. Comput. Vis. Workshops, ICCV Workshops, Kyoto, Japan, Sep. 2009, pp. 1073–1080.

[7] M. Alhussein and G. Muhammad, ‘‘Voice pathology detection using deep learning on mobile healthcare framework,’’ IEEE Access, vol. 6, pp. 41034–41041, 2018.

[8] G. Muhammad, M. F. Alhamid, and X. Long, ‘‘Computing and processing on the edge: Smart pathology detection for connected healthcare,’’ IEEE Netw., vol. 33, no. 6, pp. 44–49, Nov./Dec. 2019.

[9] X. Zhang, X. Chen, Y. Li, V. Lantz, K. Wang, and J. Yang, ‘‘A framework for hand gesture recognition based on accelerometer and EMG sensors,’’ IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, vol. 41, no. 6, pp. 1064–1076, Nov. 2011.

[10] V. E. Kosmidou and L. J. Hadjileontiadis, ‘‘Sign language recognition using intrinsic-mode sample entropy on sEMG and accelerometer data,’’ IEEE Trans. Biomed. Eng., vol. 56, no. 12, pp. 2879–2890, Dec. 2009.

[11] T. D. Bui and L. T. Nguyen, ‘‘Recognizing postures in vietnamese sign language with MEMS accelerometers,’’ IEEE Sensors J., vol. 7, no. 5, pp. 707–712, May 2007.

[12] J. Huang et al., ‘‘Sign language recognition using convolutional neural networks,’’ in Proc. Eur. Conf. Comput. Vis., Zürich, Switzerland, Sep. 2014, pp. 572–578.

[13] A. Tang, K. Lu, Y. Wang, J. Huang, and H. Li, ‘‘A real-time hand posture recognition system using deep neural networks,’’ ACM Trans. Intell. Syst. Technol., vol. 6, no. 2, pp. 1–23, May 2015.

[14] A. Mohanty, S. S. Rambhatla, and R. R. Sahay, ‘‘Deep gesture: Static hand gesture recognition using CNN,’’ in Proc. Comput. Vis. Image Process., Roorkee, India, Sep. 2017, pp. 449–461.

[15] J. C. Núñez, R. Cabido, J. J. Pantrigo, A. S. Montemayor, and J. F. Vélez, ‘‘Convolutional neural networks and long short-term memory for skeletonbased human activity and hand gesture recognition,’’ Pattern Recognit., vol. 76, pp. 80–94, Apr. 2018.

[16] N. Kumaran, M. Sri Anurag, and M. Sampath, "Hand Gesture Recognition System Using Transfer Learning," Journal of Computer Research and Engineering Sciences, vol. 4, no. 1, Paper 8,

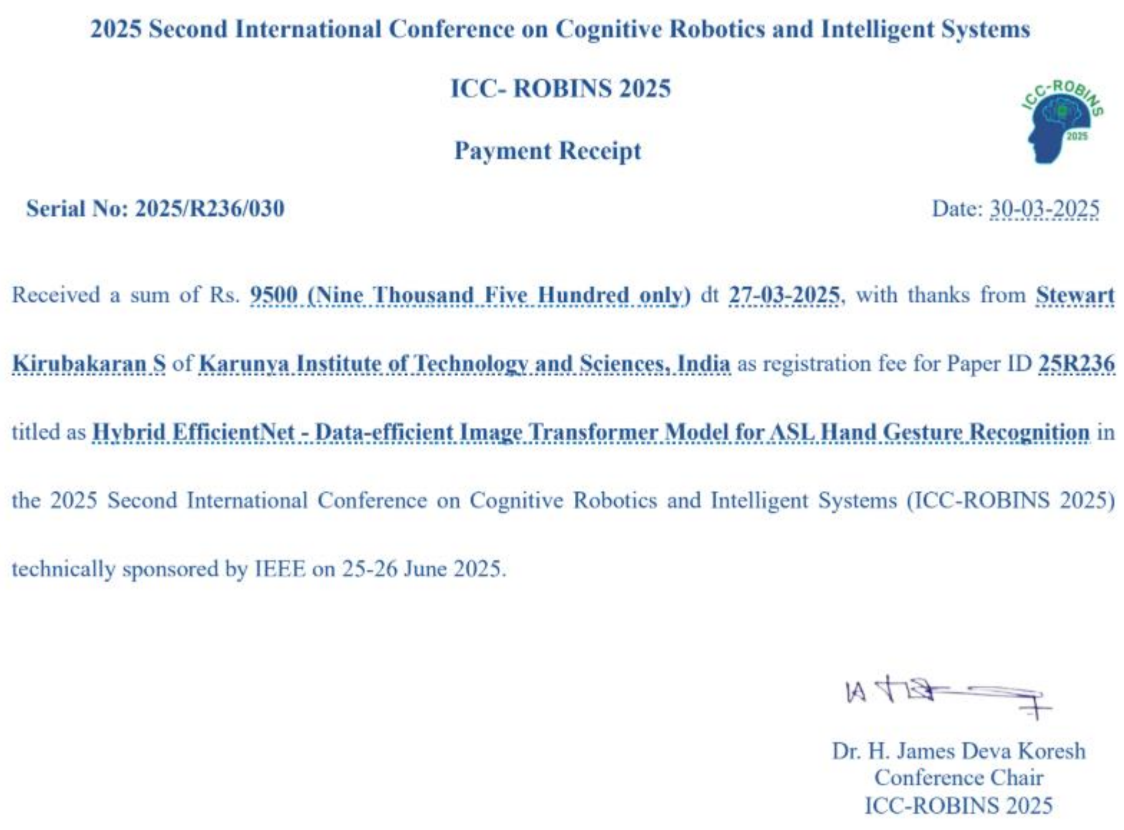
Project Report Evaluation form

Criteria	Excellent (5)	Good (4)	Satisfactory (3)	Needs Improvement (2)	Poor (1)
Report Formatting	Fully formatted as per guidelines	Minor formatting issues	Moderate formatting issues	Several formatting issues	Not formatted
Grammar and Language	No grammatical/spelling errors	Few minor errors	Noticeable errors	Frequent errors	Poor grammar, hard to understand
Referencing, Tables & Figures	All cited properly, visuals are clear	Minor citation/visual issues	Some missing citations or unclear visuals	Many missing or incorrect citations	No citations; visuals poorly presented
Technical Content Quality	In-depth, innovative, and well-explained with Original content	Clear and relevant	Meets basic expectations	Lacks depth or clarity	Incomplete or incorrect technical info

Name	
Reg. No	
Report Correctly formatted	
No Grammatical Errors	
All References are Cited Inside; Tables and Figures Represented Properly	
Technical Content	
Total	

Guide Signature

PROJECT OUTCOME:



2025 Second International Conference on Cognitive Robotics and Intelligent Systems

ICC- ROBINS 2025

Payment Receipt

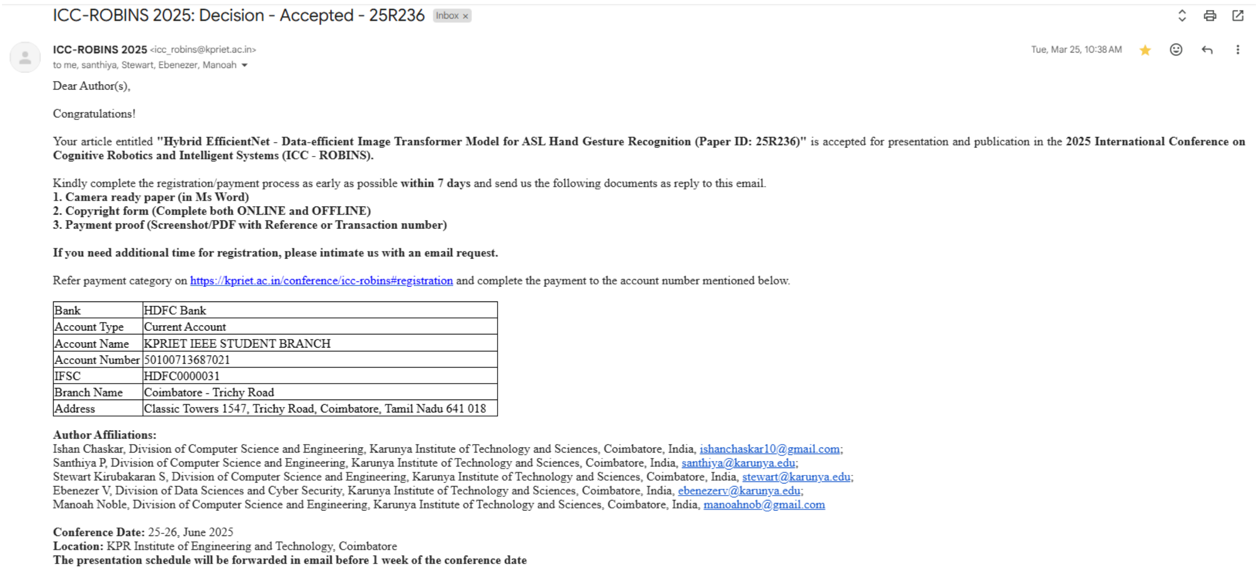
Serial No: 2025/R236/030

Date: 30-03-2025

Received a sum of Rs. 9500 (Nine Thousand Five Hundred only) dt 27-03-2025, with thanks from Stewart Kirubakaran S of Karunya Institute of Technology and Sciences, India as registration fee for Paper ID 25R236 titled as Hybrid EfficientNet - Data-efficient Image Transformer Model for ASL Hand Gesture Recognition in the 2025 Second International Conference on Cognitive Robotics and Intelligent Systems (ICC-ROBINS 2025) technically sponsored by IEEE on 25-26 June 2025.

[illegible]

Dr. H. James Deva Koresh
Conference Chair
ICC-ROBINS 2025



ICC-ROBINS 2025: Decision - Accepted - 25R236

ICC-ROBINS 2025 <icc_robins@kpriet.ac.in>
to me, santhiya, Stewart, Ebenezer, Manoah
Tue, Mar 25, 10:38 AM

Dear Author(s),

Congratulations!

Your article entitled **"Hybrid EfficientNet - Data-efficient Image Transformer Model for ASL Hand Gesture Recognition (Paper ID: 25R236)"** is accepted for presentation and publication in the **2025 International Conference on Cognitive Robotics and Intelligent Systems (ICC - ROBINS).**

Kindly complete the registration/payment process as early as possible **within 7 days** and send us the following documents as reply to this email.
1. Camera ready paper (in Ms Word)
2. Copyright form (Complete both ONLINE and OFFLINE)
3. Payment proof (Screenshot/PDF with Reference or Transaction number)

If you need additional time for registration, please intimate us with an email request.

Refer payment category on https://kpriet.ac.in/conference/icc-robins#registration and complete the payment to the account number mentioned below.

Bank	HDFC Bank
Account Type	Current Account
Account Name	KPRIET IEEE STUDENT BRANCH
Account Number	50100713687021
IFSC	HDFC0000031
Branch Name	Coimbatore - Trichy Road
Address	Classic Towers 1547, Trichy Road, Coimbatore, Tamil Nadu 641 018

Author Affiliations:
Ishan Chaskar, Division of Computer Science and Engineering, Karunya Institute of Technology and Sciences, Coimbatore, India, ishanchaskar10@gmail.com;
Santhiya P, Division of Computer Science and Engineering, Karunya Institute of Technology and Sciences, Coimbatore, India, santhiya@karunya.edu;
Stewart Kirubakaran S, Division of Computer Science and Engineering, Karunya Institute of Technology and Sciences, Coimbatore, India, stewart@karunya.edu;
Ebenezer V, Division of Data Sciences and Cyber Security, Karunya Institute of Technology and Sciences, Coimbatore, India, ebenezerv@karunya.edu;
Manoah Noble, Division of Computer Science and Engineering, Karunya Institute of Technology and Sciences, Coimbatore, India, manoahnob@gmail.com

Conference Date: 25-26, June 2025
Location: KPR Institute of Engineering and Technology, Coimbatore
The presentation schedule will be forwarded in email before 1 week of the conference date