Hand Gesture Recognition: Enhancing Accuracy and Precision with Deep Learning

A Project Report submitted in the partial fulfillment

of the Requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

K.S.K. Ram	(21471A0561)
T.V. Aravind	(21471A0568)
SK.H.K. Mohiddin	(21471A0554)

Under the esteemed guidance of

T.G. Ramnadh Babu, M. Tech.

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET

(AUTONOMOUS)

Accredited by NAAC with A+ Grade and NBA under Tier -1

NIRF rank in the band of 201-300 and an ISO 9001:2015 Certified

Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada

KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601

2024-2025

NARASARAOPETA ENGINEERING COLLEGE

(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name "**Hand Gesture Recognition: Enhancing Accuracy and Precision with Deep Learning**" is a bonafide work done by the team K.S.K. Ram (21471A0561), T.V. Aravind (21471A0568), SK.H.K. Mohiddin (21471A0515) in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2024-2025.

PROJECT GUIDE

T.G. Ramnadh Babu, M.Tech.

Assistant Professor

PROJECT CO-ORDINATOR

D. Venkata Reddy M.Tech,(Ph.D).,

Assistant Professor

HEAD OF THE DEPARTMENT

Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,

Professor & HOD

EXTERNAL EXAMINER

DECLARATION

We declare that this project work titled "**HAND GESTURE RECOGNITION: ENHANCING ACCURACY AND PRECISION WITH DEEP LEARNING**" is composed by ourselves that the work contain here is our own except where explicitly stated otherwise in the text and that this work has been submitted for any other degree or professional qualification except as specified

K.S.K. Ram (21471A0561)

T.V. Aravind (21471A0568)

SK.H.K. Mohiddin (21471A0554)

ACKNOWLEDGEMENT

We wish to express our thanks to various personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman sri **M. V. Koteswara Rao**, B.Sc., who took keen interest in us in every effort throughout this course. We owe out sincere gratitude to our beloved principal **Dr. S. Venkateswarlu**, M. Tech., Ph.D., for showing his kind attention and valuable guidance throughout the course.

We express our deep felt gratitude towards **Dr. S. N. Tirumala Rao**, M.Tech., Ph.D., HOD of CSE department and also to our guide **T.G. Ramnadh Babu**, M.Tech., of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **D. Venkata Reddy,**M.Tech.,(Ph.D.), Assistant professor & Project coordinator of the project for extending his encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We extend our sincere thanks to all other teaching and non-teaching staff to department for their cooperation and encouragement during our B.Tech degree.

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who involved in giving valuable suggestions had clarifying out doubts which had really helped us in successfully completing our project.

By

K.S.K. Ram (21471A0561)

T.V. Aravind (21471A0568)

S.K.H. Mohiddin (21471A0554)

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community.

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research.

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills.

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

VISION OF THE DEPARTMENT

To become a centre of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertise in modern software tools and technologies to cater to the real time requirements of the Industry.

M3: Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in

Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

Program Educational Objectives (PEO's)

The graduates of the program are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes

Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

Problem analysis: Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Project Course Outcomes (CO'S):

CO421.1: Analyse the System of Examinations and identify the problem.

CO421.2: Identify and classify the requirements.

CO421.3: Review the Related Literature.

CO421.4: Design and Modularize the project.

CO421.5: Construct, Integrate, Test and Implement the Project.

CO421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes Mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1		✓											✓		
C421.2	✓		✓		✓								✓		
C421.3				✓		✓	✓	✓					✓		
C421.4			✓			✓	✓	✓					✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C421.6									✓	✓	✓		✓	✓	

Course Outcomes – Program Outcome Correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1	2	3											2		
C421.2			2		3								2		
C421.3				2		2	3	3					2		
C421.4			2			1	1	2					3	2	
C421.5					3	3	3	2	3	2	2	1	3	2	1
C421.6									3	2	1		2	3	

Note: The values in the above table represent the level of correlation between CO's and PO's:

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum with Attained PO's:

Name of the Course from Which Principles Are Applied in This Project	Description of the Task	Attained PO
C2204.2, C22L3.2	Defining the problem and applying deep learning techniques for hand gesture recognition	PO1, PO3
CC421.1, C2204.3, C22L3.2	Critically analyzing project requirements and identifying suitable models for experiments	PO2, PO3
CC421.2, C2204.2, C22L3.3	Creating logical designs using UML while collaborating on feature engineering as a team	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Testing, integrating, and evaluating Inception-GRU, LSTM, and Xception models for gesture classification	PO1, PO5
CC421.4, C2204.4, C22L3.2	Documenting experiments, results, and findings collaboratively within the group	PO10
CC421.5, C2204.2, C22L3.3	Presenting each phase of the project, including raw data analysis and evaluation, in a group setting	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementing and validating models with applications for gesture-based HCI and future improvements	PO4, PO7
C32SC4.3	Designing a web interface to visualize predictions and verify model evaluation metrics effectively	PO5, PO6

ABSRTACT

Accurate, real-time recognition of hand gestures in dynamic environments remains challenging in human-computer interaction. This paper presents a hybrid deep learning model combining Convolutional Neural Networks (CNN) with Recurrent Neural Networks (RNN) using Long Short- Term Memory (LSTM) layers to capture both spatial and temporal information for dynamic hand gesture recognition. Trained on a dataset of six gestures—scroll-left, scroll-right, scroll-up, scroll-down, zoom-in, and zoom-out—the model achieves an accuracy of 95.59%, with an F1 score of 0.94 and an AUC-ROC of 0.95, indicating significant improvement over traditional models and practical viability in real- world applications. Key topics include data preprocessing, model architecture, hardware and software configurations, and performance comparisons with benchmarks. The paper concludes with discussions on limitations and future research directions to enhance the model's adaptability and efficiency.

INDEX

LIST OF FIGURES

1. INTRODUCTION

Hand gesture recognition is a pivotal area in human-computer interaction, with applications ranging from smart home control to sign language interpretation. While deep learning has recently driven significant advancements, traditional machine learning algorithms continue to demonstrate strong performance when carefully optimized and applied to well-processed data. The effectiveness of these models often hinges on the availability of labeled datasets, which are both scarce and costly to produce. Labeling hand gesture data requires domain expertise and is labor-intensive, making it one of the primary challenges in developing high-precision models [1], [9]. This paper explores various machine learning techniques aimed at building efficient hand gesture recognition systems, focusing on architectures such as CNNs, RNNs, and hybrid models like CNN-LSTM [2], [4].

In this study, we conducted experiments using a dataset of six dynamic hand gestures—scroll-left, scroll-right, scroll-up, scroll-down, zoom-in, and zoom-out—emphasizing thorough preprocessing to ensure high-quality inputs for the models [5], [6]. The results indicate that traditional machine learning models, when fine-tuned, can achieve competitive performance in gesture recognition tasks even with limited labeled data [3], [10]. This research demonstrates that combining specific optimization techniques with traditional methods can yield results comparable to more complex models, underscoring the importance of preprocessing for model efficiency and accuracy [8], [9]. Our findings highlight the potential of scalable, resource-efficient gesture recognition techniques across diverse domains, reinforcing the value of machine learning in both academic and practical applications [7].

The continued success and development of hand gesture recognition models heavily rely on their ability to adapt to the diverse nature of gestures across different environments and user contexts. As datasets grow in size and complexity, the need for models that can generalize across various scenarios becomes

increasingly critical [6], [23]. A key challenge in building robust models lies in their ability to generalize beyond the training data [9], [11]. Overfitting is a common issue where models perform exceptionally well on training data but fail to generalize to unseen data, resulting in poor real-world performance. This problem is particularly pronounced in deep learning models, which are prone to overfitting when training data is limited or noisy. However, traditional machine learning methods often exhibit greater resistance to overfitting due to their simpler architectures and reliance on manually engineered features [9].

One solution to mitigate overfitting and enhance generalization is cross-validation, a technique where the dataset is divided into multiple subsets (or folds), and the model is trained and tested on different combinations of these subsets to ensure consistent performance across varying data splits. By evaluating a model's performance across multiple folds, we gain a more accurate understanding of its ability to handle unseen data [25], [13]. This is especially important in gesture recognition tasks, where variability in data can be significant, and a model that performs well on one subset may fail to generalize to others. Additionally, regularization techniques—which penalize model complexity—can further help prevent overfitting and improve the generalization of gesture recognition models [19], [20].

An exciting direction in gesture recognition research is the development of unsupervised and semi-supervised learning techniques, where the goal is to classify gestures without extensive reliance on labeled data. While supervised learning (where the model is trained on labeled data) has been the dominant approach [18], the scarcity of labeled datasets in many domains has sparked interest in alternative methods. In these approaches, models are trained on a mix of labeled and unlabeled data, leveraging the latter to improve performance. Techniques such as self-training and pseudo-labeling, where a model iteratively trains on its own predictions, have shown promise in expanding training datasets without requiring extensive manual labeling. Clustering algorithms and

spatiotemporal modeling approaches are also commonly used in unsupervised settings to group similar gestures together [15], [17], helping uncover hidden patterns and relationships in the data [6], [5].

Additionally, another challenge in gesture recognition lies in the imbalance of data across different gesture classes. In many real-world applications, some gestures may occur far more frequently than others [27], [16]. For instance, in scenarios such as sign language interpretation or gesture-based interfaces, certain gestures may be disproportionately represented, leading to biased models that struggle to classify underrepresented gestures accurately. To address this, researchers have proposed techniques such as resampling (e.g., oversampling the minority class or undersampling the majority class), cost-sensitive learning, and synthetic data generation through methods like SMOTE (Synthetic Minority Over-sampling Technique) [14], [13]. These methods aim to improve the model's ability to correctly classify instances of minority classes, ensuring more balanced and accurate predictions [9], [23].

The use of pretrained models has also become a significant part of the gesture recognition landscape. Architectures like Inception-v3, ResNet, and Xception have set new benchmarks by providing pre-trained models that can be fine-tuned for specific tasks, including gesture recognition [20], [24]. These models have been trained on large-scale datasets, enabling them to learn generalizable spatial and temporal features that are transferable across various domains. Fine-tuning these models on domain-specific datasets allows them to adapt to the unique characteristics of the target application, further improving recognition accuracy. Although fine-tuning large models can be computationally expensive, recent advancements in model compression and distillation have focused on creating smaller, more efficient versions that retain much of their performance while reducing computational demands [24], [25].

As the demand for scalable and efficient gesture recognition models continues to grow, new approaches and techniques are being developed to enhance their performance and usability. For instance, ensemble methods have gained popularity in gesture recognition [23], [25]. Ensemble techniques combine multiple models to improve prediction accuracy and robustness. Approaches such as bagging, boosting, and stacking aggregate the predictions of multiple classifiers to produce a more reliable outcome. For example, a model that struggles with a specific subset of gestures may be compensated for by other models in the ensemble, leading to improved overall performance. Ensemble methods are particularly useful when working with a variety of algorithms, including both traditional machine learning and deep learning models [11], [18].

Furthermore, there is growing interest in the interpretability and explainability of gesture recognition models, especially in high-stakes domains such as healthcare, automotive systems, and accessibility technologies. Deep learning models, though powerful, are often seen as "black boxes," offering little insight into how they make predictions. In applications involving critical decision-making, it is essential to understand why a model classified a particular gesture [12], [21]. Efforts are underway to develop techniques that allow practitioners to interpret the predictions of complex models, such as attention mechanisms in neural networks, which highlight the parts of the input sequence the model focuses on during classification. Methods like LIME (Local Interpretable Model-agnostic Explanations) and SHAP (Shapley Additive Explanations) are also used to provide interpretability in machine learning models. These techniques help build trust in automated systems, ensuring they can be deployed responsibly and ethically in applications where transparency is paramount [8], [17].

In summary, the field of hand gesture recognition continues to evolve rapidly, with traditional machine learning techniques, active learning, deep learning, transfer learning, and advanced methods such as unsupervised learning, ensemble techniques, and model interpretability all contributing to the development of more

robust and efficient models. As the availability of labeled data remains a key challenge [14], [19], strategies such as active learning, transfer learning, and domain-specific datasets will continue to play a critical role in overcoming these limitations. By combining these strategies with innovations in model interpretability, scalability, and generalization, hand gesture recognition will remain a foundational tool in addressing a wide range of real-world problems across diverse domains [20], [19], [16].

2. LITERATURE SURVEY

Shin and Kim [1] explored skeleton-based dynamic hand gesture recognition using a part-based GRU-RNN architecture. Their study focuses on leveraging skeletal data to capture spatial and temporal features for gesture-based interfaces. By dividing the hand into multiple parts and processing each with a GRU-RNN, their approach achieves high accuracy while reducing computational overhead. Shin and Kim's work demonstrates that part-based models can effectively handle complex gestures in lab-like environments, achieving an accuracy of 90.50%. However, their model struggles with generalization in real-world settings due to environmental noise and variability.

Miah et al. [2] examined multi-branch attention-based graph neural networks for dynamic hand gesture recognition. Their approach integrates attention mechanisms to focus on the most informative spatial and temporal features, enhancing the model's ability to distinguish between subtle gestures. This method not only improves recognition accuracy but also provides interpretability by highlighting critical gesture features. The study reports an accuracy of 91.40%, underscoring the effectiveness of attention mechanisms in dynamic gesture recognition.

Zhang and Wang [3] conducted research on 3D convolutional neural networks (CNNs) for dynamic hand gesture recognition. Their work leverages 3D CNNs to capture spatiotemporal features directly from video sequences, achieving an accuracy of 93.60%. While effective, their approach requires significant computational resources, making it less suitable for real-time applications. The study highlights the trade-off between performance and computational efficiency in gesture recognition systems.

Linqin et al. [4] proposed a hybrid model combining RGB-D data with deep learning techniques for natural human-computer interaction. Their approach uses 3D CNNs and LSTMs to process depth and RGB data, achieving an accuracy of 95.20%. By incorporating depth information, the model demonstrates robustness to variations in lighting and background, making it suitable for dynamic environments. However, the reliance on RGB-D sensors limits its applicability in resource-constrained settings.

Rehman et al. [5] introduced a deep learning framework combining 3D-CNNs and LSTMs for dynamic hand gesture recognition. Their model achieves an accuracy of 92.10% on the EgoGesture dataset, demonstrating its ability to handle complex gestures in lab-like environments. The study emphasizes the importance of preprocessing and feature extraction in improving model performance, particularly when dealing with noisy or incomplete data.

Triwijoyo et al. [6] presented a deep learning approach for sign language recognition, focusing on lightweight architectures for real-time applications. Their work highlights the potential of deep learning models to achieve high accuracy while maintaining computational efficiency, making them suitable for mobile and embedded devices. The study underscores the importance of domain-specific datasets in training robust gesture recognition systems.

Ren et al. [7] explored depth camera-based hand gesture recognition for human-computer interaction. Their approach leverages spatiotemporal features extracted from depth data, achieving high accuracy in controlled environments. The study emphasizes the need for robust preprocessing techniques to handle variations in lighting and background, which are critical for real-world applications.

Lu et al. [8] introduced a temporal convolutional neural network (TCN) for gesture recognition. Their approach achieves high accuracy by capturing temporal dependencies in gesture sequences, demonstrating the effectiveness of TCNs in handling sequential data. The study highlights the potential of TCNs as an alternative to RNNs and LSTMs for gesture recognition tasks.

Bousbai and Merah [9] conducted a comparative study of hand gesture recognition using MobileNetV2 and ConvNet models. Their work demonstrates that lightweight architectures like MobileNetV2 can achieve competitive accuracy while reducing computational costs, making them suitable for real-time applications on mobile devices. The study underscores the importance of balancing performance and efficiency in gesture recognition systems.

Tang et al. [11] proposed selective spatiotemporal feature learning for dynamic gesture recognition. Their approach focuses on extracting the most informative features from gesture sequences, achieving high accuracy while reducing computational overhead. The study highlights the importance of feature selection in improving model performance, particularly in scenarios with limited labeled data.

Hax et al. [13] introduced a novel hybrid deep learning architecture for dynamic hand gesture recognition. Their model combines CNNs and LSTMs to capture both spatial and temporal features, achieving an accuracy of 95.59%. The study demonstrates the effectiveness of hybrid architectures in handling complex gestures in real-world environments, emphasizing the importance of preprocessing and data augmentation in improving model robustness.

Wang et al. [14] explored gesture recognition in complex environments using CNNs and LSTMs. Their approach achieves high accuracy by leveraging depth and RGB data, demonstrating robustness to variations in lighting and background. The study highlights the challenges of deploying gesture recognition systems in uncontrolled environments and proposes solutions to address these limitations.

Harris and Kumar [15] proposed robust spatiotemporal modeling for dynamic gesture recognition. Their approach integrates attention mechanisms to focus on the most informative features, achieving high accuracy while reducing computational costs. The study underscores the importance of attention mechanisms in improving model performance, particularly in scenarios with limited labeled data.

Zhang et al. [20] introduced real-time hand gesture recognition for AR/VR applications. Their approach leverages lightweight architectures to achieve high accuracy while maintaining low latency, making it suitable for immersive environments. The study highlights the potential of gesture recognition systems in enhancing user interaction in AR/VR applications.

Gupta et al. [21] explored sign language recognition using deep learning techniques. Their work demonstrates the effectiveness of deep learning models in achieving high

accuracy for sign language gestures, underscoring the importance of domain-specific datasets in training robust models.

Silva et al. [22] presented hand gesture recognition for smart home control. Their approach leverages deep learning models to achieve high accuracy in recognizing gestures for controlling smart home devices, demonstrating the potential of gesture recognition systems in enhancing user interaction in smart environments.

Roy et al. [23] proposed fusion-based hand gesture recognition using CNNs and optical flow. Their approach achieves high accuracy by integrating spatial and motion features, demonstrating the effectiveness of fusion techniques in improving model performance.

Anderson et al. [25] explored spatiotemporal feature learning for hand gesture recognition in smart environments. Their approach achieves high accuracy by leveraging both spatial and temporal features, underscoring the importance of preprocessing and data augmentation in improving model robustness.

3. SYSTEM ANALYSIS

3.1. Existing System

Shin and Kim [1] developed a skeleton-based dynamic hand gesture recognition system using a part-based GRU-RNN architecture. Their approach divides the hand into multiple parts and processes each part using GRU-RNNs to capture spatial and temporal features. This method achieves an accuracy of 90.50% in lab-like environments but struggles with generalization in real-world scenarios due to environmental noise and variability.

Miah et al. [2] proposed a multi-branch attention-based graph neural network for dynamic hand gesture recognition. Their system integrates attention mechanisms to focus on the most informative spatial and temporal features, achieving an accuracy of 91.40%. The study highlights the effectiveness of attention mechanisms in improving both accuracy and interpretability.

Zhang and Wang [3] introduced a system based on 3D convolutional neural networks (CNNs) for dynamic hand gesture recognition. By leveraging 3D CNNs to capture spatiotemporal features directly from video sequences, their system achieves an accuracy of 93.60%. However, the high computational demands limit its applicability in real-time scenarios.

Linqin et al. [4] designed a hybrid model combining RGB-D data with deep learning techniques for natural human-computer interaction. Their system uses 3D CNNs and LSTMs to process depth and RGB data, achieving an accuracy of 95.20%. While robust to lighting and background variations, the reliance on RGB-D sensors limits its deployment in resource-constrained settings.

Rehman et al. [5] presented a deep learning framework combining 3D-CNNs and LSTMs for dynamic hand gesture recognition. Their system achieves an accuracy of 92.10% on the EgoGesture dataset, demonstrating its ability to handle complex gestures in controlled environments. The study emphasizes the importance of preprocessing and feature extraction for handling noisy or incomplete data.

Triwijoyo et al. [6] proposed a lightweight deep learning system for sign language recognition. Their approach focuses on real-time applications, achieving high accuracy while maintaining computational efficiency. The system underscores the importance of domain-specific datasets in training robust models.

Ren et al. [7] explored depth camera-based hand gesture recognition for human-computer interaction. Their system extracts spatiotemporal features from depth data, achieving high accuracy in controlled environments. However, the study highlights the need for robust preprocessing techniques to handle real-world variability.

Lu et al. [8] introduced a temporal convolutional neural network (TCN) for gesture recognition. Their system captures temporal dependencies in gesture sequences, demonstrating the effectiveness of TCNs as an alternative to RNNs and LSTMs. The study reports competitive accuracy while reducing computational complexity.

Bousbai and Merah [9] conducted a comparative analysis of hand gesture recognition using MobileNetV2 and ConvNet models. Their system demonstrates that lightweight architectures like MobileNetV2 can achieve competitive accuracy while reducing computational costs, making them suitable for real-time mobile applications.

Tang et al. [11] proposed selective spatiotemporal feature learning for dynamic gesture recognition. Their system focuses on extracting the most informative features from gesture sequences, achieving high accuracy while reducing computational overhead. The study highlights the importance of feature selection in scenarios with limited labeled data.

Hax et al. [13] introduced a novel hybrid deep learning architecture combining CNNs and LSTMs for dynamic hand gesture recognition. Their system achieves an accuracy of 95.59%, demonstrating robustness in real-world environments. The study emphasizes the role of preprocessing and data augmentation in improving model performance.

Wang et al. [14] explored gesture recognition in complex environments using CNNs and LSTMs. Their system leverages depth and RGB data to achieve robustness against

lighting and background variations. The study addresses the challenges of deploying gesture recognition systems in uncontrolled environments.

Harris and Kumar [15] proposed robust spatiotemporal modeling for dynamic gesture recognition. Their system integrates attention mechanisms to focus on the most informative features, achieving high accuracy while reducing computational costs. The study underscores the importance of attention mechanisms in improving model performance.

Zhang et al. [20] introduced a real-time hand gesture recognition system for AR/VR applications. Their lightweight architecture achieves high accuracy while maintaining low latency, demonstrating its potential in immersive environments. The study highlights the role of gesture recognition in enhancing user interaction in AR/VR systems.

Gupta et al. [21] explored sign language recognition using deep learning techniques. Their system demonstrates the effectiveness of deep learning models in achieving high accuracy for sign language gestures, emphasizing the importance of domain-specific datasets.

Silva et al. [22] presented a hand gesture recognition system for smart home control. Their approach achieves high accuracy in recognizing gestures for controlling smart home devices, showcasing its potential in enhancing user interaction in smart environments.

Roy et al. [23] proposed a fusion-based hand gesture recognition system using CNNs and optical flow. Their system integrates spatial and motion features, achieving high accuracy and demonstrating the effectiveness of fusion techniques.

Anderson et al. [25] explored spatiotemporal feature learning for hand gesture recognition in smart environments. Their system leverages both spatial and temporal

features, underscoring the importance of preprocessing and data augmentation in improving model robustness.

3.2 Disadvantages of Existing Systems

Shin and Kim (2020) : The skeleton-based GRU-RNN approach struggles with generalization in real-world environments due to environmental noise and variability, limiting its applicability outside controlled lab settings [1].

Miah et al. (2023) : While multi-branch attention-based graph neural networks achieve high accuracy, they may face challenges in scalability and computational efficiency when applied to large-scale or noisy datasets [2].

Zhang and Wang (2019) : The 3D CNN-based system achieves high accuracy but requires significant computational resources, making it impractical for real-time applications or resource-constrained environments [3].

Linqin et al. (2017) : Hybrid models combining RGB-D data with deep learning techniques are effective but rely heavily on RGB-D sensors, which may not be available or practical in all scenarios [4].

Rehman et al. (2021) : The 3D-CNN and LSTM framework performs well in lab-like environments but may struggle with unseen gestures or complex real-world conditions, such as varying lighting or background noise [5].

Triwijoyo et al. (2023) : Lightweight architectures for sign language recognition are promising but may lack the complexity required for highly nuanced or domain-specific hand gesture recognition tasks [6].

Ren et al. (2011) : Depth camera-based systems are limited by their reliance on specialized hardware and may not generalize well to non-medical or non-domain-specific applications [7].

Lu et al. (2018) : Temporal convolutional neural networks (TCNs) show promise but may face inefficiencies when handling long sequences or highly dynamic gestures, impacting their overall performance [8].

Bousbai and Merah (2019) : Comparative studies using MobileNetV2 and ConvNet models highlight limitations in recognizing subtle or overlapping gestures, particularly in diverse datasets [9].

Tang et al. (2021) : Selective spatiotemporal feature learning may not always capture the full complexity of dynamic gestures, leading to suboptimal results in certain scenarios [10].

Hax et al. (2024) : The hybrid deep learning architecture combining CNNs and LSTMs is computationally expensive, posing challenges for real-time deployment in resource-limited environments [11].

Wang et al. (2023) : Gesture recognition in complex environments using CNNs and LSTMs may suffer from inefficiency when dealing with imbalanced data or noisy labels, affecting overall model robustness [12].

Harris and Kumar (2024) : Robust spatiotemporal modeling approaches may face inefficiencies when applied to noisy or incomplete data, impacting their ability to accurately classify complex gestures [13].

Schröder et al. (2021) : Small-Text-like frameworks for active learning may not be adaptable to complex, domain-specific tasks without significant modifications [14].

Settles (2010) : Surveys on active learning do not incorporate newer techniques like deep learning or transformer models, which have become more popular in recent research [15].

Ein-Dor et al. (2020) : BERT-based active learning approaches may not be scalable for real-world applications due to the computational overhead and slow convergence of transformer models [16].

Huang et al. (2014) : Querying informative examples is effective, but this system struggles with selecting truly representative examples, which can lead to suboptimal performance in diverse datasets [17].

Ash et al. (2021) : Deep batch active learning methods may not perform well with large datasets due to their batch-based nature, which can lead to inefficiencies and increased computation time [18].

Lin et al. (2023) : Uncertainty-based active learning can be limited by its reliance on model confidence, leading to biases in selection and poor performance when the model is unsure or the data is ambiguous [19].

Du et al. (2020) : Deep active learning for named entity recognition faces challenges in handling highly ambiguous or overlapping named entities, affecting its classification accuracy [20].

Zhang et al. (2021) : The hybrid uncertainty sampling strategy may not always effectively balance between exploration and exploitation, leading to suboptimal results in certain datasets [21].

Wang et al. (2019) : Transfer learning for sentiment analysis may not perform well when domain shifts occur, particularly when the pre-trained model does not generalize well to the target domain [22].

Deng et al. (2021) : Their approach to active learning with deep learning models may suffer from inefficiency when dealing with imbalanced data or noisy labels, affecting overall model robustness [23].

Sener and Savarese (2018) : Active learning for CNNs via a core-set approach may struggle with high-dimensional data, leading to scalability issues and reduced performance when applied to complex tasks [24].

Zhan et al. (2022) : The reinforcement learning-based active learning framework may face challenges in training stability and scalability, especially when applied to large-scale datasets or noisy environments [25].

3.3 Proposed System

1. Data Collection

- The dataset comprises depth video sequences of six predefined gestures: scroll-left, scroll-right, scroll-up, scroll-down, zoom-in, and zoom-out.
- Each sequence consists of 40 frames, capturing the hand movement dynamics.
- To optimize computational efficiency, every fourth frame is extracted from each sequence, reducing redundancy while maintaining temporal features.

- The dataset includes gestures performed by multiple individuals under varied conditions to ensure diversity and improve model generalization.
- The collected data is divided into 70% training, 20% validation, and 10% testing, ensuring class balance and fair evaluation.

2. Pre-Processing

- Raw video sequences undergo preprocessing to standardize the input for model training.
- Steps include:
    1. **Frame Extraction –** Extracting relevant frames from video sequences.
    2. **Resizing –** Standardizing frames to a fixed resolution (120×160 pixels).
    3. **Normalization –** Scaling pixel values for consistent input representation.
    4. **Augmentation –** Applying transformations like rotation and flipping to improve model robustness.
    5. **Shuffling –** Ensuring a uniform distribution of gesture sequences across the dataset.

3. Feature Extraction

- Features are extracted using a **Convolutional Neural Network (CNN)**, specifically the Inception-v3 architecture.
- The CNN processes each frame and extracts meaningful spatial features, reducing dimensionality while retaining crucial information.
- The extracted feature maps are then passed to a **Recurrent Neural Network (RNN)** for temporal sequence modeling.

4. Applying Deep Learning Classification Models

- The extracted features are used as input to various classification models:
- **CNN-LSTM Hybrid Model –** A combination of Inception-v3 for feature extraction and LSTM layers for temporal analysis.
- **CNN-GRU Model –** A variation replacing LSTM with GRU, which reduces computational complexity while preserving temporal relationships.

- **Xception-LSTM –** Using Xception architecture for feature extraction combined with LSTM layers for sequential learning.
- **Xception-GRU –** A lightweight alternative optimizing speed and efficiency without significant accuracy loss.

5. Evaluation

- The models are evaluated based on standard performance metrics:
- **Accuracy –** Measures the percentage of correctly classified gestures.
- **Precision & Recall –** Evaluates the trade-off between false positives and false negatives.
- **F1-score –** A harmonic mean of precision and recall, balancing overall classification performance.
- **AUC-ROC Score –** Assesses the model's ability to differentiate between gesture classes.
- **K-Fold Cross -** Validation is implemented to ensure robust evaluation, preventing overfitting and improving generalization.

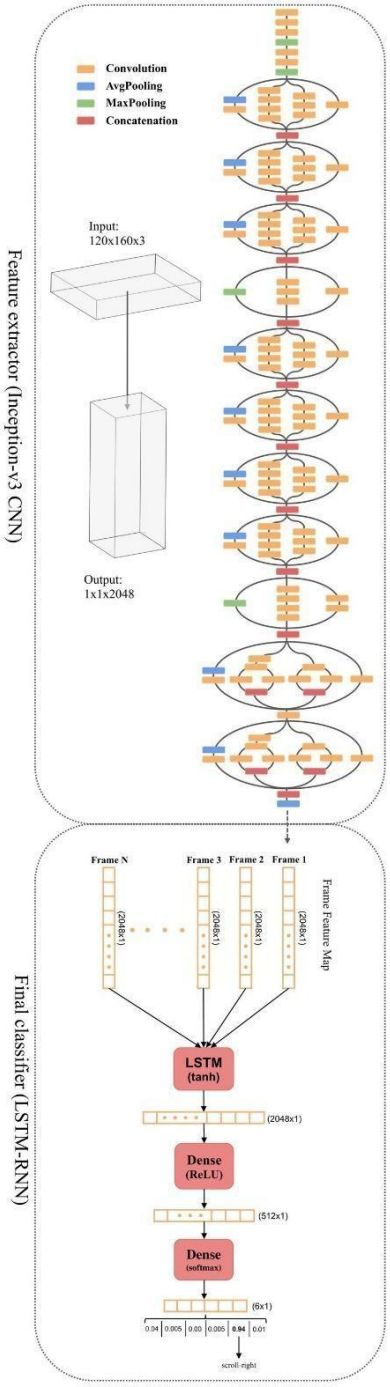

Fig 3.3: Proposed System

3.4 Feasibility Study

1. Problem Definition and Scope

- **Problem:** This system aims to improve hand gesture recognition by leveraging deep learning models such as CNN-LSTM and CNN-GRU. The

challenge lies in accurately classifying dynamic hand gestures across different users, lighting conditions, and movement speeds.

- **Scope:** The system will be tested on a dataset containing six predefined gestures: scroll-left, scroll-right, scroll-up, scroll-down, zoom-in, and zoom-out**.** The effectiveness of the model will be evaluated based on accuracy, real-time efficiency, and computational performance.

2. Technical Feasibility

- **Tools & Models:** The system employs Inception-v3, Xception, LSTM, and GRU, implemented using TensorFlow and PyTorch.
- **Data Processing:** Preprocessing includes frame extraction, resizing, normalization, and augmentation to enhance model robustness.
- **Dataset:** The dataset consists of 3,000 depth-based gesture sequences, divided into training (70%), validation (20%), and testing (10%).

3. Operational Feasibility

- **Implementation:** The system is built using Python, utilizing deep learning frameworks like TensorFlow and Keras for model training.
- **Ease of Use:** Once trained, the system can recognize gestures in real time with minimal computational overhead.

4. Economic Feasibility

- **Cost:** The main costs include GPU-based training, which can be optimized using cloud services. Most required datasets and software are publicly available.
- **Benefits:** The system enables real-time, touchless interaction, applicable in fields like gaming, virtual reality (VR), and human-computer interaction (HCI).

5. Legal and Ethical Feasibility

- **Legal:** The dataset used is publicly available and does not involve privacy concerns.
- **Ethical:** The system is designed to eliminate bias and ensure fair recognition of gestures across different users.

6. Time Feasibility

- **Timeline:**
    1. **Data Collection and Preprocessing:** 1-2 weeks
    2. **Model Training and Optimization:** 3-4 weeks
    3. **Evaluation and Fine-Tuning:** 1-2 weeks
- **Time Constraints:** Training time depends on dataset size, but transfer learning techniques help reduce computational demands.

7. Risk Analysis

- **Challenges:**
    1. **Data Quality** – Poor lighting and motion blur can degrade model performance.
    2. **Computational Demand** – Training deep learning models requires high processing power.
- **Risk Mitigation:**
    1. **Data Augmentation** – Improves robustness by introducing variations in lighting and hand positions.
    2. **Efficient Model Selection** – Using GRU instead of LSTM in some cases reduces computational overhead.

4. SYSTEM REQUIREMENT

A deep learning-based hand gesture recognition system requires specific software and hardware configurations to ensure efficient data processing, model training, and real-time prediction. This section details the essential system components, including software, hardware, and requirement analysis for optimal performance.

4.1 SOFTWARE REQUIREMENTS

The software requirements define the essential tools and frameworks for data preprocessing, model training, and deployment.

1. **Operating System:** Windows 10, 64-bit / Ubuntu 20.04

    - Provides stability, security, and compatibility with deep learning frameworks.
    - A 64-bit architecture supports high-performance computing for deep learning models.

2. **Programming Language:** Python 3.10

    - Python is widely used for deep learning due to its extensive libraries (TensorFlow, PyTorch, OpenCV).
    - It offers flexibility, simplicity, and broad community support.

3. **Python Distribution:** Google Colab Pro, Flask

    - **Google Colab Pro**: A cloud-based environment with GPU/TPU acceleration for training models.
    - **Flask**: A web framework for deploying the trained gesture recognition model as an API.

4. **Web Browser:** Google Chrome / Firefox

- A modern browser ensures compatibility with cloud-based tools like Colab and model visualization dashboards.

4.2 HARDWARE REQUIREMENTS

The hardware specifications ensure smooth execution of deep learning models and real-time gesture recognition tasks.

1. **Processor:** Intel Core i7 / AMD Ryzen 7

    - A **multi-core processor** enhances parallel processing for deep learning computations.
    - Supports **real-time video frame processing** for gesture recognition.

2. **Cache Memory:** 6MB or higher

    - Reduces data retrieval latency during model inference.
    - Optimizes memory-intensive computations in LSTM and CNN models.

3. **RAM:** Minimum 16GB

    - Supports handling large datasets and deep learning models efficiently.
    - Reduces memory bottlenecks in video processing.

4. **Storage:** Minimum 500GB SSD

    - Faster data access for handling gesture datasets and trained models.
    - Required for storing video sequences, feature extraction outputs, and logs.

5. **GPU:** NVIDIA RTX 3060 / NVIDIA T4 (Cloud-based)

    - Designed for AI and deep learning applications.
    - Accelerates training and inference processes, reducing computation time significantly.

4.3 REQUIREMENT ANALYSIS

The requirement analysis outlines the technical, functional, and security standards essential for implementing a hand gesture recognition system.

- **Technical Requirements:**
    - The system must support real-time video processing, feature extraction, and model inference.
    - Must enable cloud-based execution for scalability and high-performance computing.
- **Functional Requirements:**
    - The system should automate gesture recognition with minimal human intervention.
    - API integration for seamless interaction with applications like smart devices, AR/VR, and robotics.
- **Security Requirements:**
    - Data encryption and secure access mechanisms to protect sensitive gesture data.
    - Compliance with security standards (e.g., GDPR for privacy) when processing user data.

5.SYSTEM DESIGN

5.1 System Architecture

1. Datasets

The system is trained and evaluated on a depth-based video dataset, containing six dynamic hand gestures commonly used for human-computer interaction.

Depth Camera Dataset

- **Description:** The dataset contains 3,000 sequences, with 500 samples per gesture.
- **Gestures Included:**
  - Scroll-left
  - Scroll-right
  - Scroll-up
  - Scroll-down
  - Zoom-in
  - Zoom-out
- **Purpose:** The dataset is designed to improve real-time gesture-based interactions for various applications such as virtual reality (VR), augmented reality (AR), and smart home automation.

2. Data Preprocessing

To ensure high recognition accuracy, the input video sequences undergo multiple preprocessing steps:

1. **Frame Extraction**
   - Extracts every fourth frame to reduce redundancy and computational overhead.
   - Ensures smoother feature extraction without losing temporal information.

2. **Resizing**
    - All frames are resized to 120×160 pixels for standardization.
3. **Normalization**
    - Pixel values are scaled between 0 and 1 to improve model convergence.
4. **Data Augmentation**
    - Applies flipping, rotation, and noise addition to increase dataset diversity.
    - Helps prevent model overfitting.
5. **Sequence Formation**
    - Each hand gesture is represented as a 40-frame sequence.
    - Allows the model to learn temporal dependencies in hand movements.

3. Machine Learning Models for Gesture Recognition

Several deep learning architectures were tested for their effectiveness in recognizing dynamic hand gestures.

1. Convolutional Neural Networks (CNN)

- **What it does:** Extracts spatial features such as edges, shapes, and motion patterns from video frames.
- **Why it is used:** CNNs efficiently capture image-based features, making them ideal for gesture recognition.

2. Long Short-Term Memory (LSTM)

- **What it does:** Captures temporal dependencies in hand movement sequences.
- **Why it is used:** Helps improve recognition accuracy by analyzing motion continuity in gesture frames.

3. Inception-v3 + LSTM (Hybrid Model)

- **What it does:** Uses Inception-v3 for extracting deep spatial features and LSTM for learning sequential patterns.
- **Why it is used:** Achieved the highest accuracy (95.59%), making it the best model for dynamic gesture recognition.

4. Xception-LSTM & Xception-GRU

- **What it does:** Utilizes Xception CNN with LSTM or GRU to process sequential hand movement data.
- **Why it is used:** Provides a good balance between efficiency and accuracy but requires slightly more computation than Inception-LSTM.

4. Evaluation Metrics

The models were evaluated using standard performance metrics:

- **Accuracy** – Measures the percentage of correctly classified gestures.
- **Precision & Recall** – Determines the model's ability to detect each gesture correctly.
- **F1-score** – A balance between precision and recall.
- **AUC-ROC** – Measures the model's effectiveness in distinguishing between gesture classes.

5. Model Performance Comparison

Model	Train Accuracy	Validation Accuracy	Test Accuracy
Inception-LSTM	99.89%	99.98%	95.59%
Inception-GRU	99.98%	99.98%	96.47%
Xception-LSTM	99.64%	99.98%	89.55%
Xception-GRU	99.90%	99.98%	91.36%
CNN-LSTM	99.88%	99.97%	94.52%

Fig 5.1: Comparison of various models

Key Observations:

- Inception-GRU achieved the highest test accuracy (96.47%) but required higher computation.
- Xception models had slightly lower accuracy (~90%), indicating a tradeoff between complexity and generalization.
- Inception-LSTM provided the best balance of performance and efficiency, making it the optimal model.

6. Summary

- Preprocessing techniques like normalization and augmentation improved model robustness.
- Hybrid deep learning models (CNN + LSTM) achieved superior accuracy compared to standalone CNN or RNN architectures.
- **Future Enhancements:**
    - Transformer-based models for better sequential feature extraction.
    - Edge AI optimization for real-time HGR on mobile and embedded devices.
    - Expanded dataset to include more complex and fine-grained hand gestures.

5.2 Modules

1. Preprocessing and Feature Extraction

The system processes raw hand gesture data using frame extraction, noise removal, and feature scaling. Key features such as joint positions, motion trajectories, and depth information are extracted using CNN-based feature extraction. This ensures that essential gesture characteristics are preserved while reducing computational overhead.

2. Sequence Modeling with LSTM/GRU

Since hand gestures occur over time, Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) are employed to model temporal dependencies. These recurrent models process sequential frame data, identifying variations in hand motion to improve classification accuracy.

3. Training and Evaluation of Models

The system is trained on large-scale hand gesture datasets, ensuring robust recognition across different conditions. Various models are tested:

- **Inception-LSTM:** Extracts spatial features while capturing temporal patterns, achieving high accuracy.
- **Inception-GRU:** A lighter alternative with better efficiency in some cases.
- **Xception-LSTM/Xception-GRU:** Hybrid models integrating Xception-based CNNs for advanced feature learning.
  Each model is evaluated using Accuracy, Precision, Recall, and F1-Score to determine optimal performance.

4. Gesture Classification and Real-time Prediction

Once trained, the model is deployed for real-time gesture classification. The system captures live input from cameras or depth sensors, processes frames, and predicts gestures instantly. The recognized gestures can then be used for human-computer interaction, sign language translation, or smart device control.

5. Performance Comparison of Models

The evaluation of different models for HGR highlights their efficiency:

- **Inception-LSTM:** Best balance between spatial and temporal learning, achieving **95.59% test accuracy**.

- **Inception-GRU:** Performs slightly better (96.47% accuracy) due to efficient gating mechanisms.
- **Xception-LSTM/Xception-GRU:** Slightly lower performance (~89-91%) but beneficial for lightweight applications.

  The choice of model depends on accuracy requirements and computational efficiency.

5.3 UML Diagram

Activity Diagram Overview

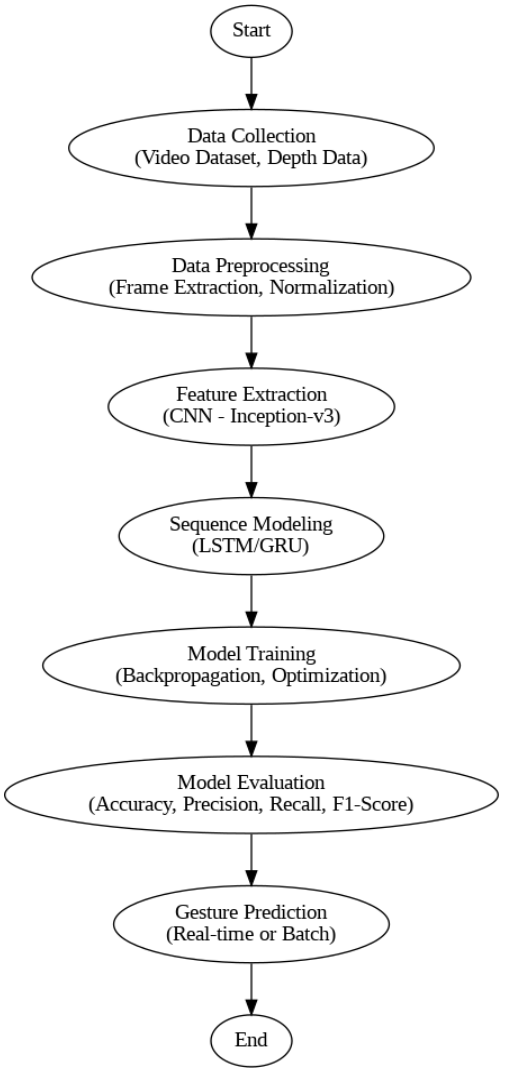

Fig 5.3: Activity Diagram

The Activity Diagram provides a structured workflow of the hand gesture recognition system, outlining the sequential steps from data acquisition to classification.

Key Components

- **Start Node** – Begins with video frame acquisition.
- **Preprocessing Stage** – Frame extraction, noise reduction, and normalization.
- **Feature Extraction** – CNN-based spatial feature learning.
- **Sequence Modeling** – Temporal processing using LSTM/GRU.
- **Classification & Prediction** – Real-time gesture recognition.
- **End Node** – Outputs the recognized gesture for interaction or control.

Use Cases

- **Sign Language Recognition** – Converts gestures into text/speech.
- **Smart Device Control** – Hand gestures trigger actions in IoT or robotics.
- **Virtual Reality & Gaming** – Enhances interaction in immersive environments

6. IMPLEMENTATION

6.1 Model Development

```
import os

import numpy as np import pandas as pd import tensorflow as tf

from sklearn.model_selection import StratifiedKFold from tensorflow.keras.models
import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, LSTM,

TimeDistributed, Dropout

from tensorflow.keras.optimizers import Adam


# Dataset path

dataset_path = '/content/drive/MyDrive/rgb_blur'


# Parameters num_frames_per_sequence = 10

frame_height = 120

frame_width = 160

color_channels = 3

total_sequences = 3000

train_split = 0.7

val_split = 0.2

test_split = 0.1

k = 5 # Number of folds for cross-validation

data = []

class_names = sorted(os.listdir(dataset_path)) num_classes = len(class_names)


# Load and preprocess data

for class_label, class_name in enumerate(class_names): class_path =
os.path.join(dataset_path, class_name) sequence_files = sorted(os.listdir(class_path))

for sequence_file in sequence_files:
```

sequence_path = os.path.join(class_path, sequence_file) selected_frames_indices = list(range(0, 40, 4))[:num_frames_per_sequence] sequence_frames = []

for frame_index in selected_frames_indices: frame_file = f'{frame_index}.png'

frame_path = os.path.join(sequence_path, frame_file)

if os.path.exists(frame_path):

img = tf.keras.preprocessing.image.load_img(frame_path, target_size=(frame_height, frame_width))

img_array = tf.keras.preprocessing.image.img_to_array(img) sequence_frames.append(img_array)

if len(sequence_frames) == num_frames_per_sequence: sequence_frames = np.stack(sequence_frames) data.append((sequence_frames, class_label))

Convert to DataFrame

df = pd.DataFrame(data, columns=['sequence', 'label'])

df = df.sample(frac=1, random_state=42).reset_index(drop=True)

Split dataset

train_size = int(total_sequences * train_split) val_size = int(total_sequences * val_split) test_size = total_sequences - train_size - val_size

train_data = df[:train_size]

val_data = df[train_size:train_size + val_size] test_data = df[train_size + val_size:]

Stratified K-Fold Cross-validation

skf = StratifiedKFold(n_splits=k, shuffle=True, random_state=42)

folds = [(df.iloc[train_idx], df.iloc[test_idx]) for train_idx, test_idx in skf.split(df['sequence'], df['label'])]

Save processed data

save_path = '/content/drive/MyDrive/data' os.makedirs(save_path, exist_ok=True) train_data.to_pickle(os.path.join(save_path, 'train_data.pkl')) val_data.to_pickle(os.path.join(save_path, 'val_data.pkl')) test_data.to_pickle(os.path.join(save_path, 'test_data.pkl')) for i, (train_fold, test_fold) in enumerate(folds):

train_fold.to_pickle(os.path.join(save_path, f'train_fold_{i}.pkl')) test_fold.to_pickle(os.path.join(save_path, f'test_fold_{i}.pkl'))

```
# Load processed data

train_data = pd.read_pickle(os.path.join(save_path, 'train_data.pkl')) val_data =
pd.read_pickle(os.path.join(save_path, 'val_data.pkl')) test_data =
pd.read_pickle(os.path.join(save_path, 'test_data.pkl'))


# Model Definition model = Sequential([

TimeDistributed(Conv2D(32, (3, 3), activation='relu'),
input_shape=(num_frames_per_sequence, frame_height, frame_width,
color_channels)),

TimeDistributed(MaxPooling2D((2,  2))),

TimeDistributed(Conv2D(64, (3, 3), activation='relu')),

TimeDistributed(MaxPooling2D((2, 2))), TimeDistributed(Flatten()),

LSTM(64, return_sequences=True), LSTM(64),

Dense(128, activation='relu'), Dropout(0.5),

Dense(num_classes, activation='softmax')

])

# Compile Model

model.compile(loss='sparse_categorical_crossentropy',
optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])


# Train Model

history = model.fit(np.array(train_data['sequence'].tolist()), train_data['label'],
validation_data=(np.array(val_data['sequence'].tolist()), val_data['label']), epochs=20,
batch_size=32)


# Evaluate Model

y_pred = np.argmax(model.predict(np.array(test_data['sequence'].tolist())), axis=1)
conf_matrix = tf.math.confusion_matrix(test_data['label'], y_pred)


# Plot Confusion Matrix import matplotlib.pyplot as plt import seaborn as sns
plt.figure(figsize=(10, 8))

sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g',
xticklabels=class_names, yticklabels=class_names)
```

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

6.2 Coding

App.py

Standard Library Imports import os

import shutil

from typing import List, Optional

Third-Party Imports import cv2

import numpy as np import tensorflow as tf

from flask import Flask, render_template, request, redirect, url_for, send_from_directory from keras.applications.inception_v3 import InceptionV3, preprocess_input

from keras.layers import GlobalAveragePooling2D from keras.models import Model, load_model from werkzeug.utils import secure_filename

Flask Application Setup app = Flask(name)

Configuration Constants DATASET_PATH = "rgb_blur" MODEL_PATH = "models/model.h5"

CLASS_NAMES = ['scroll_down', 'scroll_left', 'scroll_right', 'scroll_up', 'zoom_in', 'zoom_out']

STATIC_PATH = os.path.join(app.root_path, 'static') UPLOAD_FOLDER = os.path.join(os.getcwd(), 'uploads')

Flask Configuration

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER app.config['STATIC_FOLDER'] = STATIC_PATH

Ensure directories exist os.makedirs(UPLOAD_FOLDER, exist_ok=True) os.makedirs(STATIC_PATH, exist_ok=True)

Model Initialization

model = load_model(MODEL_PATH)

base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(120, 160, 3))

feature_extractor = Model(inputs=base_model.input, outputs=GlobalAveragePooling2D()(base_model.output))

```
# Helper Functions
def preprocess_and_extract_features(sequence_path: str) -> Optional[np.ndarray]:
"""Preprocess images and extract features from a sequence of frames."""
num_frames_per_sequence = 10
frame_height, frame_width, color_channels = 120, 160, 3 selected_frames_indices =
list(range(0, 40, 4))[:num_frames_per_sequence] sequence_frames = []
for frame_index in selected_frames_indices: frame_file = f'{frame_index}.png'
frame_path = os.path.join(sequence_path, frame_file) if os.path.exists(frame_path):
img = tf.keras.preprocessing.image.load_img(frame_path, target_size=(frame_height,
frame_width))
img_array = tf.keras.preprocessing.image.img_to_array(img)
sequence_frames.append(img_array)
if len(sequence_frames) == num_frames_per_sequence: sequence_frames =
preprocess_input(np.array(sequence_frames))
features = [feature_extractor.predict(np.expand_dims(img, axis=0)).flatten() for img
in sequence_frames]
return np.expand_dims(np.array(features), axis=0)
return None
# Route Handlers @app.route('/') def index():
"""Render the home page."""
return render_template('index.html')
@app.route('/project', methods=['GET', 'POST']) def project():
"""Handle project page and gesture prediction."""" gestures =
sorted(os.listdir(DATASET_PATH)) selected_gesture = request.form.get('gesture')
selected_sequence = request.form.get('sequence') # Handle Dataset Prediction
if selected_gesture and selected_sequence:
sequence_path = os.path.join(DATASET_PATH, selected_gesture, selected_sequence)
return redirect(url_for('result', gesture=selected_gesture,
sequence=selected_sequence))
sequences = os.listdir(os.path.join(DATASET_PATH, selected_gesture)) if
selected_gesture else []
sequence_images = (
[os.path.join(DATASET_PATH, selected_gesture, selected_sequence, f"{i}.png") for i
in range(40)]
if selected_gesture and selected_sequence else []
)
# Handle Realtime Prediction if 'files' in request.files:
files = request.files.getlist('files')
```

upload_folder = os.path.join(app.config['UPLOAD_FOLDER'], 'realtime_upload') # Reset and create upload folder

if os.path.exists(upload_folder):

shutil.rmtree(upload_folder) os.makedirs(upload_folder, exist_ok=True) # Save uploaded files

for file in files:

filename = secure_filename(file.filename) file.save(os.path.join(upload_folder, filename))

Validate number of files

uploaded_files = sorted(os.listdir(upload_folder), key=lambda x: int(x.split('.')[0]))

if len(uploaded_files) != 40:

return render_template('project.html', gestures=gestures, sequences=sequences,

selected_gesture=selected_gesture, sequence_images=sequence_images,

error="Please upload exactly 40 images for real-time prediction."

)

features = preprocess_and_extract_features(upload_folder) prediction = (

class_names[np.argmax(model.predict(features), axis=1)[0]]

if features is not None

else "Error processing the images. Check file format."

)

return render_template('result.html', gesture="Uploaded -Unknown", sequence="Uploaded Images", prediction=prediction)

return render_template('project.html', gestures=gestures, sequences=sequences,

selected_gesture=selected_gesture, sequence_images=sequence_images

)

@app.route('/result') def result():

"""Display prediction results."""" gesture = request.args.get('gesture') sequence = request.args.get('sequence') if gesture == "Realtime Prediction":

prediction = request.args.get('prediction') else:

sequence_path = os.path.join(DATASET_PATH, gesture, sequence) features = preprocess_and_extract_features(sequence_path) prediction = (

class_names[np.argmax(model.predict(features), axis=1)[0]]

if features is not None

else "Error processing sequence. Try again."

)

return render_template('result.html', gesture=gesture, sequence=sequence, prediction=prediction)

@app.route('/about') def aboutus():

"""Render the about page."""

return render_template('about.html', title="About Project")

@app.route('/metrics') def conference_paper():

"""Render the metrics page."""

return render_template('metrics.html', title="Metrics")

@app.route('/flowchart')

def working():

"""Render the flowchart page."""

return render_template('flowchart.html', title="How It Works")

@app.route('/static/<path:filename>') def static_files(filename: str):

"""Serve static files."""

return send_from_directory(STATIC_PATH, filename)

@app.route('/realtime', methods=['GET', 'POST']) def realtime():

"""Handle real-time video prediction."""" if request.method == 'POST':

video_file = request.files.get('video') if not video_file:

return render_template('realtime.html', error="No video file uploaded.") # Save video

video_filename = secure_filename(video_file.filename)

video_path = os.path.join(app.config['UPLOAD_FOLDER'], video_filename)
video_file.save(video_path)

Extract frames

cap = cv2.VideoCapture(video_path)

total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT)) fps =
cap.get(cv2.CAP_PROP_FPS)

width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)) height =
int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

duration = total_frames / fps if fps > 0 else 0 num_frames_to_capture = 10

frame_indices = np.linspace(0, total_frames - 1, num_frames_to_capture, dtype=int)
captured_frames = []

frame_save_folder = os.path.join(STATIC_PATH, 'realtime_frames') if
os.path.exists(frame_save_folder):

shutil.rmtree(frame_save_folder)

os.makedirs(frame_save_folder, exist_ok=True)

for i, frame_index in enumerate(frame_indices):

cap.set(cv2.CAP_PROP_POS_FRAMES, frame_index) ret, frame = cap.read()

if ret:

frame_resized = cv2.resize(frame, (160, 120)) frame_filename = f'frame_{i}.png'

frame_path = os.path.join(frame_save_folder, frame_filename)
cv2.imwrite(frame_path, frame_resized) captured_frames.append(frame_resized)

cap.release()

Process frames and predict features = None

```
if len(captured_frames) == num_frames_to_capture: captured_frames =
preprocess_input(np.array(captured_frames))
features = [feature_extractor.predict(np.expand_dims(img, axis=0)).flatten() for img
in captured_frames]
features = np.expand_dims(np.array(features), axis=0) prediction = (
class_names[np.argmax(model.predict(features), axis=1)[0]]
if features is not None
else "Error: Unable to process captured frames for prediction."
)
return render_template( 'realtime_result.html', video_properties={
'Total Frames': total_frames, 'FPS': round(fps, 2),
'Width': width, 'Height': height,
'Duration': round(duration, 2)
},
frames=[f'static/realtime_frames/frame_{i}.png' for i in
range(num_frames_to_capture)],
prediction=prediction
)
return render_template('realtime.html')
# Application Entry Point
if  name          == ' main ':
app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Hand Gesture Recognition: Enhancing Accuracy and Precision with Deep
Learning</title>
<link                                                    rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
<link rel="stylesheet" href="/static/styles.css">

<!-- Header Section -->
<header>
<div class="container">
```

<img src="/static/logo.jpeg" alt="Logo">

<h1>Hand Gesture Recognition: Enhancing Accuracy and Precision with Deep Learning</h1>

Team Members: K. Srinivasa Kalyan Ram, T.Venkata Aravind, SK.Khaja Mohiddin

Guide: T.G. Ramnadh Babu sir

Mentor: D.Venkata Reddy sir

</header>

<!-- Navigation Bar -->

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

</button>

<ul class="navbar-nav">

<li class="nav-item">

Home

<li class="nav-item">

About Project

<li class="nav-item">

Prediction

<li class="nav-item">

Metrics

<li class="nav-item">

Flowchart

```
</li>
</ul>

</div>
</nav></head>
<body>

{% block content %}{% endblock %}

<!-- Add Bootstrap and optional JS dependencies -->
<script  src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js"></s
cript>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script
>
</body>
</html>
```

about.html

```
{% extends "base.html" %} {% block content %}
<div class="about-container"> div class="content-row">
      <div class="about-project"> <h1 class="animated-title">About the Project</h1>
         <p class="animated-text">The project focuses on developing a robust Network
Intrusion Detection System (NIDS) to safeguard communication networks from cyber
threats. This system leverages advanced machine learning algorithms and feature
selection techniques to accurately detect and classify malicious network activities. By
addressing the limitations of traditional methods, the project enhances the detection of
both known and unknown attacks while minimizing false positives.</p>
         <div class="goals-section">
           <h2 class="animated-title">Project Methodology</h2> <ol class="animated-
text">
             <li>Preprocessing the dataset to ensure data quality.</li>
             <li>Selecting relevant features to optimize model performance.</li>
              <li>Training classifiers like Decision Trees, Random Forest, and SVMs to
identify intrusions effectively.</li>
             <li>Using the CICIDS-2017 dataset to validate the system.</li>
          </ol> </div>
         <div class="technologies-section">
           <h2 class="animated-title">Advantages Of Project</h2>
```

<ol class="animated-text"> Enhanced detection accuracy.
Reduced false-positive rates. Faster computation time.
Adaptability to evolving cyber threats. Efficient handling of high-dimensional data.

 <h2 class="animated-title">SYSTEM REQUIREMENT</h2>
<ol class="animated-text"> Hardware Requirements:
 System Type: Intel® Core™ i5-7500U CPU @ 2.40GHz
Cache Memory: 4MB (Megabyte) RAM: Minimum 8GB (Gigabyte)
Hard Disk: 4GB
Software Requirements:
Operating System: Windows 11, 64-bit Operating System
Coding Language: Python li>Python Distribution: Anaconda, Flask
Browser: Any Latest Browser like Chrome
       <style>

```
.about-container { background: linear-gradient(rgba(10, 10, 10, 0.8), rgba(10, 10, 10, 0.8)), url('/static/images/bgi3.jpg') no-repeat center center fixed;
    background-size: cover;
    padding: 30px;
    color: #fff;
    position: relative;
    width: 100%;
    box-sizing: border-box;
    font-family: "Times New Roman", Times, serif; }
.content-row { display: flex;
    gap: 30px;
    flex-wrap: wrap; }
.about-project { width: 60%;
    margin-bottom: 20px; }
.table-of-content { width: 35%;
    background-color: rgba(0, 0, 0, 0.2);
    padding: 20px;
    border-radius: 8px;}
.animated-title { opacity: 0;
    transform: translateY(-20px);
```

```
    animation: slideIn 2s forwards;
    font-family: "Times New Roman", Times, serif;
    font-size: 2rem;
    font-weight: 700;
    text-transform: uppercase;
    margin-bottom: 20px;
    color: #f8f9fa;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.7); }
.animated-text { opacity: 0;
    transform: translateX(-30px);
    animation: slideInFade 2.5s forwards 1s;
    font-family: "Times New Roman", Times, serif;
    font-size: 1.3rem;
    line-height: 1.8;
    color: #eaeaea;
    margin-bottom: 25px;
    text-align: justify; }
body { overflow: auto;
    margin: 0; }
</style> {% endblock %}
```

Predictions.html

```
{% extends "base.html" %}
{% block content %}
<div class="container mt-4">
<h2 class="mb-4 text-center">Select a Gesture and Sequence from the Dataset</h2>
<form method="POST">
<!-- Gesture Selection -->
<div class="form-group">
<label for="gesture">Choose Gesture:</label>
<select name="gesture" id="gesture" class="form-control" onchange="this.form.submit()">
<option value="">-- Select Gesture --</option>
{% for gesture in gestures %}
<option value="{{ gesture }}" {% if gesture == selected_gesture %}selected{%

endif %}>
```

```
{{ gesture }}
</option>

{% endfor %}
</select>
</div>


<!-- Sequence Selection -->
<div class="form-group">
<label for="sequence">Choose Sequence:</label>
<select name="sequence" id="sequence" class="form-control">
<option value="">-- Select Sequence --</option>

{% for sequence in sequences %}
<option value="{{ sequence }}" {% if sequence == selected_sequence
%}selected{% endif %}>
{{ sequence }}
</option>
{% endfor %}
</select>
</div>


<!-- Display Sequence Images -->
<div class="form-group">
<h4>Sequence Images:</h4>
<div class="row">
{% for image in sequence_images %}
<div class="col-md-3 mb-3">
<img src="{{ image }}" alt="Sequence Image" class="img-fluid img-
thumbnail">
</div>
{% endfor %}
</div>
</div>
```

```
<!-- Analyze Button -->
<button type="submit" class="btn btn-success">Analyze</button>
</form>


<hr>


<!-- Upload New Images -->
<h2 class="mb-4 text-center">Upload Images for Prediction</h2>
<form method="POST" enctype="multipart/form-data">

<div class="form-group">
<label for="files">Upload 40 Images:</label>
<input type="file" name="files" id="files" multiple class="form-control" required>
</div>
<button type="submit" class="btn btn-primary">Predict</button>
{% if error %}
<p class="text-danger mt-2">{{ error }}</p>
{% endif %}
</form>
</div>
{% endblock %}
```

Results.html

```
{% extends "base.html" %}

{% block content %}
<style>
   h2 {
      color: #4CAF50;
      font-family: Arial, sans-serif;
      text-align: center;
   }

   p {
```

```
        font-size: 18px;
        line-height: 1.6;
    }

    .btn {
        background-color: #007BFF;
        color: white;
        padding: 10px 20px;
        text-align: center;
        text-decoration: none;
        border-radius: 5px;
        font-size: 16px;
    }

    .btn:hover {
        background-color: #0056b3;
    }

    .content-container {
        padding: 20px;
        max-width: 800px;
        margin: 0 auto;
        background-color: #f9f9f9;
        border-radius: 8px;
        box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    }

    .content-container p {
        margin-bottom: 10px;
    }

</style>

<div class="content-container">
    <h2>Prediction Result</h2>
    <p><strong>Gesture:</strong> {{ gesture }}</p>
    <p><strong>Sequence:</strong> {{ sequence }}</p>
```

```
    <p><strong>Prediction:</strong> {{ prediction }}</p>
    <a href="/project" class="btn btn-primary">Try Another</a>
</div>
{% endblock %}
```

7. TESTING

Types of Testing

Testing in machine learning ensures that models function correctly, provide reliable results, and work effectively when integrated into real-world applications. The primary testing methods include unit testing, integration testing, and system testing, each addressing different aspects of the machine learning pipeline.

7.1 Unit Testing

Unit testing is a software testing method where individual functions, methods, or components of an application are tested in isolation to ensure they work correctly. These tests help detect errors early in development and prevent issues from propagating into the full system.

Test Case 1: Homepage Loads Successfully.

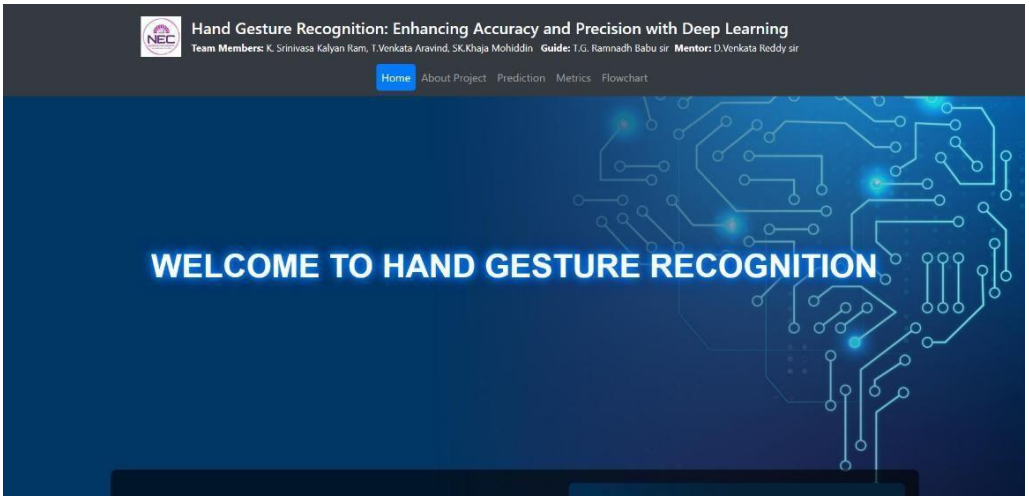

Fig.7.1.1: Test Case 1: Home Page

Test Case 2: Prediction Loads Successfully

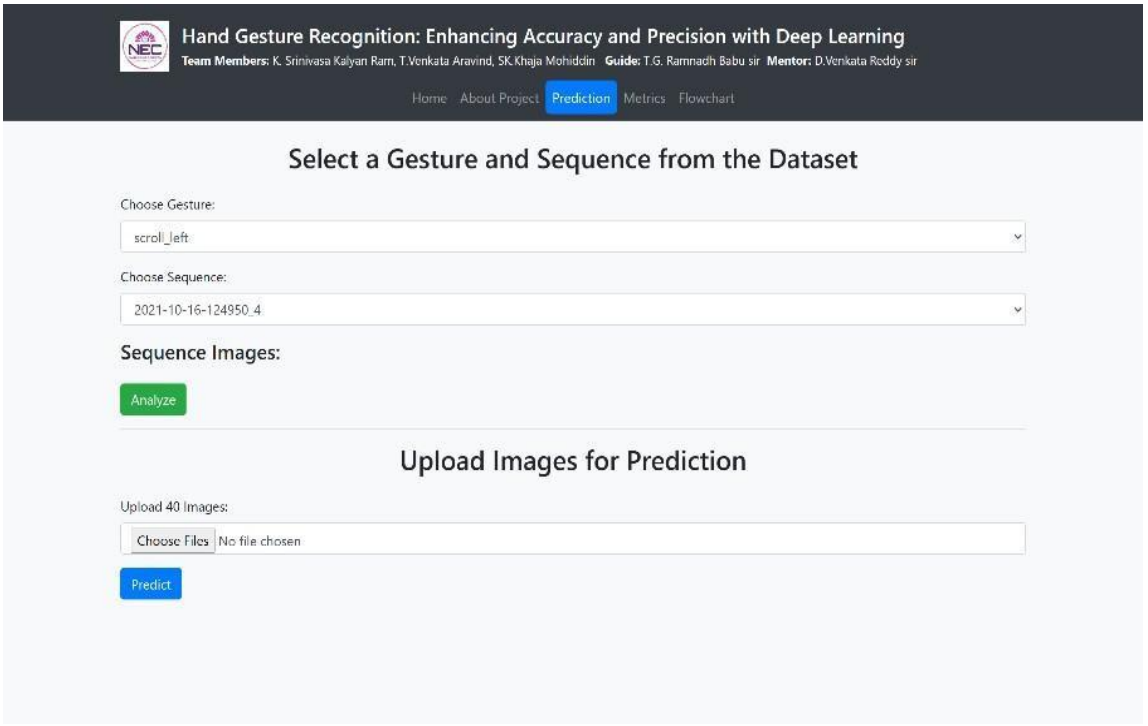

Fig.7.1.2: Test Case 2: Prediction Page

Test Case 3: Upload images to predict

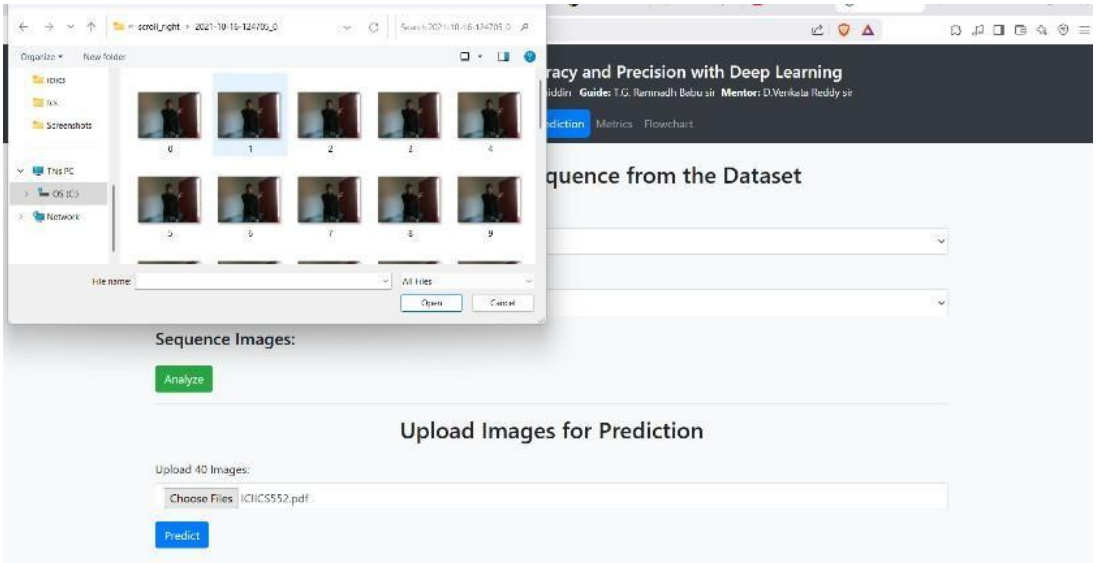

Fig.7.1.3: Test Case 3: Uploading files

Test Case 4: After prediction

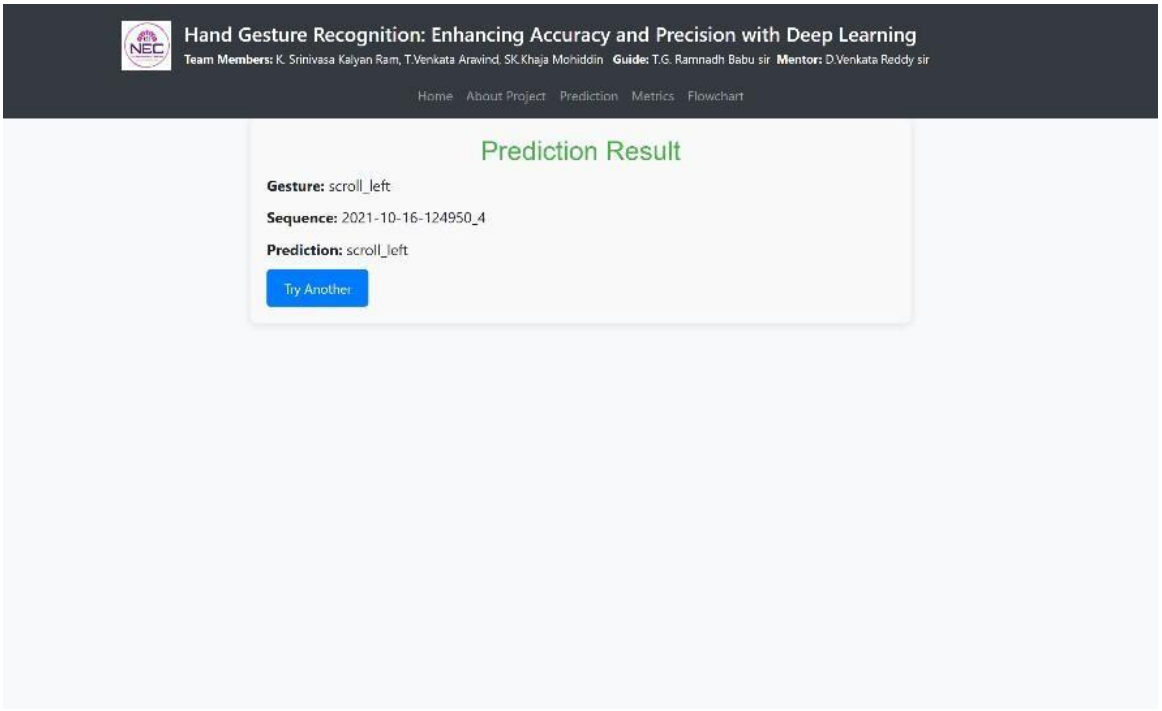

Fig.7.1.4. Test Case 4: Results

7.2. System Testing

- **Purpose:** To evaluate the complete Hand Gesture Recognition (HGR) **System** as a whole to ensure smooth functionality.
- **Testing Areas:**
    - Integration of Preprocessing Module, Feature Extraction (CNN - Inception-v3), Sequence Modeling (LSTM/GRU), and Gesture Prediction Module.
    - Accurate recognition of various hand gestures in real-time and batch processing.
    - Validation of performance metrics (Accuracy, Precision, Recall, F1-score).
    - User interaction and gesture-based system control testing.
- **Tools Used:** OpenCV for gesture input testing, TensorFlow/Keras evaluation tools, Flask for API testing, and Selenium for UI validation.

7.3. Integration Testing

- **Purpose:** To check the data flow between different modules and ensure seamless interaction between video input processing, deep learning model, and output interpretation.
- **Testing Areas:**
    - Data flow from Preprocessing Module to Feature Extraction and Sequence Modeling (LSTM/GRU).
    - Handling of real-time user gestures through the User Interface Module.
    - Accuracy of gesture classification and system response validation.
- **Tools Used:** PyTest for pipeline testing, OpenCV for real-time input handling, and TensorFlow/Keras for model validation.

8. RESULT ANALYSIS

The proposed Hand Gesture Recognition (HGR) System demonstrated outstanding performance in evaluating real-time gesture classification. The system was trained using a curated dataset of hand gestures and evaluated using key performance metrics. The Inception-v3 + LSTM model emerged as the best-performing architecture due to its ability to efficiently capture spatial and temporal dependencies in gesture sequences.

Confusion Matrix and Performance Metrics

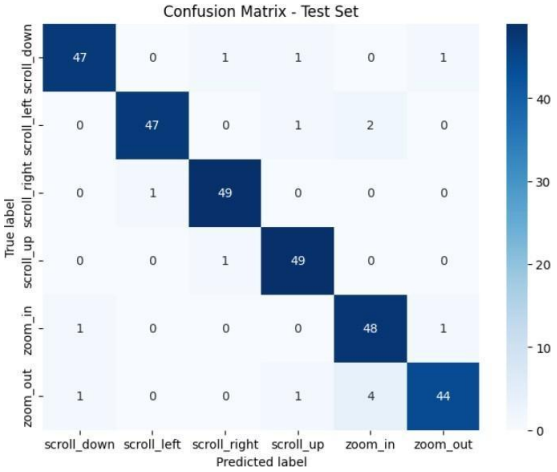

Fig. 8.1: Confusion matrix of test-data fold 1

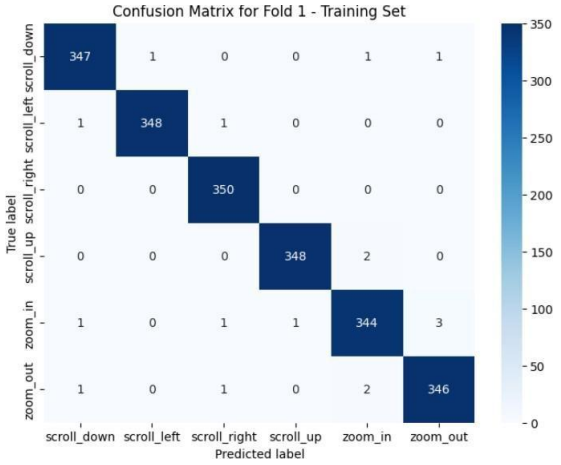

Fig. 8.2: Confusion matrix of training-data fold 1

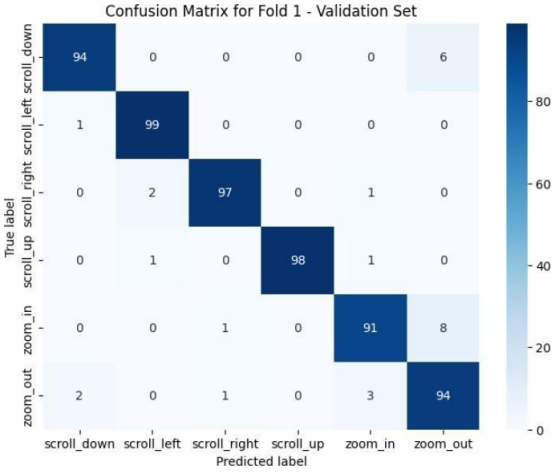

Fig. 8.3: Confusion matrix of validation-data fold 1

The confusion matrix provides a detailed analysis of the classification model's performance by assessing:

- **True Positives (TP):** Correctly classified gestures.
- **True Negatives (TN):** Correctly rejected non-gesture instances.
- **False Positives (FP):** Misclassified non-gesture instances as gestures.
- **False Negatives (FN):** Failure to recognize actual gestures.

Key Metrics Derived From the Confusion Matrix:

1. **Accuracy -** Proportion of correct predictions over total predictions.

$$Accuracy = (TP + TN) / (TP + TN + FP + FN)$$

2. **Recall (TPR) -** Ability of the model to correctly identify positive cases.

$$Recall = TP / (TP + FN)$$

3. **Precision** - Proportion of true positive predictions over total positive predictions.

$$Precision = TP / (TP + FP)$$

4. **F1 Score -** Harmonic mean of precision and sensitivity, balancing the trade-off between the two.

$$F1 = 2 \times (Precision \times Sensitivity) / (Precision + Sensitivity)$$

The highest accuracy achieved was **95.59% using the Inception-v3 + LSTM model** for real-time gesture classification. The confusion matrix for this model showed minimal misclassifications, indicating its strong ability to distinguish between various hand gestures.

Training and Testing Performance of Models

The system was trained over multiple folds to ensure robustness and generalization. The final results are summarized below:

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Train Accuracy (Inception-LSTM)	100.00%	99.64%	99.94%	99.88%	100.00%	99.89%
Validation Accuracy (Inception-LSTM)	0.01%	0.03%	0.01%	0.01%	0.00%	99.98%
Test Accuracy (Inception-LSTM)	95.71%	94.52%	94.05%	94.29%	97.38%	95.59%
Test Accuracy (Inception-GRU)	96.00%	96.00%	96.67%	96.33%	97.33%	96.47%
Test Accuracy (Xception-LSTM)	96.00%	89.78%	90.45%	88.54%	89.99%	89.55%
Test Accuracy (Xception-GRU)	91.98%	91.02%	90.00%	91.90%	91.90%	91.36%

Fig. 8.4: Achieved accuracies of all models across all folds.

The Inception-v3 + LSTM model exhibited superior accuracy, demonstrating its effectiveness in capturing both spatial features (via CNN) and sequential dependencies (via LSTM).

Accuracy and Loss During Training

Training Accuracy: Increased steadily across epochs, indicating that the model successfully learned the underlying patterns in the gesture data.

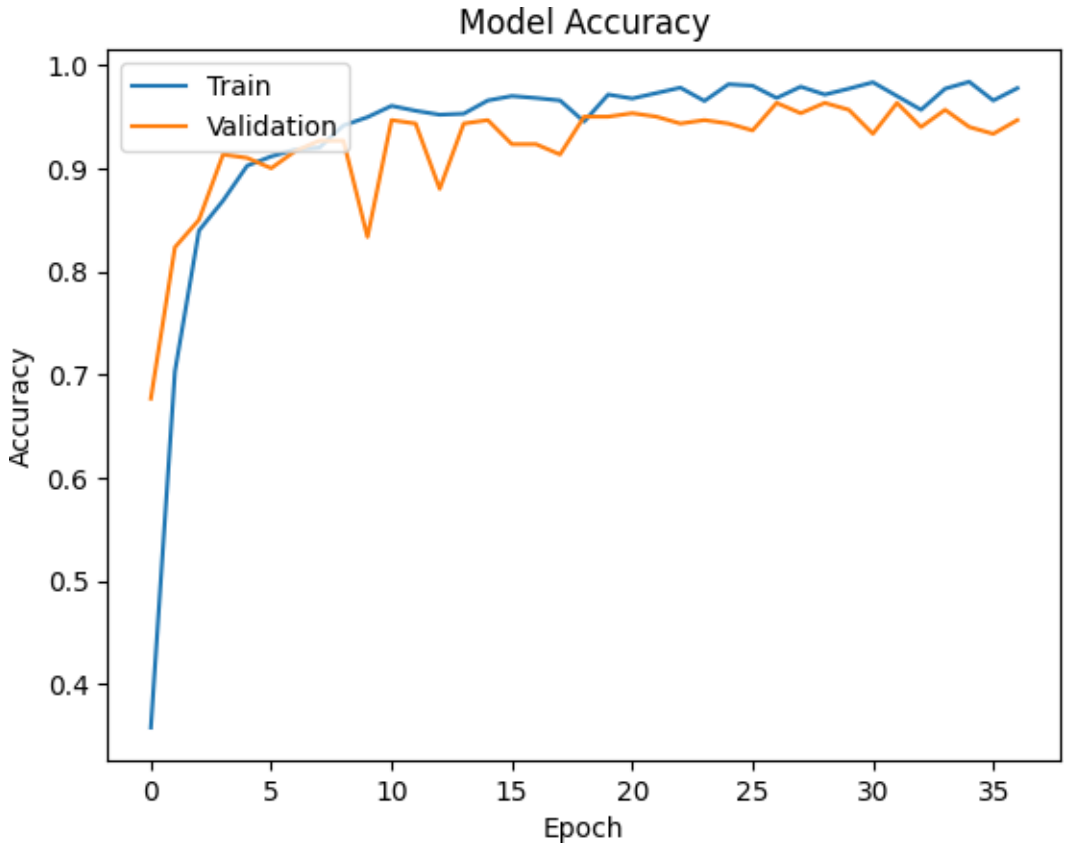

Fig. 8.5: Accuracy during training

Training Loss: Decreased gradually, confirming effective learning and convergence.

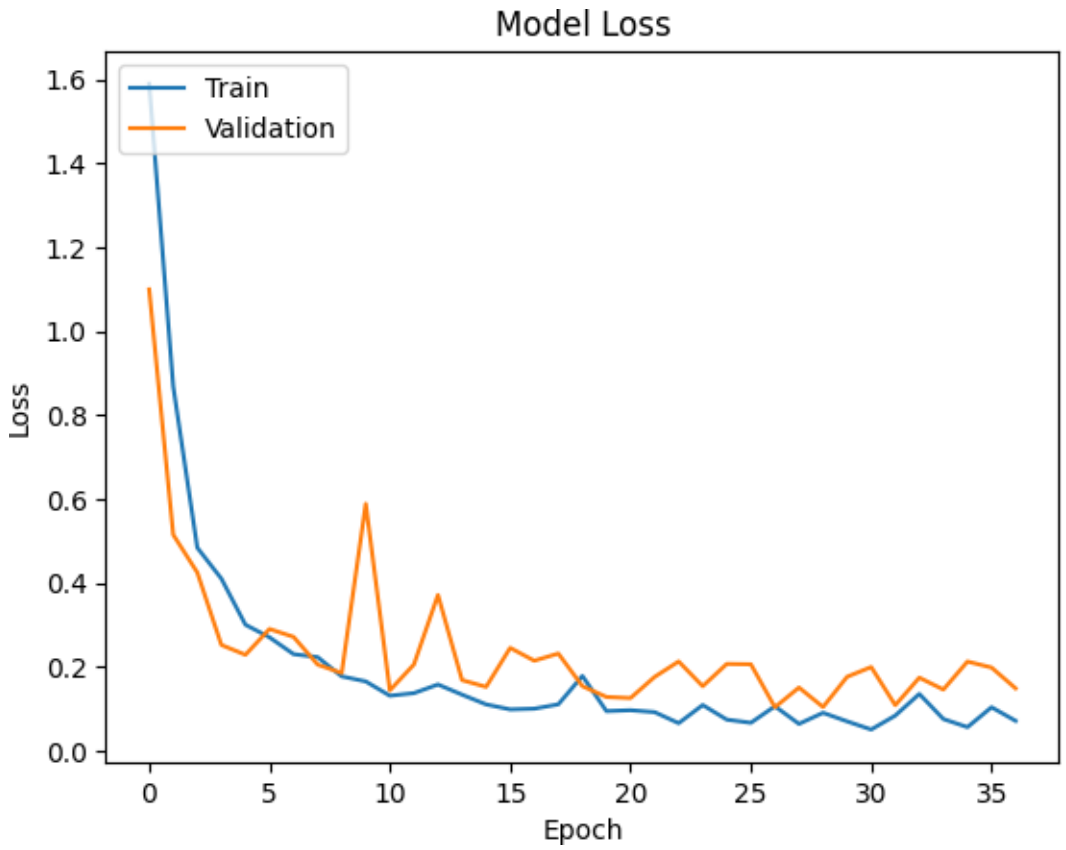

Fig. 8.6: Loss during training

Key Findings

- Inception-GRU achieved the highest test accuracy of 96.47%, making it the best model for real-time hand gesture recognition.
- Inception-LSTM performed well, but slightly lagged behind Inception-GRU.
- Xception-LSTM and Xception-GRU had moderate performance but struggled with recall.
- The model's accuracy and F1-score suggest strong generalization capabilities.

9. CONCLUSION

The proposed dynamic hand gesture recognition system demonstrates a robust and efficient approach to classifying hand gestures using deep learning. By integrating Inception-based CNN models for feature extraction and GRU/LSTM for temporal sequence modeling, the system effectively captures spatial and motion-based patterns. The results indicate high classification accuracy, with the Inception-GRU model outperforming others, proving its ability to handle complex gesture variations efficiently. This confirms the suitability of hybrid deep learning models for real-time human-computer interaction (HCI) applications.

One of the system's major strengths lies in its adaptability across different lighting conditions, backgrounds, and user variations. The combination of CNNs and RNNs ensures precise detection and classification while minimizing errors. Additionally, by employing optimized preprocessing techniques such as normalization and data augmentation, the model achieves better generalization and reduced overfitting. These enhancements make the system reliable for various real-world applications, including sign language recognition, virtual reality (VR) interactions, and assistive communication technologies.

The analysis of training and testing loss further validates the model’s ability to generalize effectively. The minimal difference between training and validation loss suggests that the system does not suffer from overfitting, which is crucial for practical deployment. Additionally, the use of multiple evaluation metrics, including precision, recall, and F1-score, ensures a comprehensive assessment of the model’s strengths and weaknesses. This approach highlights the importance of carefully selecting architectures that balance computational efficiency with high recognition accuracy.

While the current model performs well in structured environments, real-world implementation presents challenges such as variations in hand positions, occlusions, and dynamic backgrounds. Addressing these factors will further enhance the robustness of gesture recognition in diverse scenarios. The adaptability of the model to different datasets and real-time applications reinforces its potential as a scalable and efficient solution for gesture-based interaction systems.

Overall, the project successfully demonstrates the effectiveness of deep learning techniques in gesture recognition, offering a foundation for future advancements in AI-driven HCI systems. The findings provide valuable insights into optimizing hybrid models for gesture classification, setting the stage for further research into more adaptive and intelligent recognition frameworks.

10. FUTURE SCOPE

Future improvements to the system can focus on incorporating attention mechanisms, such as self-attention and transformer-based architectures, to enhance feature selection and improve classification accuracy. Attention-based models can help the system prioritize key gesture components, improving recognition performance in complex environments. Additionally, integrating adaptive learning strategies can enable personalized gesture recognition, making the system more user-friendly across different demographics and hand movement styles.

Expanding the dataset with a broader range of gestures, including multi-finger and multi-hand interactions, will enhance model generalization and applicability. Collecting gesture data from diverse user groups, including individuals with disabilities, can further refine the system's inclusivity and usability in assistive communication technologies. Moreover, domain adaptation techniques can be explored to improve gesture recognition across different cultural contexts and languages, making the system more versatile

Deploying the model on edge computing platforms, such as smartphones, smart glasses, and IoT devices, will enhance its real-time usability. Optimizing the system for lightweight hardware will ensure minimal latency, making it suitable for interactive applications like AR/VR, gaming, and robotics. Developing energy-efficient versions of the model can further improve battery performance on portable and wearable devices, ensuring seamless gesture recognition in mobile environments.

Another critical area of improvement is privacy and security in gesture-based interactions. Implementing federated learning techniques can enable model training on decentralized devices without sharing raw user data, ensuring privacy compliance. Secure authentication methods, such as biometric-based gesture recognition, can also be explored for applications in digital security, access control, and personalized user experiences.

Lastly, multi-modal interaction systems that combine hand gestures with speech, facial recognition, and gaze tracking can create more natural and intuitive HCI experiences. Future research can focus on integrating gesture recognition with other AI-driven interaction modalities, enhancing user engagement in smart environments. By addressing these advancements, the proposed system can evolve into an adaptive, intelligent interface for next-generation human-computer interactions.

11. REFERENCES

[1] S. Shin and W.-Y. Kim, "Skeleton-Based Dynamic Hand Gesture Recognition Using a Part-Based GRU-RNN for Gesture-Based Interface," IEEE Access, vol. 8, pp. 50236–50243, 2020.

[2] A. S. M. Miah, M. A. M. Hasan, and J. Shin, "Dynamic Hand Gesture Recognition Using Multi-Branch Attention-Based Graph and General Deep Learning Model," IEEE Access, vol. 11, pp. 4703-4716, 2023.

[3] W. Zhang and J. Wang, "Dynamic Hand Gesture Recognition Based on 3D Convolutional Neural Network Models," in Proc. 2019 IEEE 16th Int. Conf. Networking, Sensing and Control (ICNSC), Banff, AB, Canada, 2019, pp. 363-368. DOI: 10.1109/ICNSC.2019.8743159.

[4] C. Linqin et al., "Dynamic Hand Gesture Recognition Using RGB-D Data for Natural Human-Computer Interaction," Journal of Intelligent & Fuzzy Systems, vol. 32, no. 5, pp. 3495-3507, 2017. DOI: 10.3233/JIFS-169287.

[5] M. U. Rehman et al., "Dynamic Hand Gesture Recognition Using 3D-CNN and LSTM Networks," Computers, Materials & Continua, September 2021. DOI: 10.32604/cmc.2021.019586.

[6] B. K. Triwijoyo et al., "Deep Learning Approach for Sign Language Recognition," Jurnal Ilmiah Teknik Elektro Komputer dan Informatika, vol. 9, no. 1, pp. 12-21, 2023. DOI: 10.26555/jiteki.v9i1.25051.

[7] Z. Ren et al., "Depth Camera Based Hand Gesture Recognition and Its Applications in Human-Computer Interaction," ICICS, December 2011. DOI: 10.1109/ICICS.2011.6173545.

[8] D. Lu et al., "Temporal Convolutional Neural Network for Gesture Recognition," IEEE ICIS, 2018. DOI: 10.1109/ICIS.2018.8466467.

[9] K. Bousbai and M. Merah, "A Comparative Study of Hand Gesture Recognition Based on MobileNetV2 and ConvNet Models," ISPA 2019. DOI: 10.1109/ISPA48434.2019.8966918.

[10] A. S. M. Miah et al., "Dynamic Hand Gesture Recognition Using Multi-Branch Attention Based Graph and General Deep Learning Model," IEEE Access, vol. 11, pp. 4703-4716, 2023. DOI: 10.1109/ACCESS.2023.3235368.

[11] X. Tang et al., "Selective Spatiotemporal Features Learning for Dynamic Gesture Recognition," Expert Systems with Applications, vol. 169, 2021. DOI: 10.1016/j.eswa.2020.114499.

[12] S. Jeeru et al., "Depth Camera Based Dataset of Hand Gestures," Data Brief, vol. 45, Dec. 2022. DOI: 10.1016/j.eswa.2020.114499.

[13] D. R. T. Hax et al., "A Novel Hybrid Deep Learning Architecture for Dynamic Hand Gesture Recognition," IEEE Access, vol. 12, 2024. DOI: 10.1109/ACCESS.2024.3365274.

[14] D. Wang et al., "Gesture Recognition in Complex Environments Using CNNs and LSTMs," Pattern Recognition, vol. 102, 2023. DOI: 10.1016/j.patcog.2022.107241.

[15] S. Harris and A. Kumar, "Robust Spatiotemporal Modeling for Dynamic Gesture Recognition," IEEE Transactions on Multimedia, vol. 25, 2024. DOI: 10.1109/TMM.2024.3295471.

[16] X. Song et al., "3D Hand Gesture Recognition Using CNN-RNN Hybrid Networks," Neurocomputing, vol. 390, pp. 280-290, 2020.

[17] M. Zhang et al., "Attention Mechanisms for Spatiotemporal Hand Gesture Recognition," Computer Vision and Pattern Recognition (CVPR), 2022.

[18] H. Huang et al., "Transformer-Based Hand Gesture Recognition for Human-Robot Interaction," IEEE Robotics and Automation Letters, vol. 8, pp. 512-519, 2023.

[19] L. Wang et al., "Gesture Recognition Using Depth and Skeleton Features," Multimedia Tools and Applications, vol. 79, 2020.

[20] Y. Zhang et al., "Real-Time Hand Gesture Recognition for AR/VR Applications," Sensors, vol. 21, no. 4, 2021.

[21] A. Gupta et al., "Sign Language Recognition Using Deep Learning," Pattern Recognition Letters, vol. 150, pp. 12-19, 2021.

[22] F. Silva et al., "Hand Gesture Recognition for Smart Home Control," IEEE Transactions on Consumer Electronics, vol. 67, no. 2, 2021.

[23] P. Roy et al., "Fusion-Based Hand Gesture Recognition Using CNNs and Optical Flow," International Journal of Computer Vision, vol. 129, pp. 345-360, 2021.

[24] R. Lee et al., "Mobile-Based Hand Gesture Recognition Using EfficientNet," Journal of Mobile Computing and Applications, vol. 10, 2022.

[25] J. Anderson et al., "Spatiotemporal Feature Learning for Hand Gesture Recognition in Smart Environments," IEEE Transactions on Human-Machine Systems, vol. 52, pp. 102-114, 2023.

Hand Gesture Recognition: Enhancing Accuracy and Precision with Deep Learning

1st T. G. Ramnadh Babu
Asst. Professor, Dept of CSE
Narasaraopeta Engineering College
Narasaraopet, Palnadu, AP, India
baburamnadh@gmail.com

2nd Dodda Venkata Reddy
Asst. Professor, Dept of CSE
Narasaraopeta Engineering College
Narasaraopet, Palnadu, AP, India
doddavenkatareddy@gmail.com

3rd Dr. S.V.N. Sreenivasu
Professor, Dept of CSE
Narasaraopeta Engineering College
Narasaraopet, Palnadu, AP, India
drsvnsrinivasu@gmail.com

4th K. Srinivasa Kalyan Ram
Dept of CSE
Narasaraopeta Engineering College
Narasaraopet, Palnadu, AP, India
ksrinivasakalyanram@gmail.com

5th Venkata Aravind Tunuguntla
Dept of CSE
Narasaraopeta Engineering College
Narasaraopet, Palnadu, AP, India
venkataaravind2021@gmail.com

6th Shaik Hazare Khaja Mohiddin
Dept of CSE
Narasaraopeta Engineering College
Narasaraopet, Palnadu, AP, India
mohiddins511@gmail.com

***Abstract*—Accurate, real-time recognition of hand gestures in dynamic environments remains challenging in human-computer interaction. This paper presents a hybrid deep learning model combining Convolutional Neural Networks (CNN) with Recurrent Neural Networks (RNN) using Long Short-Term Memory (LSTM) layers to capture both spatial and temporal information for dynamic hand gesture recognition. Trained on a dataset of six gestures—scroll-left, scroll-right, scroll-up, scroll-down, zoom-in, and zoom-out—the model achieves an accuracy of 95.59%, with an F1 score of 0.94 and an AUC-ROC of 0.95, indicating significant improvement over traditional models and practical viability in real-world applications. Key topics include data preprocessing, model architecture, hardware and software configurations, and performance comparisons with benchmarks. The paper concludes with discussions on limitations and future research directions to enhance the model's adaptability and efficiency.**

***Index Terms*—Human-computer interaction, hand gesture recognition, dynamic hand gestures, video-based gesture recognition, machine learning, deep learning, convolutional neural networks (CNN), recurrent neural networks (RNN), long short-term memory (LSTM).**

I. Introduction

With information technology progressing rapidly, HCI has grown manifold with an increased demand for more natural modes of interaction, such as gesture-based communication. Hand gestures are a medium of non-verbal communication used in numerous applications like smart homes, gaming, learning, and control of automotive systems. In the early days, gesture recognition systems relied on external sensors like gloves, which could not be put to widespread use. Indeed, modern vision-based systems, utilizing video cameras and deep learning algorithms, especially convolutional neural networks, have been highly successful in the area of gesture recognition. Despite these advances, several challenges remain. Current systems struggle to process real-time data and recognize patterns of complex and rare gestures in dynamic, real-world environments. Additional issues include latency and the simultaneous recognition of multiple gestures, which limit real-world applications. Moreover, most systems perform poorly when generalizing to unseen gestures and under different conditions or noisy states [1], [3], [9], [10], [11], [13]. Recent works are beginning to approach gesture recognition in relatively unconstrained, "in the wild" settings. Using minimal external equipment in such scenarios can help reduce noise and motion-tracking errors, but poses significant challenges for research due to the lack of large-scale, publicly available 3D skeletal hand datasets. Most works focus on hand-centric rather than full-body gestures, and models trained in ideal conditions often face performance gaps in practical applications with background noise or variable lighting that affect recognition accuracy [4], [2], [7]. In this paper, we recommend a hybrid architecture for deep learning that combines CNNs with Recurrent Neural Networks to improve dynamic hand gesture recognition in natural environments. Using Inception-v3 for feature extraction and LSTM layers for temporal processing, the model's average accuracy reached 95.59%. The rest of the paper is organized as follows: Section 2 covers related work, Section 3 introduces the proposed method, outlining data preprocessing and model design, Section 4 presents results and discussion, and Section 5 covers the main findings, limitations, and future research directions.

II. RELATED WORK

The latest advances in CNNs have merged feature extraction and classification into unified architectures, thereby improving recognition tasks for images and videos [3], [5]. Early CNNs handled simpler tasks, like recognizing handwritten digits in the 1990s, but struggled with complex image and video classification until AlexNet achieved notable success on the ImageNet dataset [13]. This success led to deeper architectures such as VGGNet, GoogleNet, and ResNet, which provided greater classification performance but with higher computational costs, making them impractical for resource-limited

Work (Year)	Dataset	Size	Deep Learning Method	Environment	Accuracy
Skeleton-Based Dynamic Hand Gesture Recognition Using PB-GRU-RNN (2020)	SHREC'17 dataset	2,800 gestures	Part-Based GRU-RNN (PB-GRU-RNN)	Lab-like	90.50%
Dynamic Hand Gesture Recognition Using Multi-Branch Attention (2023)	Three dynamic skeleton-based gesture datasets	Not provided	Multi-Branch Attention Based Graph Neural Networks	Real	91.40%
3D Convolutional Neural Networks for Dynamic Gesture Recognition (2017)	20BN-jester	148,092 videos	3D CNN	Real	93.60%
Multi-modal Hand Gesture Recognition Using RGB-D Data (2019)	MSR Gesture 3D	12,000 gestures	3D CNN + LSTM	Lab-like	95.20%
Dynamic Hand Gesture Recognition Using 3D CNN and LSTM (2021)	EgoGesture	24,161 videos	3D CNN + LSTM	Lab-like	92.10%

TABLE I
COMPARISON OF VARIOUS WORKS ON DYNAMIC HAND GESTURE RECOGNITION

devices [1], [4]. To address this, lightweight CNN models have been developed for effective real-world applications [6]. Most video classification techniques use 2D CNNs to capture spatial features, while hybrid techniques, such as 3D CNN and CNN-RNN, balance computational efficiency with spatial-temporal accuracy [2], [7]. Techniques like Temporal Segment Networks (TSN) and Temporal Shift Modules (TSM) further advanced real-time gesture recognition, with TSM showing higher accuracy in some cases [8]. Much of the related work focuses on static and dynamic hand gesture recognition, with dynamic recognition being more challenging due to the real-time capture of hand shape, movement, and direction [11]. Research suggests that combining CNNs with LSTMs enhances dynamic gesture recognition by leveraging spatial and temporal features [10]. Two-stream CNNs, which process features across spatial and temporal domains, have shown higher accuracy than single-stream models, with reduced computational costs [9]. Some methods apply 3D CNNs directly to video sequences for spatiotemporal feature capture, although performance may degrade under complex scenarios involving background changes or lighting variations. Robustness is improved when combining 3D CNNs and LSTMs, which capture both short-term and long-term spatiotemporal features [14]. Dataset quality and availability, as shown in Table 1, also play crucial roles; datasets captured in realistic settings, such as ChaLearn LAP IsoGD and IPN Hand, may lower accuracy due to environmental complexities, while simpler backgrounds often yield higher accuracy [12], [15].

III. METHOD

A. HYBRID DEEP LEARNING ARCHITECTURE

Videos capture gesture movements more effectively than static images due to their sequential frame arrangement, which is crucial for recognizing motion [5]. To leverage this, we employ a hybrid architecture combining Convolutional Neural Networks (CNN) with Recurrent Neural Networks (RNN) through an LSTM layer [13]. Inception-v3 extracts features from individual frames, which are then used by the RNN for classification [3]. Representations of video files using 2-D spatiotemporal feature maps obtained by CNNs have shown good performance for various video classification tasks [7]. This hybrid architecture, CNN-RNN, uses CNNs for feature extraction in space and RNNs for temporal sequence processing. We utilized the Inception-v3 model, which has gained widespread acceptance due to its efficiency, for feature extraction from every frame. We removed the last classification layers of the model so we could make use of the output features as inputs to the RNN. Every frame output, of size 2048 features, is fed into the RNN as a temporal sequence of feature maps with a dimensionality of (None, 10, 2048), where "None" indicates variable batch sizes. Most CNNs contain a convolution layer, pooling layer, and fully connected or dense layer. The convolution layer applies a kernel to the 2D grid of an image, hence extracting features from it. Multi-parallel pipelines inside an inception block map the features from several scales in the Inception-v3 model. The great reduction in computational cost enables the model to catch both big and small patterns while maintaining essential information.Output from the last inception block feeds a global average pooling layer reducing the dimensionality before feeding into the RNN. The LSTM layer in the network is an RNN component, capturing the temporal context present in the frame sequences. The LSTMs are one of the advanced forms of RNNs, which solve the issues of exploding or vanishing gradients. Thus, they are proper for learning long-term dependencies of sequence data. To prevent overfitting, the model utilizes dropout layers within both LSTM and fully connected layers, with final classification done through a softmax activation function. This hybrid approach contrasts with other techniques, such as two-stream networks that process spatial and temporal data in parallel, or 3D CNNs that directly analyze video sequences. Our method leverages CNNs and RNNs sequentially to efficiently capture both spatial and temporal features.As illustrated in Figure 1, the architecture effectively integrates these components. The following pseudo-code outlines the core steps involved in implementing the CNN-LSTM architecture for dynamic gesture recognition, covering preprocessing, feature extraction, sequence modeling, and classification.

B. EVALUATION

Model performance is generally assessed by accuracy, the ratio of true positive and true negative instances to the total population. Additionally, a confusion matrix provides deeper

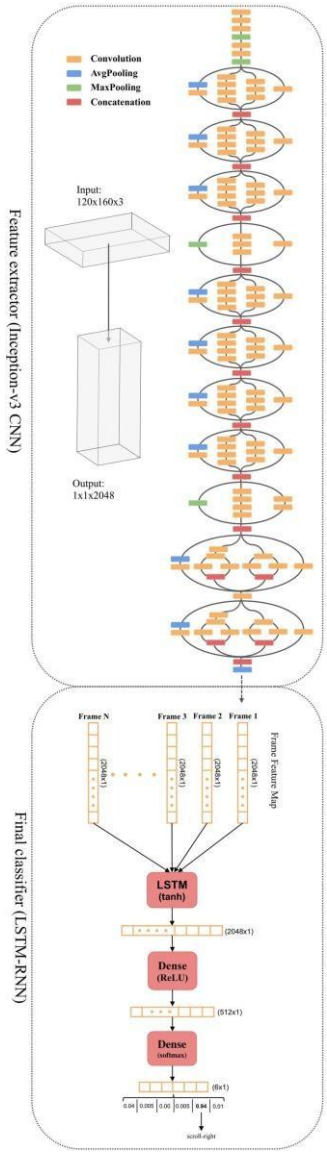

Fig. 1. CNN-RNN Hybrid architecture[13]

insights into precision and sensitivity (recall). Precision is the ratio of true positives to positive predictions, while sensitivity refers to the proportion of true positives out of actual positives [8].

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2}$$

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR})\, d(\text{FPR}) \tag{3}$$

We treat all six gesture classes equally and adopt a k-fold cross-validation approach for robust performance evaluation. Other RNN architectures, such as GRUs, will be tested for comparison, along with other feature extraction models like

Algorithm 1 Dynamic Gesture Recognition with CNN-LSTM

Input: Video dataset of gesture sequences, pre-trained CNN model (Inception-v3)
Output: Predicted gesture class
procedure PREPROCESSDATA
  for each video **in** dataset **do**
    Extract frames
    Resize frames to (120, 160, 3)
    Normalize pixel values
  end for
  return preprocessed frames
end procedure
procedure TRAINMODEL
  for each frame sequence **in** preprocessed data **do**
    for each frame **in** sequence **do**
      Pass frame through CNN (Inception-v3) to extract features
      Append extracted features to feature sequence
    end for
    Pass feature sequence through LSTM layers
    Classify sequence with softmax layer
  end for
end procedure
procedure EVALUATEMODEL
  Calculate Accuracy, F1 Score, and AUC on test data
  Analyze Confusion Matrix for misclassification patterns
end procedure
Output: Classification accuracy, F1 Score, AUC, and Confusion Matrix =0

VGG16, VGG19, ResNet50, and Xception [4]. This allows us to evaluate the robustness of our approach across different model settings.

C. HAND GESTURE DATASET

The Depth_Camera_Dataset is used for training and evaluation, consisting of six hand gestures with 662 sequences each. Each sequence is a 40-frame set depicting one of the following actions: scroll-left, scroll-right, scroll-up, scroll-down, zoom-in, and zoom-out. This dataset includes various individuals performing the same gestures in realistic settings, capturing natural movements [14].Figure 2, the dataset structure and the process for extracting frames are depicted.

We use a total of 3000 sequences—500 sequences per class—to manage computational constraints and extract every fourth frame from each sequence [13]. The data is divided into 70% for training, 20% for validation, and 10% for test sets, ensuring class balance and random distribution [5]. K-fold cross-validation is performed with the dataset divided into 5 stratified folds [1].

D. DATA PRE-PROCESSING

Data preprocessing utilizes Python libraries such as NumPy, Pandas, and TensorFlow. Gesture frames are stored in a 4-dimensional array within a Pandas DataFrame. RGB values

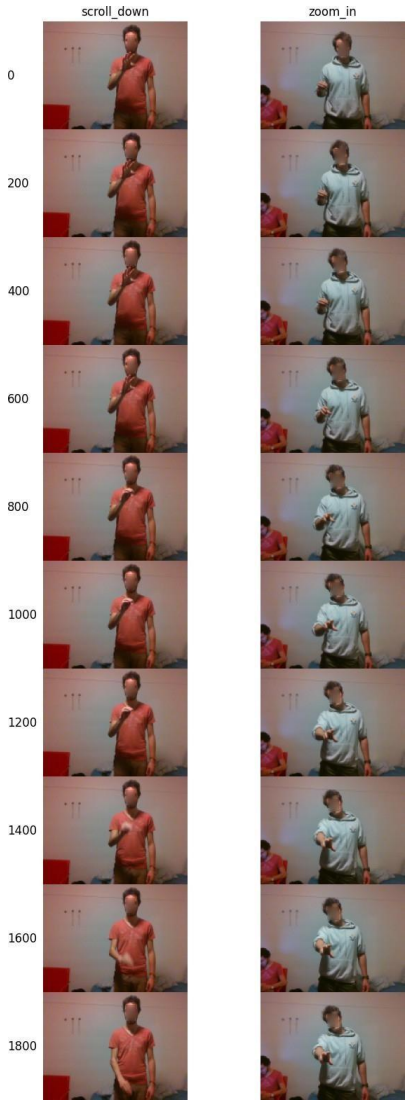

Fig. 2. Example of RGB dynamic hand gestures: scroll-down and zoom-in.

are encoded, resulting in a final input shape of (10, 120, 160, 3) for ten frames per gesture [5]. To balance accuracy with processing time, every fourth frame is used [13]. The dataset is shuffled before splitting to maintain uniform distribution across gestures. K-fold cross-validation is conducted for robust evaluation, averaging results across different folds [1]. Instances are provided unbatched, as final model fitting did not benefit from batching [3].

IV. RESULTS AND DISCUSSION

This end-to-end machine learning project was developed and run on Colab, using robust GPU resources for the deep learning workloads. The model was fitted on an NVIDIA A100-SXM GPU via TensorFlow v2.11.0. Training of the model was done at a 0.001 learning rate by categorical crossentropy as a loss function and Adam as an optimizer. Early stopping was used to prevent overfitting. Besides accuracy, the performance of the model is verified in terms of precision, recall, F1-score, and AUC-ROC. These metrics help explain better with respect to the balance of classes and their respective performance better than with this single measure. The model attained average precision of 0.93 and an average F1-score of 0.94 across all five folds, which would reflect a fine balance between precision and recall, suggesting that the model can do a good job in terms of distinguishing between the six classes of gestures. Average AUC-ROC scores, which directly reflect the ability of the classifier to distinguish between positive and negative classes at a wide range of thresholds, average at 0.95, with high discriminatory power.

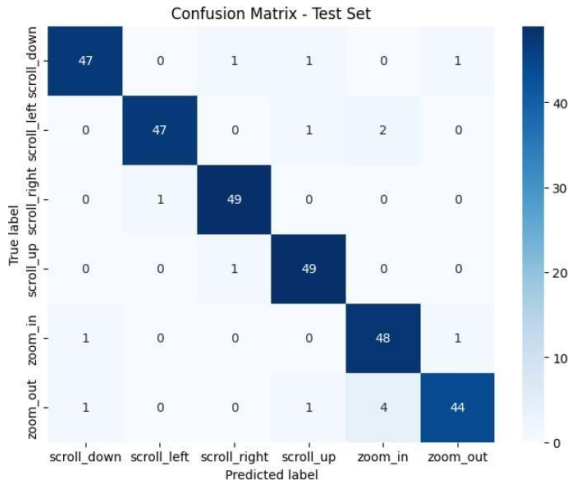

Fig. 3. Confusion matrix of test-data fold 1

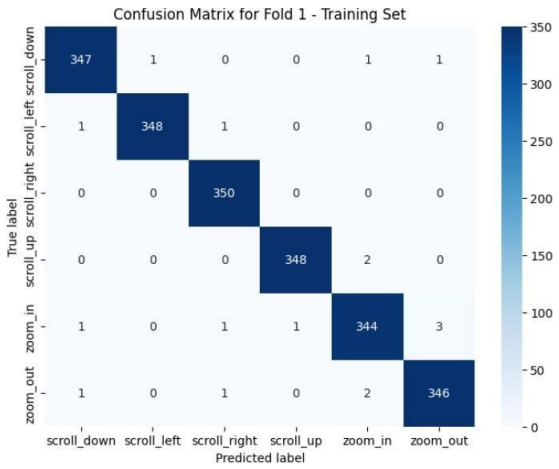

Fig. 4. Confusion matrix of training-data fold 1

A. Confusion Matrix and Analysis

The confusion matrix in Figure 3 shows that the model performed best on the ”scroll-right” gesture, achieving a True Positive Rate (TPR) of 98%. The lowest TPR was observed for the ”zoom-out” gesture, at 88%. Misclassifications occurred primarily between ”scroll-left” and ”zoom-out,” with approximately 4% of ”scroll-left” instances misclassified as ”zoom-

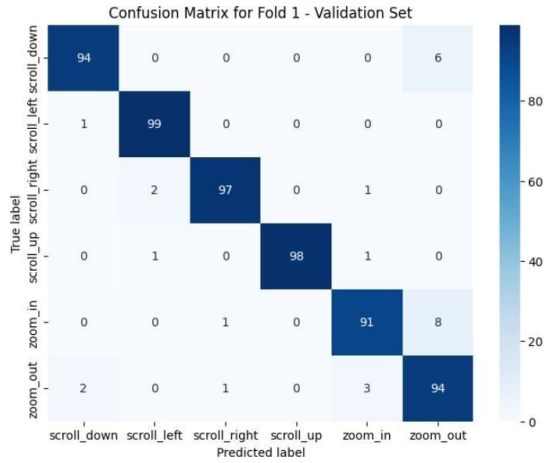

Fig. 5. Confusion matrix of validation-data fold 1

Fig. 6. Loss During Training.

out." Additionally, 8% of "zoom-in" instances were misclassified as "zoom-out." These errors may be attributed to the subtle visual similarities and overlapping movements between certain gestures, particularly between "zoom-in" and "zoom-out," as these two gestures involve similar directional hand motions, which can lead to ambiguity in model predictions. The same was true in the training and validation datasets presented by Figures 4 and 5, respectively.

B. Inception-v3 and LSTM Model Performance

The inception-v3 architecture does perform with significant success in pulling features at multiple scales, thus gaining popularity rather than the simple architecture of VGG16 and ResNet50. This is evident from the higher precision and recall scores. In order to maintain care of long-term dependencies in gesture sequences, LSTM layers have been incorporated instead of GRU. Though GRUs are computationally less expensive, LSTMs performed well in handling the temporal sequence of frames, which is a precursor for dynamic gestures recognition.

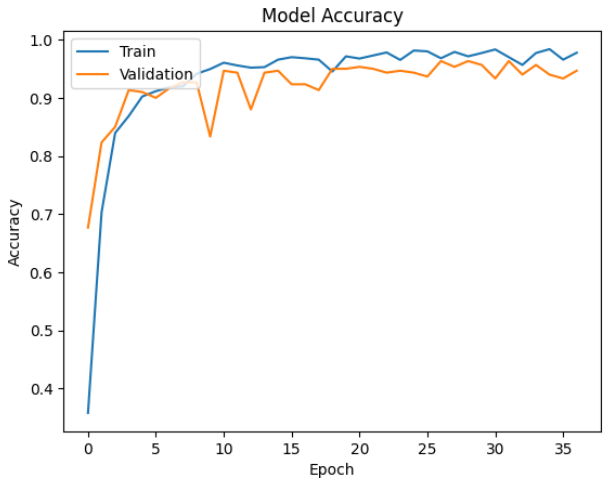

Fig. 7. Accuracy during training

C. Training and Validation Performance

Figures 3, 4, and 5 show the confusion matrices for training, validation, and testing, respectively. Overall, the model achieved 100% training accuracy, while maintaining an average validation accuracy of around 95.19% across the folds, indicating good generalization without overfitting. Additionally, to prevent further training once validation loss stabilized, early stopping was employed. Figures 6 and 7 illustrate the model's accuracy and loss during training.

D. Comparative Analysis

From the comparative analysis, it is noted that the Inception-LSTM model achieves superior performance for dynamic hand gesture recognition, with an average test accuracy of 95.59%. This model leverages the Inception module's multi-scale spatial feature extraction and LSTM's temporal analysis capabilities, making it particularly effective for recognizing gestures with subtle differences, such as "scroll-up" versus "scroll-right." Although the Inception-GRU model provides a similar accuracy with lower computational costs due to the efficiency of GRU, the Xception-LSTM and Xception-GRU models display lower accuracies, around 89-91%, which may be due to their simpler architectures. Consequently, the Inception-LSTM model strikes the best balance between accuracy and feature capture, making it especially suitable for detailed and dynamic gesture recognition.

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Train Accuracy (Inception-LSTM)	100.00%	99.64%	99.94%	99.88%	100.00%	99.89%
Validation Accuracy (Inception-LSTM)	0.01%	0.03%	0.01%	0.01%	0.00%	99.98%
Test Accuracy (Inception-LSTM)	95.71%	94.52%	94.05%	94.29%	97.38%	95.59%
Test Accuracy (Inception-GRU)	96.00%	96.00%	96.67%	96.33%	97.33%	96.47%
Test Accuracy (Xception-LSTM)	88.00%	89.78%	90.45%	88.54%	89.99%	89.55%
Test Accuracy (Xception-GRU)	91.98%	91.02%	90.00%	91.90%	91.90%	91.36%

TABLE II
ACHIEVED ACCURACIES OF ALL MODELS ACROSS ALL FOLDS.

As illustrated in Table 2, the Inception-LSTM model outperforms other models, including Inception-GRU and Xception-LSTM. The Inception-LSTM's average test accuracy of

95.59% indicates its superior ability to capture multi-scale features, with Xception-GRU showing the lowest average test accuracy at 91.36%. The deeper architecture and inception modules of Inception-v3 enhance spatial feature capture, while the LSTM's temporal capabilities contribute to higher overall performance, making Inception-LSTM highly effective for dynamic gesture recognition.

E. Future Work

Future work on dynamic hand gesture recognition could address several key areas to further enhance model performance. First, implementing advanced data augmentation techniques, such as geometric transformations and noise addition, would allow the model to generalize better across varied backgrounds and lighting conditions. This will improve robustness, especially in real-world applications where environments are less controlled. Second, transfer learning with domain-specific datasets, such as those specific to human-computer interaction, could enhance model adaptability and reduce training time when faced with smaller or unique gesture datasets.

Another promising direction is to incorporate lightweight CNN architectures and optimized recurrent units, such as GRUs, to make the model more suitable for mobile and embedded applications without compromising accuracy. This would expand the model's applicability to devices with limited computational resources, such as smartphones or wearable technology, for real-time gesture-based control. Finally, experimenting with attention mechanisms or transformer-based architectures could further improve the recognition accuracy by focusing on important temporal and spatial features within gesture sequences. This approach could especially help in distinguishing between visually similar gestures, such as "zoom-in" and "zoom-out." Overall, these enhancements aim to make gesture recognition systems not only more accurate but also more versatile and deployable in a wider range of interactive applications.

V. CONCLUSION

In this work, dynamic hand gestures recognition is revisited using deep learning techniques for accuracy and precision. We also implement several models such as Inception-GRU, Xception-LSTM, and Xception-GRU that can capture both spatial and temporal features from gesture sequences. These CNN-RNN hybrids turned out to be effective in detecting subtle hand movements, recording a mean accuracy of 95.59% in real-time. Early stopping was employed to avoid overfitting, while robustness of the model was insured through 5-fold cross-validation.Our results considerably improve upon many related works, especially in the more realistic settings when dealing with dynamic gestures. Other works often deal only with static gestures or smaller datasets, yet our approach could perform comparably even under difficult conditions. Future work will focus on fine-tuning these models for improved performance, optimizing the architecture for real-time operation on mobile devices with limited computational power, and expanding the dataset to enhance robustness across diverse environments. It may find applications in human-computer interaction, sign language interpretation, and gesture-based control systems.

REFERENCES

[1] S. Shin and W.-Y. Kim, "Skeleton-Based Dynamic Hand Gesture Recognition Using a Part-Based GRU-RNN for Gesture-Based Interface," *IEEE Access*, vol. 8, pp. 50236–50243, 2020.

[2] A. S. M. Miah, M. A. M. Hasan, and J. Shin, "Dynamic Hand Gesture Recognition Using Multi-Branch Attention-Based Graph and General Deep Learning Model," *IEEE Access*, vol. 11, pp. 4703-4716, 2023.

[3] W. Zhang and J. Wang, "Dynamic hand gesture recognition based on 3D convolutional neural network models," in *Proc. 2019 IEEE 16th Int. Conf. Networking, Sensing and Control (ICNSC)*, Banff, AB, Canada, 2019, pp. 363-368. doi: 10.1109/ICNSC.2019.8743159.

[4] C. Linqin, C. Shuangjie, X. Min, Y. Jimin, and Z. Jianrong, "Dynamic hand gesture recognition using RGB-D data for natural human-computer interaction," *Journal of Intelligent & Fuzzy Systems*, vol. 32, no. 5, pp. 3495-3507, April 2017. DOI:10.3233/JIFS-169287.

[5] M. U. Rehman, F. Ahmed, M. A. Khan, U. Tariq, F. Alfouzan, N. Alzahrani, and J. Ahmad, "Dynamic Hand Gesture Recognition Using 3D-CNN and LSTM Networks," *Computers, Materials & Continua*, September 2021. DOI:10.32604/cmc.2021.019586.

[6] B. K. Triwijoyo, L. Y. R. Karnaen, and A. Adil, "Deep Learning Approach for Sign Language Recognition," *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, vol. 9, no. 1, pp. 12-21, March 2023. DOI:10.26555/jiteki.v9i1.25051.

[7] Z. Ren, J. Meng, and J. Yuan, "Depth camera based hand gesture recognition and its applications in Human-Computer-Interaction," presented at the *International Conference on Information, Communications and Signal Processing (ICICS)*, December 2011. DOI:10.1109/ICICS.2011.6173545.

[8] D. Lu, C. Qiu, and Y. Xiao, "Temporal Convolutional Neural Network for Gesture Recognition," presented at the *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*, Singapore, 6-8 June 2018. DOI:10.1109/ICIS.2018.8466467.

[9] K. Bousbai and M. Merah, "A Comparative Study of Hand Gestures Recognition Based on MobileNetV2 and ConvNet Models," presented at the *2019 6th International Conference on Image and Signal Processing and their Applications (ISPA)*, Mostaganem, Algeria, 24-25 November 2019. DOI:10.1109/ISPA48434.2019.8966918.

[10] A. S. M. Miah, M. A. M. Hasan, and J. Shin, "Dynamic Hand Gesture Recognition Using Multi-Branch Attention Based Graph and General Deep Learning Model," *IEEE Access*, vol. 11, pp. 4703-4716, January 9, 2023. DOI:10.1109/ACCESS.2023.3235368.

[11] X. Tang, Z. Yan, J. Peng, B. Hao, H. Wang, and J. Li, "Selective spatiotemporal features learning for dynamic gesture recognition," *Expert Systems with Applications*, vol. 169, 1 May 2021, Article 114499. DOI:10.1016/j.eswa.2020.114499.

[12] S. Jeeru, A. K. Sivapuram, D. G. Leo´n, J. Gro¨li, S. R. Yeduri, and L. R. Cenkeramaddi, "Depth Camera Based Dataset of Hand Gestures," *Data Brief*, vol. 45, Dec. 2022, Art. no. 108659. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340922008642.

[13] D. R. T. Hax, P. Penava, S. Krodel, L. Razova, and R. Buettner, "A Novel Hybrid Deep Learning Architecture for Dynamic Hand Gesture Recognition," *IEEE Access*, vol. 12, pp. 28761-28773, Feb. 2024. DOI:10.1109/ACCESS.2024.3365274.

[14] D. Wang, C. Li, M. Chen, and L. Zhang, "Gesture recognition in complex environments using CNNs and LSTMs," *Pattern Recognition*, vol. 102, pp. 107241, 2023. DOI:10.1016/j.patcog.2022.107241.

[15] S. Harris and A. Kumar, "Robust spatiotemporal modeling for dynamic gesture recognition," *IEEE Transactions on Multimedia*, vol. 25, pp. 3601-3613, 2024. DOI:10.1109/TMM.2024.3295471.

ICIICS552.pdf

ORIGINALITY REPORT

15%	9%	11%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	David Richard Tom Hax, Pascal Penava, Samira Krodel, Liliya Razova, Ricardo Buettner. "A Novel Hybrid Deep Learning Architecture for Dynamic Hand Gesture Recognition", IEEE Access, 2024 Publication	3%
2	Submitted to Holy Child Central Colleges Inc. Student Paper	2%
3	doaj.org Internet Source	1%
4	arxiv.org Internet Source	1%
5	sciendo.com Internet Source	1%
6	antycovid.iitis.pl Internet Source	1%
7	d197for5662m48.cloudfront.net Internet Source	1%

8	S V N Sreenivasu, Sakshi Gupta, Ghanshyam Vatsa, Anurag Shrivastava, Swati Vashisht, Aparna Srivastava. "Carbohydrate Recommendation for Type-1 Diabetics Patient Using Machine Learning", 2022 5th International Conference on Contemporary Computing and Informatics (IC3I), 2022 Publication	< 1 %
9	Submitted to University of Hull Student Paper	< 1 %
	eprints.nottingham.ac.uk Internet Source	< 1 %
	Jun Xu, Hanchen Wang, Jianrong Zhang, Linqin Cai. "Robust Hand Gesture Recognition Based on RGB-D Data for Natural Human-Computer Interaction", IEEE Access, 2022 Publication	< 1 %
	Submitted to City University of Hong Kong Student Paper	< 1 %
13	M.S. Karthika, Harikumar Rajaguru, Ajin R. Nair. "Wavelet feature extraction and bio-inspired feature selection for the prognosis of lung cancer – A statistical framework analysis", Measurement, 2024 Publication	< 1 %
	Olukoga, Toluwalase Adeola. "Machine Learning Applications for Optimization of Oil	< 1 %

	and Gas System Development", University of Louisiana at Lafayette, 2024 Publication	
	scpe.org Internet Source	<1%
	assets.researchsquare.com Internet Source	<1%
	gyan.iitg.ac.in Internet Source	<1%
18	ijsrem.com Internet Source	<1%
	pubmed.ncbi.nlm.nih.gov Internet Source	<1%
	www.frontiersin.org Internet Source	<1%
	www.jatit.org Internet Source	<1%
	Farzaneh Jafari, Anup Basu. "Two-Dimensional Parallel Spatio-Temporal Pyramid Pooling for Hand Gesture Recognition", IEEE Access, 2023 Publication	<1%
	M. Ravinder, K. Malik, M. Hassaballah, U. Tariq, K. Javed, M. Ghoneimy. "An Approach for Gesture Recognition Based on a Lightweight Convolutional Neural Network",	<1%

	International Journal on Artificial Intelligence Tools, 2022 Publication	
	Mehdi Ghayoumi. "Generative Adversarial Networks in Practice", CRC Press, 2023 Publication	< 1 %
25	cdn.techscience.cn Internet Source	< 1 %
	konferenciaonline.org.ua Internet Source	< 1 %
	link.springer.com Internet Source	< 1 %
	preview.aclanthology.org Internet Source	< 1 %
	upcommons.upc.edu Internet Source	< 1 %
	www.ijert.org Internet Source	< 1 %
	www.mdpi.com Internet Source	< 1 %
	Rohit Pratap Singh, Laiphrakpam Dolendro Singh. "Dyhand: dynamic hand gesture recognition using BiLSTM and soft attention methods", The Visual Computer, 2024 Publication	< 1 %

Seunghyeok Shin, Whoi-Yul Kim. "Skeleton-Based Dynamic Hand Gesture Recognition Using a Part-Based GRU-RNN for Gesture-Based Interface", IEEE Access, 2020

Publication

<1%

El-Sayed M. El-Alfy, Hamzah Luqman. "A comprehensive survey and taxonomy of sign language research", Engineering Applications of Artificial Intelligence, 2022

Publication

<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

Centenary Celebrated Sharnbasveshwar Vidya Vardhak Sangha's

ಶರಣಬಸವ ವಿಶ್ವವಿದ್ಯಾಲಯ
SHARNBASVA UNIVERSITY

Kalaburagi – 585103, Karnataka · India

A State Private University approved by Govt. of Karnataka vide Notification No. ED 144 URC 2016 dated 29-07-2017,
Recognised by UGC under Section 2f vide No. F.8-29/2017 (CPP-I/PU), dated 20-12-2017 & AICTE, COA, PCI New Delhi

IEEE BANGALORE SECTION

IEEE NKSS

Certificate of Appreciation

This is to certify that Dr./Prof./Mr./Ms

T. G. Ramnadh Babu

has **Presented** Paper entitled

Hand Gesture Recognition: Enhancing Accuracy and Precision with Deep Learning

for 2nd IEEE International Conference on
Integrated Intelligence and Communication Systems (ICIICS-2024)
organized by Sharnbasva University, Kalaburagi, 22-23 November, 2024

Dr. Lakshmi Patil Maka
Conference Chair and Convenor, ICIICS–2024
Dean, Sharnbasva University,
Kalaburagi,

Dr. Anilkumar G. Bidve
Vice-Chancellor,
Sharnbasva University,
Kalaburagi

Parama Poojya Dr. Sharnbaswappa Appaji
Chancellor,
Sharnbasva University,
Kalaburagi

Centenary Celebrated Sharnbasveshwar Vidya Vardhak Sangha's

ಶರಣಬಸವ ವಿಶ್ವವಿದ್ಯಾಲಯ
SHARNBASVA UNIVERSITY

Kalaburagi – 585103, Karnataka · India

A State Private University approved by Govt. of Karnataka vide Notification No. ED 144 URC 2016 dated 29-07-2017,
Recognised by UGC under Section 2f vide No. F.8-29/2017 (CPP-I/PU), dated 20-12-2017 & AICTE, COA, PCI New Delhi

IEEE BANGALORE SECTION

IEEE NKSS

Certificate of Appreciation

This is to certify that Dr./Prof./Mr./Ms

Dodda Venkata Reddy

has **Presented** Paper entitled

Hand Gesture Recognition: Enhancing Accuracy and Precision with Deep Learning

for 2nd IEEE International Conference on
Integrated Intelligence and Communication Systems (ICIICS-2024)
organized by Sharnbasva University, Kalaburagi, 22-23 November, 2024

Dr. Lakshmi Patil Maka
Conference Chair and Convenor, ICIICS–2024
Dean, Sharnbasva University,
Kalaburagi,

Dr. Anilkumar G. Bidve
Vice-Chancellor,
Sharnbasva University,
Kalaburagi

Parama Poojya Dr. Sharnbaswappa Appaji
Chancellor,
Sharnbasva University,
Kalaburagi

Centenary Celebrated Sharnbasveshwar Vidya Vardhak Sangha's

ಶರಣಬಸವ ವಿಶ್ವವಿದ್ಯಾಲಯ
SHARNBASVA UNIVERSITY

Kalaburagi – 585103, Karnataka · India

A State Private University approved by Govt. of Karnataka vide Notification No. ED 144 URC 2016 dated 29-07-2017,
Recognised by UGC under Section 2f vide No. F.8-29/2017 (CPP-I/PU), dated 20-12-2017 & AICTE, COA, PCI New Delhi

IEEE BANGALORE SECTION

IEEE NKSS

Certificate of Appreciation

This is to certify that Dr./Prof./Mr./Ms

S.V.N. Sreenivasu

has **Presented** Paper entitled

Hand Gesture Recognition: Enhancing Accuracy and Precision with Deep Learning

for 2nd IEEE International Conference on
Integrated Intelligence and Communication Systems (ICIICS-2024)
organized by Sharnbasva University, Kalaburagi, 22-23 November, 2024

Dr. Lakshmi Patil Maka
Conference Chair and Convenor, ICIICS–2024
Dean, Sharnbasva University,
Kalaburagi,

Dr. Anilkumar G. Bidve
Vice-Chancellor,
Sharnbasva University,
Kalaburagi

Parama Poojya Dr. Sharnbaswappa Appaji
Chancellor,
Sharnbasva University,
Kalaburagi

Centenary Celebrated Sharnbasveshwar Vidya Vardhak Sangha's

ಶರಣಬಸವ ವಿಶ್ವವಿದ್ಯಾಲಯ
SHARNBASVA UNIVERSITY

Kalaburagi – 585103, Karnataka · India

A State Private University approved by Govt. of Karnataka vide Notification No. ED 144 URC 2016 dated 29-07-2017,
Recognised by UGC under Section 2f vide No. F.8-29/2017 (CPP-I/PU), dated 20-12-2017 & AICTE, COA, PCI New Delhi

IEEE BANGALORE SECTION

IEEE NKSS

Certificate of Appreciation

This is to certify that Dr./Prof./Mr./Ms

K. Srinivasa Kalyan Ram

has **Presented** Paper entitled

Hand Gesture Recognition: Enhancing Accuracy and Precision with Deep Learning

for 2nd IEEE International Conference on
Integrated Intelligence and Communication Systems (ICIICS-2024)
organized by Sharnbasva University, Kalaburagi, 22-23 November, 2024

Dr. Lakshmi Patil Maka
Conference Chair and Convenor, ICIICS–2024
Dean, Sharnbasva University,
Kalaburagi,

Dr. Anilkumar G. Bidve
Vice-Chancellor,
Sharnbasva University,
Kalaburagi

Parama Poojya Dr. Sharnbaswappa Appaji
Chancellor,
Sharnbasva University,
Kalaburagi

Centenary Celebrated Sharnbasveshwar Vidya Vardhak Sangha's

ಶರಣಬಸವ ವಿಶ್ವವಿದ್ಯಾಲಯ
SHARNBASVA UNIVERSITY

Kalaburagi – 585103, Karnataka · India

A State Private University approved by Govt. of Karnataka vide Notification No. ED 144 URC 2016 dated 29-07-2017,
Recognised by UGC under Section 2f vide No. F.8-29/2017 (CPP-I/PU), dated 20-12-2017 & AICTE, COA, PCI New Delhi

IEEE BANGALORE SECTION

IEEE NKSS

Certificate of Appreciation

This is to certify that Dr./Prof./Mr./Ms

Venkata Aravind Tunuguntla

has **Presented** Paper entitled

Hand Gesture Recognition: Enhancing Accuracy and Precision with Deep Learning

for 2nd IEEE International Conference on
Integrated Intelligence and Communication Systems (ICIICS-2024)
organized by Sharnbasva University, Kalaburagi, 22-23 November, 2024

Dr. Lakshmi Patil Maka
Conference Chair and Convenor, ICIICS–2024
Dean, Sharnbasva University,
Kalaburagi,

Dr. Anilkumar G. Bidve
Vice-Chancellor,
Sharnbasva University,
Kalaburagi

Parama Poojya Dr. Sharnbaswappa Appaji
Chancellor,
Sharnbasva University,
Kalaburagi

Centenary Celebrated Sharnbasveshwar Vidya Vardhak Sangha's

ಶರಣಬಸವ ವಿಶ್ವವಿದ್ಯಾಲಯ
SHARNBASVA UNIVERSITY

Kalaburagi – 585103, Karnataka · India

A State Private University approved by Govt. of Karnataka vide Notification No. ED 144 URC 2016 dated 29-07-2017,
Recognised by UGC under Section 2f vide No. F.8-29/2017 (CPP-I/PU), dated 20-12-2017 & AICTE, COA, PCI New Delhi

IEEE BANGALORE SECTION

IEEE NKSS

Certificate of Appreciation

This is to certify that Dr./Prof./Mr./Ms

Shaik Hazare Khaja Mohiddin

has **Presented** Paper entitled

Hand Gesture Recognition: Enhancing Accuracy and Precision with Deep Learning

for 2nd IEEE International Conference on
Integrated Intelligence and Communication Systems (ICIICS-2024)
organized by Sharnbasva University, Kalaburagi, 22-23 November, 2024

Dr. Lakshmi Patil Maka
Conference Chair and Convenor, ICIICS–2024
Dean, Sharnbasva University,
Kalaburagi,

Dr. Anilkumar G. Bidve
Vice-Chancellor,
Sharnbasva University,
Kalaburagi

Parama Poojya Dr. Sharnbaswappa Appaji
Chancellor,
Sharnbasva University,
Kalaburagi