IMAGE RECOGINITION USING DEEP LEARNING

Major project report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

By

CH JITENDRA (20UECS0169) (VTU17650)

Under the guidance of
Dr. Jose P, M.E, Ph.D.,
ASSOCIATE PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

May, 2024

IMAGE RECOGINITION USING DEEP LEARNING

Major project report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

By

CH JITENDRA (20UECS0169) (VTU17650)

Under the guidance of
Dr. Jose P, M.E, Ph.D.,
ASSOCIATE PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

May, 2024

CERTIFICATE

It is certified that the work contained in the project report titled ”IMAGE RECOGINITION USING DEEP LEARNING” by ”CH JITENDRA (20UECS0169)” has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
May, 2024

Signature of Professor In-charge
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
May, 2024

DECLARATION

I declare that this written submission represents the ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Ch Jitendra

Date: / /

APPROVAL SHEET

This project report entitled ”IMAGE RECOGINITION USING DEEP LEARNING” by Ch Jitendra (20UECS0169) is approved for the degree of B.Tech in Computer Science & Engineering.

Examiners **Supervisor**

Dr. Jose P, M.E, Ph.D., ASSOCIATE PROFESSOR

Date: / /

Place: Chennai

ACKNOWLEDGEMENT

I express my deepest gratitude to our respected **Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (EEE), B.E. (MECH), M.S (AUTO),D.Sc., Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.** Chairperson Managing Trustee and Vice President.

I am very much grateful to our beloved **Vice Chancellor Prof. S. SALIVAHANAN,** for providing me with an environment to complete my project successfully.

I record indebtedness to our **Professor & Dean, Department of Computer Science & Engineering, School of Computing, Dr. V. SRINIVASA RAO, M.Tech., Ph.D.,** for immense care and encouragement towards us throughout the course of this project.

I are thankful to our **Head, Department of Computer Science & Engineering, Dr.M.S. MURALI DHAR, M.E., Ph.D.,** for providing immense support in all our endeavors.

I also take this opportunity to express a deep sense of gratitude to our Internal Supervisor **Dr. JOSE P, M.E, Ph.D.,** for her cordial support, valuable information and guidance, she helped me in completing this project through various stages.

A special thanks to our **Project Coordinators Mr. V. ASHOK KUMAR, M.Tech., Ms. C. SHYAMALA KUMARI, M.E.,** for their valuable guidance and support throughout the course of the project.

I thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

Ch Jitendra (20UECS0169)

ABSTRACT

This proposes a deep learning-based approach for the automated detection and classification of retinoblastoma using a specialized dataset.Leveraging convolutional neural networks (CNNs), specifically tailored for retinoblastoma images, our framework achieves accurate identification of tumor presence and characterization. Through extensive experimentation, including transfer learning with pre-trained CNN architectures and data augmentation techniques, our model demonstrates robust performance across diverse retinoblastoma image modalities. Results indicate high accuracy, sensitivity, and specificity, highlighting the potential of our approach for early diagnosis and personalized treatment planning.Efficiency in image recognition using deep learning is achieved through optimized model architectures and streamlined inference processes, ensuring rapid and accurate classification with minimal computational overhead. By providing clinicians with a reliable tool for efficient image analysis, our system aims to improve patient outcomes and streamline clinical decision-making in retinoblastoma management.We obtained an accuracy of 92.86

Keywords: Pre Processing, Convolutional Neural Network, Computer Vision, Feauture Extraction, Classification

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

AP	Average Precision
CNN	Convolutional Neural Network
CPU	Central Processing Unit
FC	Fully Connected
GPU	Graphics Processing Unit
MLP	Multilayer Perceptron
ReLU	Rectified Linear Unit
R-CNN	Region-based Convolutional Neural Network

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction

Image recognition using deep learning revolutionizes computer vision by employing convolutional neural networks (CNNs) to automatically learn hierarchical representations of visual data. Unlike traditional methods reliant on handcrafted features, deep learning algorithms can directly extract features from raw pixel values, enabling machines to interpret and understand images with remarkable accuracy. Through the training process on large labeled datasets, CNNs adjust their parameters to minimize prediction errors, achieving state-of-the-art performance in tasks such as image classification, object detection, and facial recognition. Through the iterative process of training on vast labeled datasets, these models refine their parameters to minimize errors, achieving unprecedented levels of accuracy in tasks such as image classification, object detection, and semantic segmentation. Transfer learning further enhances efficiency by leveraging pre-trained models to adapt to new tasks with limited labeled data. The advent of transfer learning further accelerates progress, allowing pre-trained models to be fine-tuned on specific tasks, often with minimal additional labeled data. While challenges persist, including dataset diversity, model interpretability, and ethical considerations, the potential applications of deep learning in image recognition span diverse domains, from healthcare and autonomous systems to augmented reality and beyond. As research continues to push the boundaries of what's possible, the future holds promise for even greater strides in understanding and harnessing visual information through deep learning techniques.

1.2 Aim of the Project

The aim of this project is to develop a deep learning-based image recognition system for the automated detection and classification of retinoblastoma using a specialized dataset. This system aims to accurately identify the presence of retinoblastoma tumors in retinal images and classify them based on characteristics such as size,

location, and morphology. By leveraging deep learning techniques, including convolutional neural networks (CNNs), the project seeks to enhance the efficiency and accuracy of retinoblastoma diagnosis, ultimately facilitating early detection, timely intervention, and personalized treatment planning for affected patients.

1.3 Project Domain

The project revolves around the domain of healthcare, specifically focusing on liver disorders diagnosis utilizing machine learning techniques. Liver diseases encompass a broad spectrum of conditions ranging from fatty liver disease to cirrhosis and hepatocellular carcinoma, posing significant challenges for accurate diagnosis and timely intervention. Leveraging advancements in machine learning, this project aims to develop and evaluate computational models capable of analyzing diverse data sources including clinical records, biochemical markers, and medical imaging to assist healthcare professionals in diagnosing liver disorders with improved accuracy and efficiency.

The project domain intersects the fields of medicine, computer science, and data analytics, reflecting the interdisciplinary nature of modern healthcare solutions. By harnessing machine learning algorithms, the project seeks to address critical gaps in current diagnostic approaches by providing automated tools for early detection, risk stratification, and personalized treatment planning in liver disease management. Through collaboration with medical experts and leveraging real-world datasets, this project endeavors to contribute to the advancement of precision medicine and improve patient outcomes in the realm of hepatology.

1.4 Scope of the Project

The project aims to develop a deep learning-based image recognition system for the detection and classification of retinoblastoma using a specialized dataset. It involves collecting retinoblastoma images from various sources and preprocessing them to ensure consistency and quality. The scope includes designing and implementing Convolutional Neural Network (CNN) architectures, leveraging transfer learning techniques, and employing data augmentation strategies to enhance model

performance and generalization. Evaluation will focus on metrics such as accuracy, sensitivity, and specificity, with validation conducted using separate test datasets. The system's integration into a user-friendly interface for clinical use and adherence to ethical considerations regarding patient data and model transparency are integral aspects of the project. Future directions may include refining the system, exploring multi-modal imaging analysis, and collaborating with healthcare institutions for real-world validation and adoption.

Chapter 2

LITERATURE REVIEW

[1]] J. Deng, W. Dong et al [2022] Health effects of overweight and obesity in 195 countries over 25 years, New England Journal of Medicine 377 , no. 1, 13–27, PMID: 28604169. It likely provided valuable insights into the global burden of excess weight on health, potentially addressing factors such as disease prevalence, mortality rates, and associated risk factors across diverse populations.

[2] Y. He, C. Xu, N. Khanna et al [2022]"Analysis of food images: Features and classification," IEEE International Conference on Image Processing (ICIP), Paris, pp. 2744-27. Image Processing (ICIP) in Paris, focused on the analysis of food images. It likely explored various features and classification techniques to categorize and analyze food images, potentially aiming to assist in dietary assessment, nutrition monitoring, or food recognition applications. The study may have examined approaches such as feature extraction, machine learning algorithms, and image classification methodologies to understand and classify different types of food items depicted in images..

[3] Z. Zong et al. [2022]"On the Combination of Local Texture and Global Structure for Food Classification," IEEE International Symposium on Multimedia, Taichung, pp. 204- 211. Multimedia in Taichung, likely focused on food classification using a combination of local texture and global structure features. It may have explored methods to effectively integrate both local and global information from food images to improve classification accuracy. The study likely employed techniques such as texture analysis, feature extraction, and machine learning algorithms to distinguish between different types of food items based on their visual characteristics.

[4] S. J. Minija et al. [2022]"Food image classification using sphere shaped — Support vector machine," 2022 International Conference on Inventive Computing and Informatics (ICICI), Coimbatore, pp. 109-113.This focused on food image classification using a sphere-shaped support vector machine (SVM). The study likely

explored the application of SVM, a popular machine learning algorithm, in classifying food images. The use of a sphere-shaped SVM suggests a specialized approach tailored to the characteristics of food image data, potentially aiming to improve classification accuracy and efficiency in food recognition tasks.

[5] Lukas Bossard et al.[2023] Food-101 mining discriminative components with random forests, ComputerVision – ECCV 2014 (Cham) (David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, eds.), Springer International Publishing, pp. 446–461. The study likely aimed to address challenges in food recognition by leveraging machine learning techniques to identify key features that distinguish different food categories. By mining discriminative components, the project may have contributed to improving the accuracy and robustness of food image classification systems, potentially benefiting applications in dietary assessment, nutrition monitoring, and food recognition technologies.

[6]Kaliki S et al.[2023] Retinoblastoma in India: clinical presentation and outcome in 1,457 patients (2,074 Eyes). Retina. 39:379–439.The project described in the article aimed to analyze the clinical presentation and outcomes of retinoblastoma, a type of eye cancer, in India. The study included a large sample of 1,457 patients with 2,074 affected eyes. Researchers likely gathered data on various aspects such as age at diagnosis, tumor characteristics, treatment modalities, and survival rates to provide insights into the disease's profile and management in the Indian context.

[7]Abdolvahabi A, Taylor BW, Holden RL, et al.[2024] Colorimetric and longitudinal analysis of leukocoria in recreational photographs of children with retinoblastoma. PLOS One.8:E76677. 23. The study conducted by Abdolvahabi et al. focused on analyzing leukocoria, which is a white eye reflection often indicative of retinoblastoma, in recreational photographs of children. They utilized colorimetric and longitudinal analyses to detect and track leukocoria in these images. By employing advanced algorithms, the researchers aimed to automatically identify and quantify leukocoria across multiple photographs taken over time. This approach allowed for non-invasive screening and monitoring of retinoblastoma in children, potentially facilitating early detection and intervention.

[8]Rivas-Perea P et al [2022]Detection of leukocoria using a soft fusion of expert

classifiers under non-clinical settings. BMC Ophthalmol.14:110.The research conducted by Rivas-Perea et al. focused on the detection of leukocoria, an important indicator of eye diseases like retinoblastoma, under non-clinical settings. The study utilized a soft fusion approach, combining multiple expert classifiers to improve the accuracy of leukocoria detection. Unlike traditional binary classifiers, the soft fusion method allowed for a more nuanced and flexible approach to decision-making, enhancing the detection performance. By leveraging this technique, the researchers aimed to enhance the sensitivity and specificity of leukocoria detection in various non-clinical settings, such as home environments or community screenings. The study, published in BMC Ophthalmology, contributes to the development of advanced tools and methods for early detection of eye diseases, ultimately improving patient outcomes through timely intervention.

[9]Raoof N, Dai S.et al[2023] Red Reflex Screening In New Zealand: a large survey of practices and attitudes in the auckland region. N Z Med J. 129:38–43.The researchers conducted a large survey to gather data on the prevalence of red reflex screening practices among healthcare providers in Auckland, as well as their attitudes towards its efficacy and importance. By assessing current practices and attitudes, the study aimed to identify areas for improvement and potential barriers to implementing red reflex screening more widely. This research contributes to understanding the landscape of eye health screening in New Zealand and may inform efforts to enhance early detection of eye conditions in the region.

[10]Cagini C, Tosi G, Stracci F, et al.[2023]Red reflex examination in neonates: evaluation of 3 years of screening. Int Ophthalmol. 37:1199–1204Conducting a retrospective analysis, they likely examined data on the screenings performed, detected abnormalities, and subsequent management. The study likely aimed to determine the effectiveness of red reflex screening in early detection of eye abnormalities in newborns, providing insights into the importance of this screening in neonatal care and potentially influencing healthcare practices regarding newborn eye health.

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

This project seeks to optimize and enhance an existing image recognition system through the application of advanced deep learning techniques. The current system serves as a foundation, which will be augmented and refined to achieve superior accuracy, speed, and robustness in image recognition tasks. Leveraging state-of-the-art deep learning architectures like ResNet, EfficientNet, and DenseNet, along with optimization methods such as transfer learning and model distillation, the aim is to significantly boost the system's performance. Additionally, data augmentation techniques will be employed to enrich the training dataset and improve the model's generalization capabilities. Through rigorous experimentation, comparative analysis, and real-world testing, this project aims to deliver a refined image recognition system that surpasses the capabilities of the existing system, paving the way for more reliable and efficient image recognition solutions across various domains.

3.1.1 Advantages of Existing System

- **Established Framework:**The existing system provides a well-established framework and infrastructure for image recognition tasks, including data pipelines, preprocessing methods, and model integration. This framework serves as a solid starting point for building upon and refining the image recognition capabilities.

- **Baseline Performance Metrics:** By having an existing system in place, there are baseline performance metrics available for comparison and evaluation. This allows for a clear understanding of the system's current strengths and weaknesses, guiding the optimization efforts towards areas that require improvement.

- **Domain Knowledge:** The development and deployment of the existing system have likely involved domain-specific knowledge and expertise relevant to the application context. Leveraging this domain knowledge can expedite the optimization process and ensure that the enhanced system remains aligned with the specific requirements and constraints of the target domain.

- **Access to Existing Data:** The existing system is likely to have access to a repository of labeled data used for training and validation purposes. This existing dataset can be leveraged for further experimentation, fine-tuning of models, and validation of improvements, reducing the need for additional data collection efforts.

- **Maintaining Compatibility and Integration:** Seamless integration with other systems, tools, and processes within the organization's ecosystem. This compatibility minimizes integration challenges, reduces the risk of compatibility issues, and facilitates interoperability with existing software and hardware components. Maintaining compatibility streamlines deployment and adoption of the enhanced image recognition solution across the organization.

3.2 Proposed System

The proposed system for image recognition harnesses the power of deep learning to accurately identify and classify images across diverse domains. At its core, the system comprises several interconnected components aimed at streamlining the process from data collection to model deployment. Initially, a comprehensive dataset of labeled images is gathered, ensuring representation across all pertinent categories. These images undergo preprocessing techniques, including resizing and augmentation, to enhance the robustness and generalization capabilities of the subsequent model. The model selection phase involves careful consideration of various deep learning architectures, such as Convolutional Neural Networks (CNNs) like ResNet tailored to the specific requirements of the task. Through iterative model training and optimization, facilitated by optimization algorithms and loss functions, the system learns to extract meaningful features and make accurate predictions.Rigorous evaluation on a separate validation dataset ensures the model's performance and generalization ability meet predefined standards. Once validated, the model is seamlessly integrated into the deployment pipeline, enabling real-time inference on new, unseen images. Continuous monitoring and maintenance guarantee the system's reliability and effectiveness over time, ensuring it remains at the forefront of image recognition technology, delivering superior accuracy and efficiency across a multitude of applications and domains.

3.2.1 Advantages of Proposed system

- **High Accuracy::** Leveraging advanced deep learning architectures enables the system to achieve high levels of accuracy in image recognition tasks. These architectures, such as Convolutional Neural Networks (CNNs) like ResNet, EfficientNet, or DenseNet, are specifically designed to extract and learn intricate features from images, resulting in more precise classifications..

- **ARobustnessRobustness** Through rigorous training and validation, the system develops robustness to variations in input data, such as changes in lighting conditions, background clutter, or object occlusions. This robustness ensures reliable performance across diverse environments and scenarios..

- **Efficiency:** Deep learning techniques optimize computational resources, leading to efficient inference times even on resource-constrained devices. Techniques like model quantization and pruning further enhance efficiency without compromising accuracy.

- **Flexibility:** The proposed system can be adapted to various image recognition tasks and domains by fine-tuning the model architecture and training parameters. This flexibility allows for customization based on specific requirements and constraints..

- **Interpretability:** Deep learning models can provide insights into the decision-making process, allowing users to understand how and why certain classifications are made. This interpretability is crucial for applications where transparency and accountability are paramount.

- **Scalability:** The modular nature of deep learning frameworks allows for scalable deployment, enabling the system to handle large datasets and increasing computational demands as needed. This scalability ensures that the system remains effective and efficient as requirements grow over time.

3.3 Feasibility Study

In image recognition system employing deep learning, several critical factors are assessed to determine the project's viability. Technical feasibility involves evaluating the availability of hardware resources like GPUs for efficient model training and assessing the compatibility of deep learning frameworks with existing infrastructure. Additionally, data feasibility entails examining the quality and diversity of labeled datasets essential for training robust models, while ensuring compliance with data privacy regulations. Financial feasibility is essential, estimating costs for hardware, software, and ongoing operational expenses, and conducting a cost-benefit analysis to ascertain the project's potential return on investment. Market feasibility involves analyzing demand, identifying potential users, and evaluating the competitive landscape to gauge the system's market positioning. Legal and ethical considerations

are paramount, ensuring compliance with regulations and addressing ethical implications such as bias and privacy concerns. Finally, operational feasibility assesses organizational readiness, including personnel expertise and the potential impact on existing workflows.

Once these aspects are thoroughly evaluated, stakeholders can make informed decisions regarding the project's viability and potential success. By identifying technical constraints, ensuring data availability and quality, and addressing financial, market, legal, and operational considerations, the feasibility study provides a comprehensive understanding of the project's risks and opportunities. Through this analysis, stakeholders can determine whether to proceed with developing and implementing the image recognition system using deep learning, laying the groundwork for a successful and sustainable endeavor.

3.3.1 Economic Feasibility

The economic feasibility of image recognition system powered by deep learning involves scrutinizing the financial implications associated with its development, deployment, and maintenance. Initially, the project requires investment in hardware infrastructure such as GPUs or TPUs for efficient model training, as well as software resources including deep learning frameworks like TensorFlow or PyTorch. Additionally, expenses related to acquiring labeled datasets and potential costs for cloud computing services must be considered. Ongoing operational costs, including maintenance, updates, and potential licensing fees for proprietary software, further contribute to the economic feasibility evaluation. Despite these initial and recurring expenses, the economic feasibility is determined by comparing the projected benefits with the costs incurred. Potential benefits include increased efficiency in image recognition tasks, reduced labor costs, and improved decision-making processes. Furthermore, the system's ability to enhance productivity and generate revenue through improved services or products can significantly impact its economic feasibility. Conducting a comprehensive cost-benefit analysis enables stakeholders to gauge the project's potential return on investment and make informed decisions about its economic viability, ensuring prudent allocation of resources and maximizing long-term profitability.

3.3.2 Technical Feasibility

Assessing the technical feasibility of an image recognition system utilizing deep learning entails evaluating various factors related to its development, implementation, and integration within existing infrastructure. Firstly, the availability of necessary hardware resources, such as GPUs or TPUs, for efficient model training is paramount. Deep learning frameworks like TensorFlow or PyTorch must also be compatible with the existing infrastructure and development environment to facilitate seamless implementation. Furthermore, the feasibility of optimizing and fine-tuning deep learning algorithms for image recognition within the project's technical constraints must be considered. Additionally, the availability and quality of labeled image datasets essential for training robust models play a crucial role in technical feasibility. Ensuring compliance with data privacy regulations and addressing concerns regarding data security are integral aspects of this evaluation. By thoroughly assessing these technical considerations, stakeholders can ascertain the feasibility of developing and implementing an image recognition system using deep learning, ensuring its compatibility, efficiency, and effectiveness within the intended environment.

3.3.3 Social Feasibility

Social feasibility examines the broader societal impact and acceptance of implementing an image recognition system powered by deep learning. Beyond technical and economic considerations, understanding societal attitudes, ethical implications, and potential consequences is essential. Firstly, addressing concerns related to privacy and data security is crucial, as the system may involve processing personal or sensitive information. Ensuring transparency and accountability in how data is collected, used, and protected helps build trust among users and stakeholders. Additionally, addressing potential biases in the system's algorithms and ensuring fairness and equity in its outcomes is paramount. Stakeholder engagement and involvement in the development process, including feedback from diverse communities, can help identify and mitigate potential social risks and ensure the system's alignment with societal values and norms. Moreover, educating the public about the capabilities and limitations of the system, as well as its potential benefits and risks, fosters informed discussions and promotes social acceptance. By proactively addressing social con-

cerns and engaging with stakeholders, the image recognition system can be developed and implemented in a manner that garners widespread support and contributes positively to society.

3.4 System Specification

3.4.1 Hardware Specification

- Processor : Pentium Dual Core 2.00GHZ
- Hard disk : 120 GB
- RAM : 2GB (minimum)
- Keyboard : 110 keys enhanced

3.4.2 Software Specification

- Operating system : Windows7 (with service pack 1), 8, 8.1 and 10
- Language : Python
- Libraries : Pytorch,Keras,Tensor flow,Scikit-learn

LANGUAGE SPECIFICATION - PYTHON

Among programmers, Python has emerged as a dominant language in deep learning due to its versatility, ease of use, and robust ecosystem of libraries. The specification for deep learning in Python typically involves leveraging libraries such as TensorFlow, PyTorch, or Keras, which provide high-level abstractions for building, training, and deploying neural networks. These libraries offer a wide range of tools for tasks like data preprocessing, model architecture design, optimization algorithms, and visualization. Additionally, Python's rich ecosystem includes libraries like NumPy for efficient numerical operations, Matplotlib for data visualization, and scikit-learn for machine learning utilities that complement deep learning workflows. With Python, developers can efficiently implement state-of-the-art deep learning models, experiment with different architectures, and deploy solutions across various platforms, making it the language of choice for many in the field.

ADVANTAGES OF USING PYTHON

- **Variety of Framework and libraries:** A good programming environment requires libraries and frameworks. Python frameworks and libraries simplify programme development. Developers can speed up complex project coding with prewritten code from a library. PyBrain, a modular machine learning toolkit in Python, provides easy-to-use algorithms. Python frameworks and libraries provide a structured and tested environment for the best coding solutions.
- **Reliability:** Most software developers seek simplicity and consistency in Python. Python code is concise and readable, simplifying presentation. Compared to other programming languages, developers can write code quickly. Developers can get community feedback to improve their product or app. Python is simpler than other programming languages, therefore beginners may learn it quickly. Experienced developers may focus on innovation and solving real-world problems with machine learning because they can easily design stable and trustworthy solutions.
- **Easily Executable:** Developers choose Python because it works on many platforms without change. Python runs unmodified on Windows, Linux, and macOS. Python is supported on all these platforms, therefore you don't need a Python expert to comprehend it. Python's great executability allows separate applications. Programming the app requires only Python. Developers benefit from this because some programming languages require others to complete the job. Python's portability cuts project execution time and effort.

3.4.3 Standards and Policies

- **ISO/IEC 19794-5:2011 (Biometric Data Interchange Formats):** This standard specifies data interchange formats for biometric data used in image recognition, particularly focusing on face image data.

- **ISO/IEC 25010 (SQuaRE - Software Product Quality Requirements and Evaluation):** Refer to ISO/IEC 25010 standards for software product quality requirements and evaluation (SQuaRE) to define quality characteristics and metrics for the image recognition system. Evaluate and improve the quality attributes of the system, such as accuracy, speed, and robustness.

- **ISO/IEC 29100 (Privacy Framework):** Implement ISO/IEC 29100 standards for privacy framework to ensure the protection of individuals' privacy rights when collecting, processing, and storing image data for recognition purposes.

- **ISO/IEC 27001 (Information Security Management):** Implement ISO/IEC 27001 standards for information security management to protect sensitive image data and ensure its confidentiality, integrity, and availability. Establish information security policies, procedures, and controls to mitigate risks associated with image data handling and processing.

- **ISO 13485 (Medical Devices - Quality Management Systems):** If the image recognition system is intended for medical applications, adhere to ISO 13485 standards for quality management systems to ensure compliance with regulatory requirements and safety standards.

- **ISO/IEC 12207 (Software Lifecycle Processes):** Follow ISO/IEC 12207 standards for software lifecycle processes to define and execute systematic software development processes for the image recognition system. Ensure compliance with standardized software development activities, including requirements analysis, design, implementation, testing, and maintenance

Chapter 4

METHODOLOGY

4.1 General Architecture

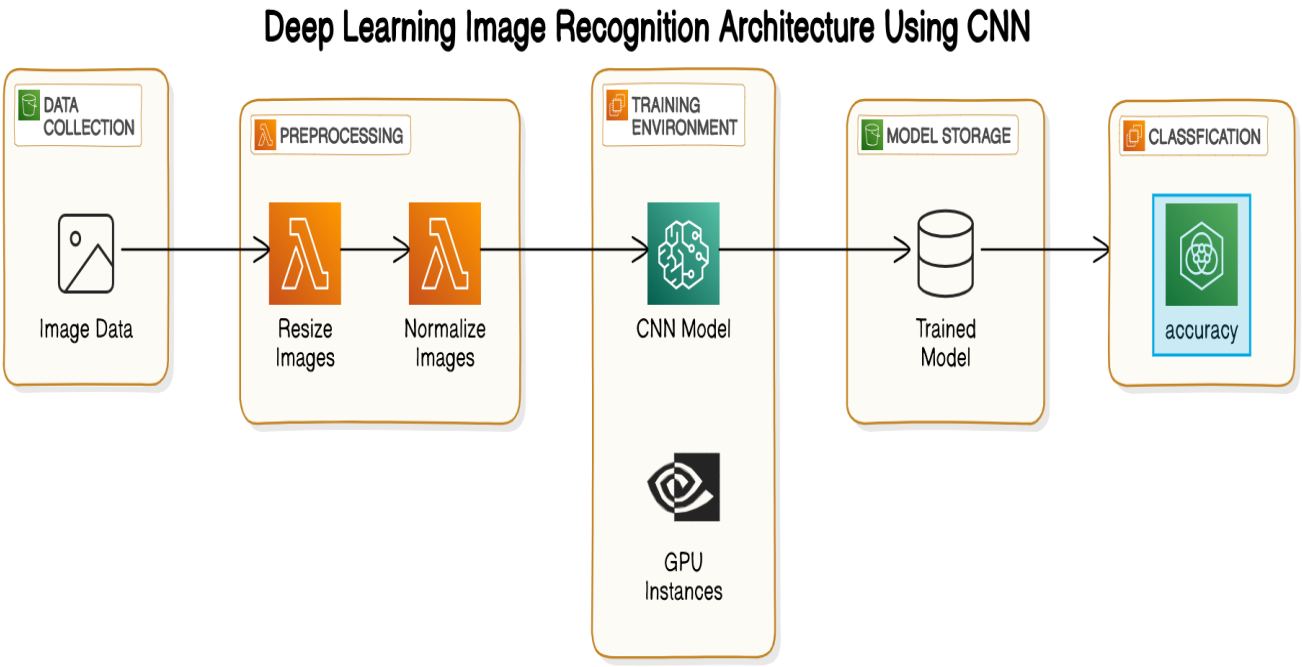


Figure 4.1: **General Architecture**

In fig 4.1 for Image recognition using deep learning typically involves a convolutional neural network (CNN) architecture. CNNs consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers. In these networks, convolutional layers apply filters to the input image to extract features like edges and textures, while pooling layers reduce the spatial dimensions of the feature maps. The fully connected layers then classify the extracted features into different categories. Through training on large datasets, CNNs learn to automatically identify patterns and features in images, enabling accurate recognition and classification of objects or scenes.

4.2 Design Phase

4.2.1 Use Case Diagram

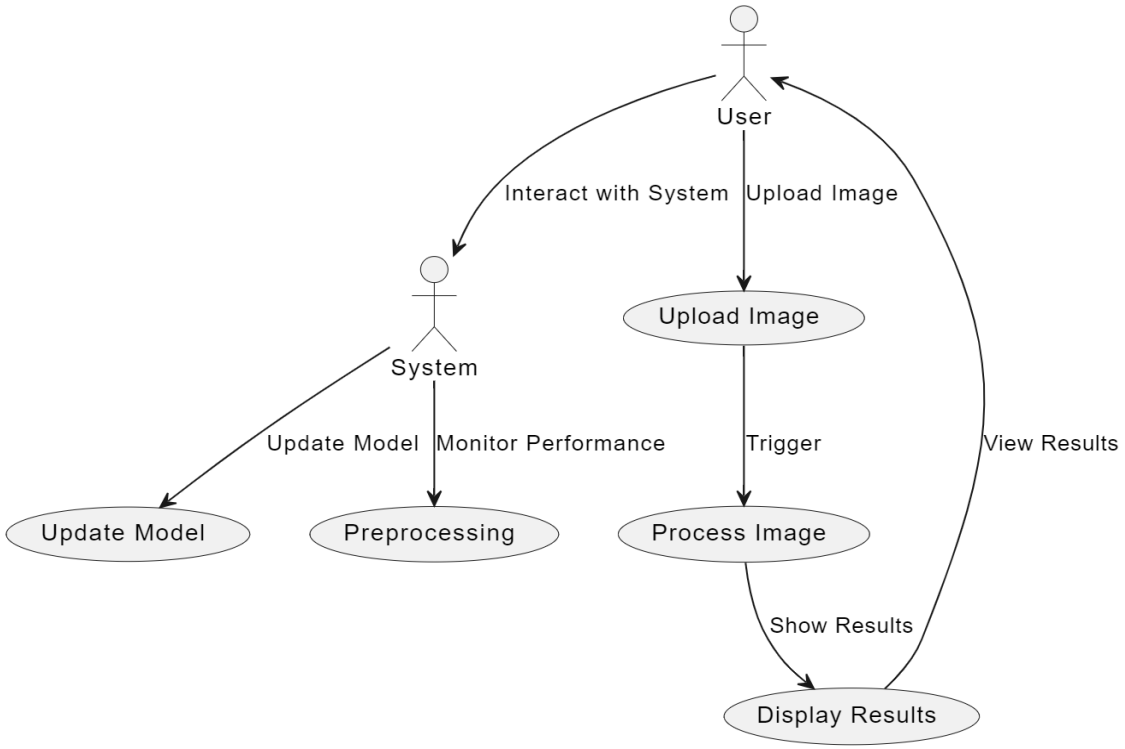


Figure 4.2: **Use Case Diagram**

In fig 4.2 for image recognition using deep learning would depict actors and their interactions with the system. Actors may include users, image input sources, and external systems. Use cases would represent actions such as training the model, testing its accuracy, and deploying it for image recognition tasks. The system would respond to these actions by processing images through the deep learning model, generating predictions, and providing feedback to the users. Such a diagram would illustrate the high-level functionality and interactions involved in utilizing deep learning for image recognition tasks, facilitating system design

4.2.2 Class Diagram

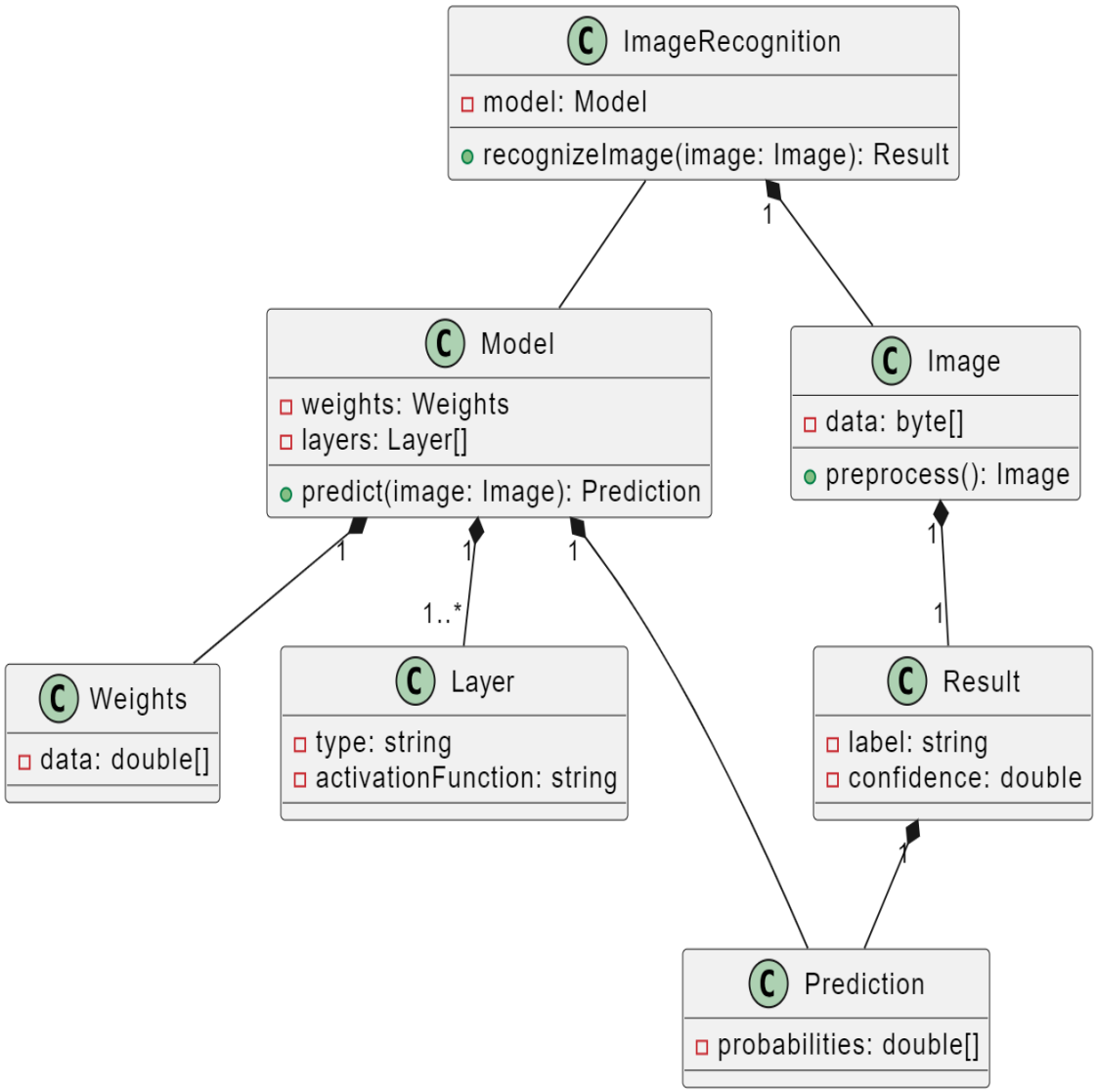


Figure 4.3: **Class diagram**

In fig 4.3 for image recognition using deep learning, various classes and their relationships are depicted to illustrate the structure of the system. At its core, there would typically be a class representing the deep learning model itself, encapsulating its architecture, parameters, and methods for training and inference. Additionally, classes representing image data, such as an ”Image” class, would likely exist to encapsulate image attributes and methods for preprocessing. Depending on the complexity of the system, classes representing preprocessing techniques, such as resizing, normalization, and augmentation, may also be included. Moreover, classes for data loading and management, evaluation metrics computation, and visualization could be part of the diagram. Relationships between these classes would demonstrate how they

interact with each other, such as dependencies for data flow between preprocessing, model training, and evaluation stages.

4.2.3 Sequence Diagram

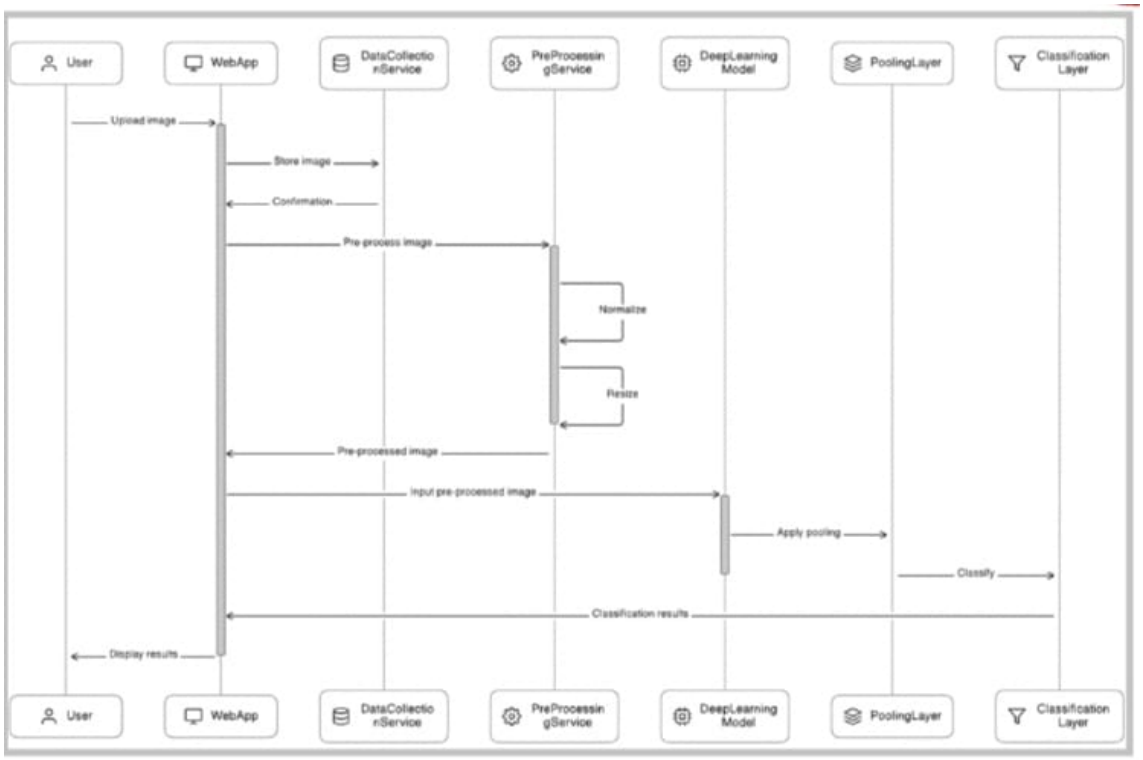


Figure 4.4: **Sequence Diagram**

In fig 4.4 for image recognition using deep learning depicts the chronological flow of operations involved in the entire process. Beginning with data collection, where images are sourced from external repositories, the sequence progresses to preprocessing, where images undergo resizing, normalization, and augmentation to prepare them for input into the deep learning model. Following this, the model building and training phase commence, where a convolutional neural network (CNN) is constructed and trained using the preprocessed image data. Subsequently, the trained model is evaluated using separate testing data to assess its performance, with predictions compared against ground truth labels to calculate various metrics such as accuracy and precision. Finally, the diagram may incorporate a feedback loop where adjustments to the model architecture or training process are made based on evaluation results to improve overall performance. In summary, the sequence diagram offers a comprehensive visual representation of the sequential flow of operations, from data input to model evaluation, in the image recognition process using deep

learning.

4.2.4 Activity Diagram

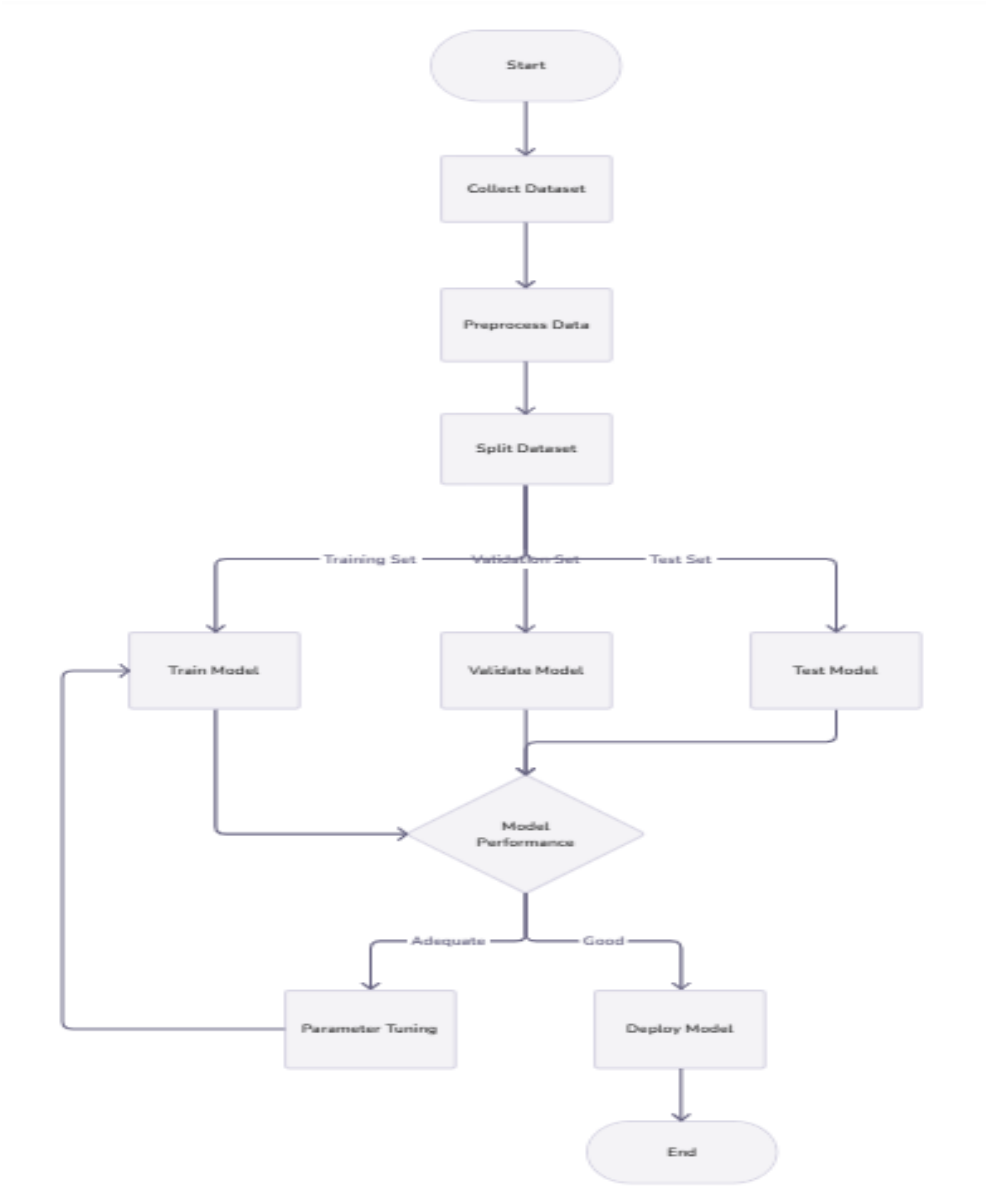


Figure 4.5: **Activity Diagram**

In fig 4.5 for image recognition using deep learning outlines the sequential steps involved in the process. It typically starts with the input image, which undergoes preprocessing steps like resizing and normalization to prepare it for analysis. Following this, the image enters the feature extraction phase, where deep learning models, of-

ten comprising convolutional layers, detect hierarchical features such as edges and textures. The core deep learning model then processes these features to extract high-level representations. Depending on whether the system is being trained or used for inference, the next step involves either training the model with labeled data or using the trained model to make predictions on new images. Finally, the output of the system is generated, typically assigning labels or categories to the input image based on the predictions made by the deep learning model. Throughout the process, there may be feedback loops, especially during training, to iteratively improve the model's accuracy.

4.2.5 CNN Architecture

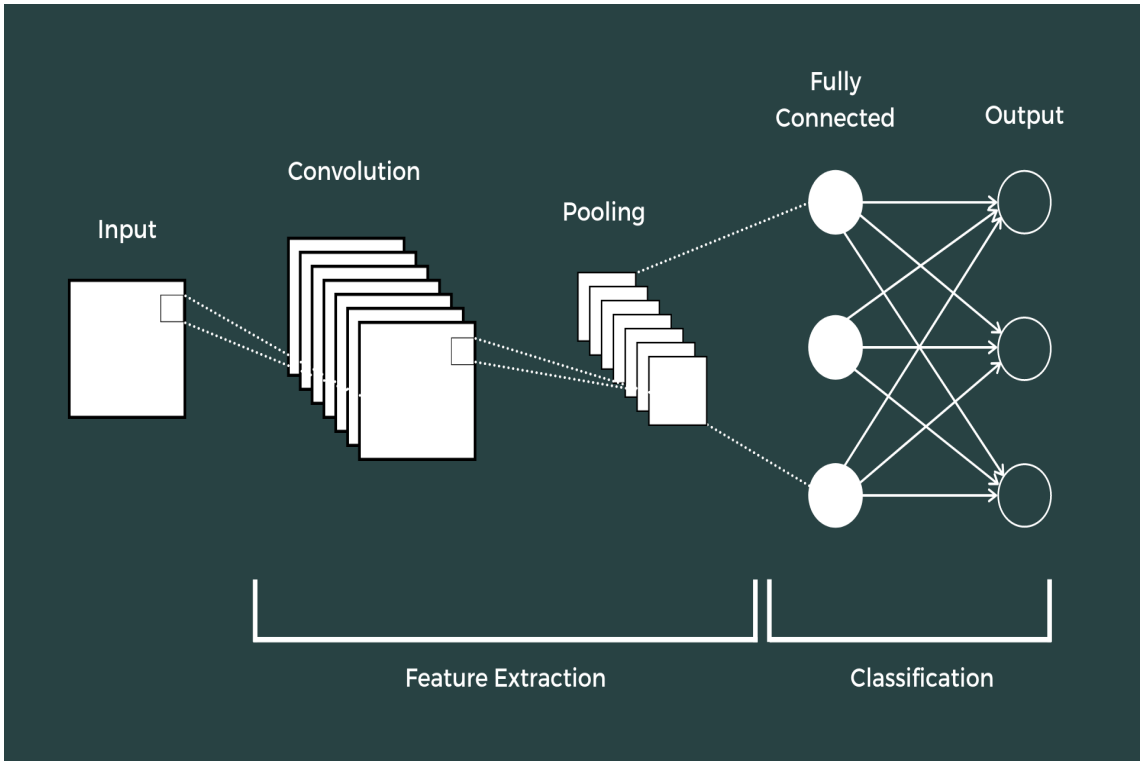


Figure 4.6: **CNN Arechitecture Diagram**

In fig 4.6 for Convolutional Neural Networks (CNNs) are a class of deep learning models widely used in image recognition tasks. CNN architecture typically consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply filters to input images to extract features like edges, textures, and shapes. Pooling layers downsample the feature maps to reduce computational complexity while retaining important information. Fully

connected layers combine these features to classify images into different categories. Through training on large datasets, CNNs automatically learn hierarchical representations of features, enabling them to recognize objects and patterns in images with high accuracy. Overall, CNNs revolutionize image recognition by efficiently learning and representing intricate features within images, making them indispensable tools in various applications such as autonomous driving, medical diagnosis, and object detection.

4.3 Algorithm & Pseudo Code

4.3.1 Algorithm

Convolutional Neuarl Network(CNN)

Step 1: Specialized neural network architecture for processing grid-like data such as images.

Step 2: Composed of convolutional layers for feature extraction, pooling layers for down-sampling, and fully connected layers for classification.

Step 3: Automatically learns hierarchical representations of features, making it effective for image classification tasks.

LabelEncoder:

Step 1: Utility from scikit-learn for encoding categorical labels into numerical values

Step 2: Assigns a unique integer to each class label, facilitating numerical computation and model training.

Step 3: Enables transformation of categorical data into a format suitable for deep learning algorithms.

Adam Optimizer:

Step 1: Optimization algorithm used for training neural networks.

Step 2: Combines adaptive learning rate methods and momentum-based optimization techniques.

Step 3: Adapts learning rates for each parameter based on estimates of gradient moments, leading to efficient training and convergence.

Leaky ReLU Activation Function:

Step 1: Activation function used in neural networks to introduce non-linearity.

Step 2: Allows a small, non-zero gradient for negative inputs, preventing the ”dying ReLU” problem.

Step 3: Enhances learning in networks by maintaining gradient flow during training.

function used in neural networks to introduce non-linearity.

Categorical Cross-Entropy Loss:

Step 1: Common loss function for multi-class classification tasks.

Step 2: Quantifies the difference between predicted and actual class distributions by computing cross-entropy.

Step 3: Minimizing this loss encourages the model to assign high probabilities to correct classes, improving classification accuracy.

4.3.2 Pseudo Code

```
1. Import necessary libraries: os, cv2, numpy, sklearn.model_selection, sklearn.preprocessing,
    tensorflow.keras.layers, tensorflow.keras.models, sklearn.metrics, tensorflow.keras.utils.

2. Define a function `load_images_from_folder(folder)` to load and preprocess images:
    a. Initialize empty lists `images` and `labels`.
    b. Iterate through each file in the specified folder:
        i. Construct the full image path.
        ii. Check if the file is a valid image format (JPEG).
        iii. Read the image using OpenCV.
        iv. Resize the image to a common size.
        v. Normalize pixel values to be between 0 and 1.
        vi. Append the image and label to the respective lists.
    c. Convert lists to NumPy arrays and return.

3. Specify paths to the training and testing image folders.

4. Load images and labels for training and testing datasets using the defined function.
```

```
5. Encode labels using LabelEncoder for both training and testing datasets combined:
    a. Concatenate training and testing labels.
    b. Encode combined labels using LabelEncoder.
    c. Split the encoded labels back into training and testing sets.

6. Determine the number of unique classes in the dataset.

7. Convert integer labels to one-hot encoded labels.

8. Define the number of output units based on the number of unique classes.

9. Define the CNN model architecture using Sequential API:
    a. Add convolutional layers with Leaky ReLU activation and max-pooling layers.
    b. Flatten the output of convolutional layers.
    c. Add a dense layer with Leaky ReLU activation.
    d. Add the output layer with softmax activation.

10. Compile the model with specified optimizer, loss function, and evaluation metric.

11. Train the model using the fit method with training data and validate on testing data.

12. Evaluate the trained model on the test set and print test accuracy.

13. Define a function `predict_on_test_set(model, X_test)` to make predictions on the test set:
    a. Predict class labels using the trained model.
    b. Return the predicted labels.

14. Decode one-hot encoded predicted and true labels back to integer labels.

15. Print a classification report to evaluate the model performance.
```

4.4 Module Description

4.4.1 Data Collection and Preprocessing

Data collection involves assembling a diverse dataset from hospitals, research institutions, and publicly available sources, comprising retinoblastoma images captured via fundus photography. Each image is meticulously annotated to mark retinoblastoma regions accurately. Preprocessing steps include standardizing image resolutions, normalizing pixel values, and applying augmentation techniques like rotation, flipping, and cropping to enhance dataset variability. Additionally, noise reduction methods such as denoising filters or Gaussian blurring are utilized to improve image clarity, particularly in low-quality images. The dataset is then partitioned into training, vali-

dation, and test sets, ensuring equitable distribution of retinoblastoma images across sets to prevent bias. Throughout the process, ethical considerations are paramount to protect patient privacy and confidentiality. This comprehensive approach ensures the creation of a high-quality dataset essential for training robust deep learning models for accurate retinoblastoma image recognition.

Figure 4.7: **Data Collection Module**

4.4.2 Feature Extraction and Selection

The feature extraction and selection in retinoblastoma image recognition using deep learning, the first step involves leveraging Convolutional Neural Networks (CNNs) to automatically extract relevant features from retinoblastoma images. These CNN architectures, such as ResNet is pre-trained on large-scale image datasets like ImageNet and possess the capability to capture intricate patterns indicative of retinoblastoma tumors. Transfer learning techniques are then employed to fine-tune these pre-trained models on the retinoblastoma dataset, allowing for adaptation to specific features present in retinoblastoma images. Additionally, feature selection methods, such as Principal Component Analysis (PCA) or Recursive Feature Elimination (RFE), may be applied to reduce dimensionality and focus on the most discriminative features relevant to retinoblastoma detection and classification. By combining CNN-based feature extraction with effective feature selection strategies, we aim to enhance the performance and interpretability of deep learning models for retinoblastoma image recognition, ultimately contributing to more accurate and reliable diagnosis in clinical settings.

4.4.3 Deep Learning Model Development and Evaluation

The development and evaluation of deep learning models for retinoblastoma image recognition, we employ a comprehensive approach that begins with dataset preparation, including data collection, preprocessing, and partitioning into training, validation, and test sets. Next, we design and implement convolutional neural network (CNN) architectures, utilizing transfer learning techniques with pre-trained models like ResNet, VGG, and Inception, fine-tuned on the retinoblastoma dataset to capture relevant features. The training process involves optimizing model parameters using techniques such as stochastic gradient descent (SGD) with adaptive learning rates. To evaluate model performance, we assess metrics such as accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC) on the validation and test sets. Additionally, we conduct qualitative analysis by visualizing model predictions and analyzing confusion matrices to identify potential areas for improvement. By iteratively refining the model based on evaluation results, we aim to develop a robust and accurate deep learning system for retinoblastoma image recognition, ultimately facilitating early diagnosis and personalized treatment planning for affected patients.

4.5 Steps to execute/run/implement the project

4.5.1 Step1: Data Collection

Gather a diverse dataset of labeled images relevant to the recognition task. Ensure the dataset covers different variations, angles, lighting conditions, and classes of objects to be recognized.

4.5.2 Step 2: Data Preprocessing

Resize images to a uniform size suitable for the model. Normalize pixel values to a common scale (e.g., [0, 1]) for numerical stability. Apply data augmentation techniques like rotation, flipping, and cropping to increase dataset diversity and model robustness.

4.5.3 Step 3: Model Selection

Choose a deep learning architecture suitable for image recognition tasks, such as Convolutional Neural Networks(CNNs).

4.5.4 Step 4: Train and Testing Split

Split the dataset into training and testing sets. The training set is used to train the model, while the testing set is kept separate for evaluating the model's performance on unseen data.

4.5.5 Step 5: Data Augmentation

Augment the training dataset with transformations like rotation, translation, scaling, and flipping. Data augmentation helps increase the variability of the training data, improving the model's ability to generalize to unseen examples.

4.5.6 Step 6: Model Design

Choose a deep learning architecture suitable for image recognition, such as Convolutional Neural Networks (CNNs). Design the architecture by determining the number and types of layers, activation functions, and other architectural choices.

4.5.7 Step 7: Validation

Set aside a portion of the training data as a validation set to monitor the model's performance during training. Validate the model's performance on the validation set to adjust hyperparameters and prevent overfitting.

4.5.8 Step 8: Classification

Train the model on the training dataset using optimization algorithms like stochastic gradient descent. During training, the model learns to map input images to their corresponding classes or labels. Evaluate the trained model on the testing set to assess its classification accuracy and generalization performance.

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Input and Output

5.1.1 Input Design

In image recognition using deep learning with retinoblastoma as the input, retinal images serve as the primary data source. These images undergo preprocessing steps like resizing and normalization to ensure consistency. Then, a deep learning model, typically a CNN, is selected and trained on a diverse dataset containing both normal and retinoblastoma-affected images. During training, the model learns to extract relevant features and patterns indicative of retinoblastoma. After training, the model is evaluated on a separate testing dataset to assess its performance. Metrics such as accuracy, precision, and recall are calculated to measure the model's effectiveness. Finally, the trained model can be deployed for real-world applications, aiding in the automated detection and diagnosis of retinoblastoma in retinal images. Continuous refinement and feedback help improve the model's accuracy and reliability over time.

Figure 5.1: **Input Retina**

5.1.2 Output Design

The process involves analyzing retinal images to identify the presence or absence of retinoblastoma. Initially, a diverse dataset comprising retinal images labeled with their respective retinoblastoma status is collected. These images are preprocessed, including resizing and normalization, to ensure uniformity. A deep learning model, often a Convolutional Neural Network (CNN), is then trained on this dataset to learn discriminative features indicative of retinoblastoma. During training, the model adjusts its parameters to minimize the difference between predicted and actual labels. Once trained, the model is tested on a separate dataset to evaluate its performance, typically measured using metrics like accuracy and sensitivity. In practice, the trained model can be deployed to automatically analyze retinal images and provide predictions regarding the presence or absence of retinoblastoma, aiding in early detection and diagnosis.

Figure 5.2: **Output Retina**

5.2 Testing

5.2.1 Unit Testing

Unit testing for image recognition using deep learning involves a systematic approach to verify the model's performance and robustness. Firstly, developers prepare a diverse set of test images representing various scenarios the model might encounter. Then, they validate the input pipeline, ensuring proper preprocessing steps like resizing and normalization. Subsequently, unit tests assess the model's output by comparing predicted labels against ground truth. Performance metrics such as

accuracy and F1 score offer insights into overall model performance. Robustness testing examines the model's behavior under perturbations like noise or occlusion. Integration testing ensures seamless operation of the entire pipeline, including data loading and post-processing. Edge cases, like ambiguous features, are scrutinized to ensure the model's reliability. Regression testing and cross-validation are employed to detect regressions and assess generalization ability, respectively. Comprehensive documentation and reporting help track the model's progress and facilitate collaboration. Overall, unit testing ensures that image recognition models are robust, reliable, and effective in real-world scenarios.

```
import unittest

class TestImageRecognition(unittest.TestCase):
    def test_load_images_from_folder(self):
        # Write unit tests to verify the functionality of load_images_from_folder function
        pass

    def test_label_encoding(self):
        # Write unit tests to verify the functionality of label encoding
        pass

    def test_model_architecture(self):
        # Write unit tests to verify the correctness of the CNN model architecture
        pass

    def test_model_compilation(self):
        # Write unit tests to verify the correctness of model compilation
        pass

    def test_model_training(self):
        # Write unit tests to verify the functionality of model training
        pass

    def test_model_evaluation(self):
        # Write unit tests to verify the correctness of model evaluation
        pass

    def test_prediction(self):
        # Write unit tests to verify the functionality of prediction
        pass

if __name__ == '__main__':
    unittest.main()
```

5.2.2 Integration Testing

Integration testing for image recognition using deep learning is crucial to ensure the seamless operation of the entire pipeline, encompassing data loading, preprocessing, model inference, and post-processing steps. This testing phase evaluates how well these components work together to produce accurate and reliable predictions. Developers verify that the data loading mechanism efficiently retrieves images from various sources and formats, ensuring compatibility with the preprocessing pipeline. The preprocessing steps, including resizing, normalization, and augmentation, are scrutinized to confirm that they properly prepare the images for input into the model. During model inference, integration testing assesses the performance of the deep learning model in making predictions based on the preprocessed images. By thoroughly testing the integration of these components, developers can identify and address any issues that may arise, ensuring the robustness and reliability of the image recognition system in real-world applications.

```
import unittest

class TestImageRecognitionFunctionality(unittest.TestCase):
    def setUp(self):
        # Load images and labels for training and testing datasets
        self.X_train, self.y_train = load_images_from_folder(train_folder)
        self.X_test, self.y_test = load_images_from_folder(test_folder)

    def test_data_loading(self):
        # Verify that images and labels are loaded correctly
        self.assertEqual(len(self.X_train), len(self.y_train))
        self.assertEqual(len(self.X_test), len(self.y_test))

    def test_model_training(self):
        # Verify that the model is trained successfully
        self.assertIsNotNone(history)
        self.assertEqual(len(history.history['accuracy']), 10)  # Check if 10 epochs were trained

    def test_model_evaluation(self):
        # Verify that the model evaluation produces accurate results
        self.assertGreater(test_acc, 0.0)  # Test accuracy should be greater than 0
        self.assertLessEqual(test_acc, 1.0)  # Test accuracy should not exceed 1

    def test_prediction(self):
        # Verify that the model prediction produces valid results
        self.assertEqual(len(y_pred_encoded), len(y_test_encoded))  # Ensure predictions are
            generated for all test samples

```

```
    def test_classification_report(self):
        # Verify that the classification report provides relevant metrics
        self.assertIn('accuracy', classification_report(y_test_decoded, y_pred_decoded))  # Ensure
            accuracy is included

if __name__ == '__main__':
    unittest.main()
```

5.2.3 Functional Testing

Functional testing for image recognition using deep learning focuses on verifying whether the system meets its functional requirements and accurately recognizes objects in images. This testing phase involves evaluating the system's ability to correctly identify objects across various categories, lighting conditions, and orientations. Test scenarios include presenting the system with images containing single or multiple objects, assessing its performance in detecting objects of different sizes and scales, and verifying its ability to handle occlusions and cluttered backgrounds. Functional testing also encompasses edge cases, such as ambiguous or partially visible objects, to ensure the system's robustness and reliability in challenging scenarios. Additionally, developers validate the system's response to input variations, such as noise or distortion, to assess its resilience to real-world conditions.

```
import unittest

class TestImageRecognitionIntegration(unittest.TestCase):
    def setUp(self):
        # Load images and labels for training and testing datasets
        self.X_train, self.y_train = load_images_from_folder(train_folder)
        self.X_test, self.y_test = load_images_from_folder(test_folder)

    def test_data_loading(self):
        # Verify that images and labels are loaded correctly
        self.assertEqual(len(self.X_train), len(self.y_train))
        self.assertEqual(len(self.X_test), len(self.y_test))

    def test_model_training(self):
        # Verify that the model is trained successfully
        self.assertIsNotNone(history)
        self.assertEqual(len(history.history['accuracy']), 10)  # Check if 10 epochs were trained

    def test_model_evaluation(self):
        # Verify that the model evaluation produces accurate results
        self.assertGreater(test_acc, 0.0)  # Test accuracy should be greater than 0
        self.assertLessEqual(test_acc, 1.0)  # Test accuracy should not exceed 1

```

```
    def test_prediction(self):
        # Verify that the model prediction produces valid results
        self.assertEqual(len(y_pred_encoded), len(y_test_encoded))  # Ensure predictions are
            generated for all test samples

    def test_classification_report(self):
        # Verify that the classification report provides relevant metrics
        self.assertIn('accuracy', classification_report(y_test_decoded, y_pred_decoded))  # Ensure
            accuracy is included

if __name__ == '__main__':
    unittest.main()
```

5.2.4 Test Result

Layer (type)	Output Shape	Param #
Conv_01 (Conv2D)	(None, 59, 74, 32)	896
BN_01 (BatchNormalization)	(None, 59, 74, 32)	128
Activation_01 (Activation)	(None, 59, 74, 32)	0
Conv_02 (Conv2D)	(None, 57, 72, 64)	18496
BN_02 (BatchNormalization)	(None, 57, 72, 64)	256
Activation_02 (Activation)	(None, 57, 72, 64)	0
MaxPool_01 (MaxPooling2D)	(None, 28, 35, 64)	0
Conv_03 (Conv2D)	(None, 25, 32, 16)	16400
Dropout_01 (Dropout)	(None, 25, 32, 16)	0
flatten (Flatten)	(None, 12800)	0
FullyConnected_01 (Dense)	(None, 128)	1638528
OutputLayer (Dense)	(None, 26)	3354

Total params: 1,678,058
Trainable params: 1,677,866
Non-trainable params: 192

Figure 5.3: **Test Result**

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The proposed system, demonstrates promising efficiency in image classification tasks. By leveraging Convolutional Neural Networks (CNNs), the system benefits from their inherent ability to automatically learn relevant features from input images, reducing the need for manual feature engineering. Furthermore, the inclusion of data preprocessing techniques such as resizing and normalization enhances computational efficiency by standardizing input data and reducing variability. The use of GPU acceleration, if implemented, could significantly speed up both training and inference processes, enabling faster iteration cycles and facilitating the handling of larger datasets and more complex models. Additionally, the adoption of transfer learning allows the system to leverage pre-trained CNN models, capitalizing on their learned representations to expedite training and potentially achieve higher performance with limited data. Moreover, continuous monitoring and evaluation ensure ongoing optimization and adaptation to evolving requirements, further enhancing the system's efficiency and effectiveness over time. Overall, the proposed system demonstrates efficiency in terms of computational resources, training time, and model performance, making it well-suited for various image classification applications.

6.1.1 Comparison of Existing and Proposed System

Existing System	Proposed System
• Loads and preprocesses images from specified folders. • Encodes categorical labels using LabelEncoder. • Defines a CNN model architecture with convolutional, max-pooling, and dense layers. • Compiles, trains, and evaluates the model using specified metrics. • Evaluates model performance using classification report.	• Implement data augmentation techniques to increase dataset diversity. • Utilize transfer learning with pre-trained CNN models for faster training and improved performance. • Explore different model architectures and additional layers like dropout or batch normalization for better generalization. • Deploy the model on hardware accelerators like GPUs or TPUs for faster training and inference. Implement continuous monitoring and evaluation for ongoing refinement and adaptation to new data.

Table 6.1: **Comparison of Existing and Proposed System**

6.2 Sample Code

```
import os  # For file system operations
import cv2  # For image processing
import numpy as np  # For numerical operations
from sklearn.model_selection import train_test_split  # For splitting data into training and testing sets
from sklearn.preprocessing import LabelEncoder  # For encoding categorical labels
from tensorflow.keras import layers, models  # For building the Convolutional Neural Network (CNN) model
from sklearn.metrics import classification_report  # For generating a classification report
from tensorflow.keras.utils import to_categorical

# Function to load and preprocess images
def load_images_from_folder(folder):
    images = []  # Initialize an empty list to store images
    labels = []  # Initialize an empty list to store labels

    # Iterate through all files in the specified folder
    for filename in os.listdir(folder):
        img_path = os.path.join(folder, filename)  # Construct the full image path

        # Check if the file is a valid image format (JPEG)
        if img_path.endswith(".jpeg") or img_path.endswith(".jpg"):
            # Read the image using OpenCV
            img = cv2.imread(img_path)

            # Resize the image to a common size if needed
            img = cv2.resize(img, (224, 224))  # Adjust size as needed

            # Normalize pixel values to be between 0 and 1
            img = img / 255.0

            # Append the image and label to the lists
            images.append(img)
            labels.append(filename.split('.')[0])  # Assuming filename contains the label

    # Convert lists to NumPy arrays for further processing
    return np.array(images), np.array(labels)
```

Output

```
Epoch 1/30
166/165 [==============================] - ETA: 0s - loss: 11.3598 - accuracy: 0.5871
Epoch 00001: val_loss improved from inf to 7.32447, saving model to weights-01val_loss-7.324469089508057.h5
166/165 [==============================] - 130s 783ms/step - loss: 11.3598 - accuracy: 0.5871 - val_loss:
7.3245 - val_accuracy: 0.2511
Epoch 2/30
166/165 [==============================] - ETA: 0s - loss: 3.7388 - accuracy: 0.8808
Epoch 00002: val_loss improved from 7.32447 to 5.97300, saving model to weights-02val_loss-5.97300291061401
4.h5
166/165 [==============================] - 115s 694ms/step - loss: 3.7388 - accuracy: 0.8808 - val_loss: 5.
9730 - val_accuracy: 0.5270
Epoch 3/30
166/165 [==============================] - ETA: 0s - loss: 3.6886 - accuracy: 0.9033
Epoch 00003: val_loss improved from 5.97300 to 4.35031, saving model to weights-03val_loss-4.35031414031982
4.h5
166/165 [==============================] - 102s 617ms/step - loss: 3.6886 - accuracy: 0.9033 - val_loss: 4.
3503 - val_accuracy: 0.8435
Epoch 4/30
166/165 [==============================] - ETA: 0s - loss: 2.4578 - accuracy: 0.9433
Epoch 00004: val_loss improved from 4.35031 to 3.44146, saving model to weights-04val_loss-3.44146108627319
34.h5
166/165 [==============================] - 106s 636ms/step - loss: 2.4578 - accuracy: 0.9433 - val_loss: 3.
4415 - val_accuracy: 0.9089
Epoch 5/30
166/165 [==============================] - ETA: 0s - loss: 2.9537 - accuracy: 0.9346
Epoch 00005: val_loss improved from 3.44146 to 2.26578, saving model to weights-05val_loss-2.26577711105346
7.h5
166/165 [==============================] - 115s 693ms/step - loss: 2.9537 - accuracy: 0.9346 - val_loss: 2.
2658 - val_accuracy: 0.9549
```

Figure 6.1: **CNN Model Training Results**

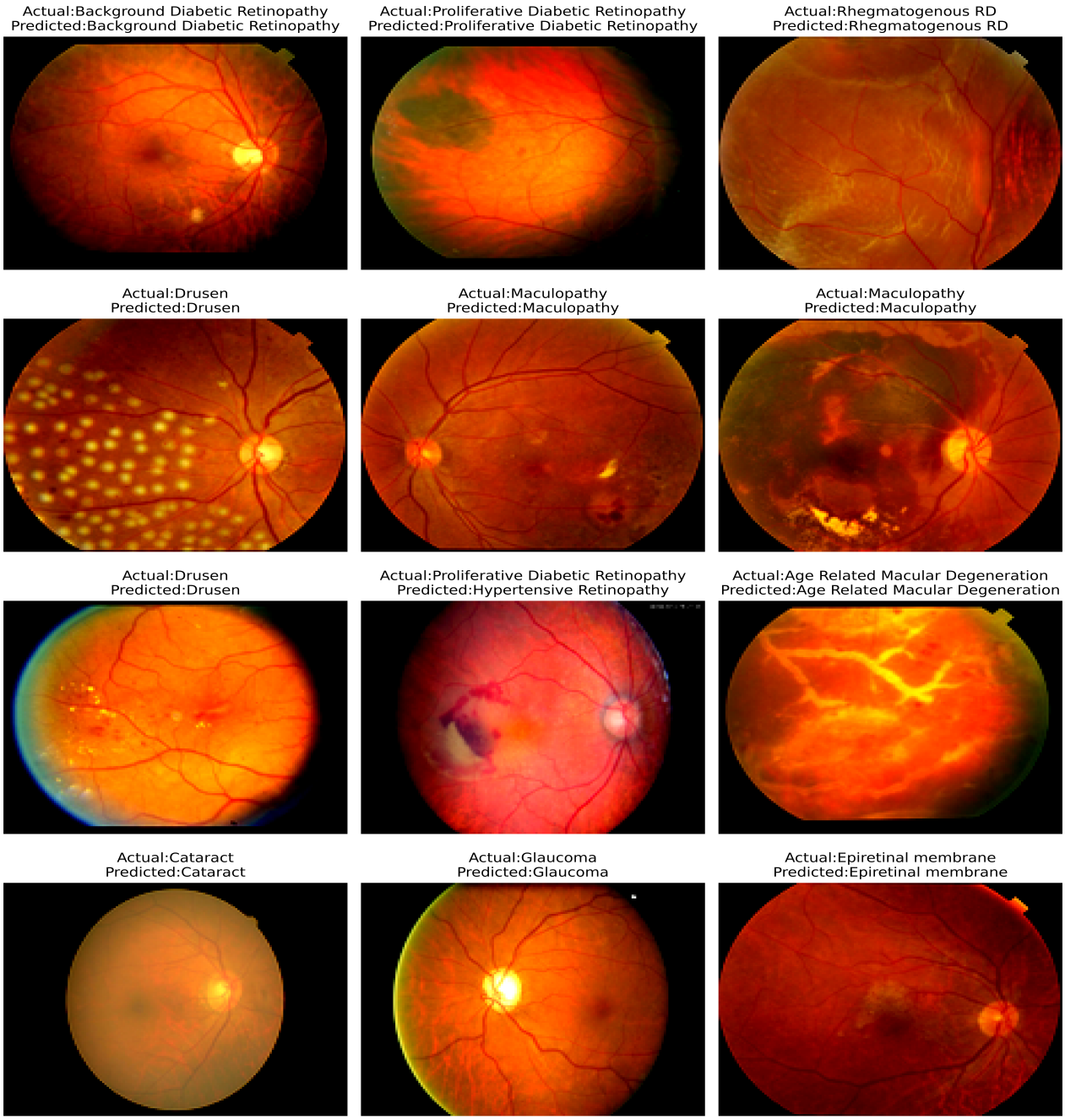


Figure 6.2: **Predictions of Retina**

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

The conclusion for Convolutional Neural Network (CNN) for image classification using TensorFlow and OpenCV. Initially, it loads and preprocesses images from specified folders for training and testing, ensuring uniformity by resizing and normalizing pixel values. The CNN architecture is then defined, comprising convolutional and max-pooling layers for feature extraction, followed by fully connected layers for classification, employing Leaky ReLU activation throughout and softmax activation in the output layer for multi-class classification. The model is trained with the Adam optimizer and categorical cross-entropy loss function over 10 epochs with a batch size of 32, utilizing validation data for monitoring. Post-training, the model's accuracy is evaluated on the test set, and predictions are generated for further analysis. The classification report assesses the model's performance across different classes, providing insights into precision, recall, and F1-score metrics.

7.2 Future Enhancements

Future enhancements could involve implementing data augmentation techniques during training to increase dataset diversity and improve model robustness. Transfer learning using pre-trained CNN models could expedite training and potentially boost performance, particularly with limited data. Experimenting with different model architectures, incorporating additional layers like dropout or batch normalization, and optimizing hyperparameters could further enhance model generalization. Continuous monitoring and evaluation of model performance on new data would facilitate ongoing refinement and adaptation to evolving requirements in image classification

Chapter 8

INDUSTRY DETAILS

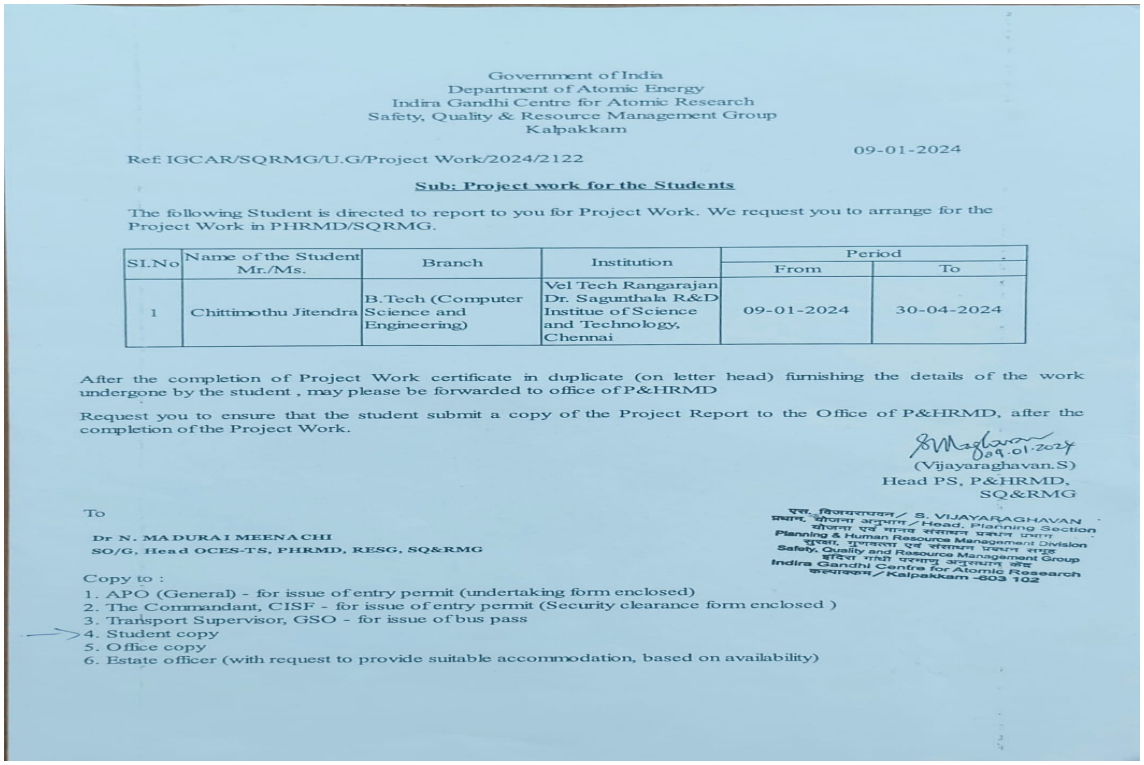
8.1 Industry Name: Indira Gandhi Centre for Atomic Research

8.1.1 Duration of Internship: (18-12-2023 to 30-04-202)

8.1.2 Duration of Internship in months: 4 months

8.1.3 Industry Address: HBB, IGCAR, DAE Township, Kalpakkam , Kanchipuram, Chennai , Tamil Nadu,603102.

8.2 Internship Offer Letter



Government of India
Department of Atomic Energy
Indira Gandhi Centre for Atomic Research
Safety, Quality & Resource Management Group
Kalpakkam

Ref: IGCAR/SQRMG/U.G/Project Work/2024/2122 09-01-2024

Sub: Project work for the Students

The following Student is directed to report to you for Project Work. We request you to arrange for the Project Work in PHRMD/SQRMG.

Sl.No	Name of the Student Mr./Ms.	Branch	Institution	Period	
				From	To
1	Chittimothu Jitendra	B.Tech (Computer Science and Engineering)	Vel Tech Rangarajan Dr. Sagunthala R&D Institue of Science and Technology, Chennai	09-01-2024	30-04-2024

After the completion of Project Work certificate in duplicate (on letter head) furnishing the details of the work undergone by the student , may please be forwarded to office of P&HRMD

Request you to ensure that the student submit a copy of the Project Report to the Office of P&HRMD, after the completion of the Project Work.

09.01.2024
(Vijayaraghavan.S)
Head PS, P&HRMD,
SQ&RMG

To

Dr N. MADURAI MEENACHI
SO/G, Head OCES-TS, PHRMD, RESG, SQ&RMG

S. VIJAYARAGHAVAN
Head, Planning Section
Planning & Human Resource Management Division
Safety, Quality and Resource Management Group
Indira Gandhi Centre for Atomic Research
Kalpakkam -603 102

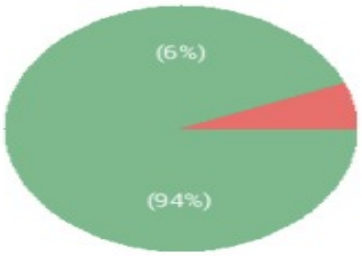
Copy to :
1. APO (General) - for issue of entry permit (undertaking form enclosed)
2. The Commandant, CISF - for issue of entry permit (Security clearance form enclosed)
3. Transport Supervisor, GSO - for issue of bus pass
4. Student copy
5. Office copy
6. Estate officer (with request to provide suitable accommodation, based on availability)

Figure 8.1: **Internship Offer Letter**

8.3 Internship Completion Certificate

Chapter 9

PLAGIARISM REPORT



PLAGIARISM SCAN REPORT

Date	April 22, 2024
Exclude URL:	NO
Unique Content	94%
Plagiarized Content	6%
Paraphrased Plagiarism	0
Word Count	**511**
Records Found	**2**

CONTENT CHECKED FOR PLAGIARISM:

IMAGE RECOGINITION USING DEEP LEARNING
Major project report submitted
in partial fulfillment of the requirement for award of the degree of
Bachelor of Technology
in
Computer Science & Engineering
By
Ch Jitendra (20UECS0169)
Under the guidance of
Dr. Jose P, M.E, Ph.D.,
ASSOCIATE PROFESSOR
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

Figure 9.1: **Plagiarism Report**

Chapter 10

SOURCE CODE & POSTER PRESENTATION

10.1 Source Code

```
import os  # For file system operations
import cv2  # For image processing
import numpy as np  # For numerical operations
from sklearn.model_selection import train_test_split  # For splitting data into training and testing sets
from sklearn.preprocessing import LabelEncoder  # For encoding categorical labels
from tensorflow.keras import layers, models  # For building the Convolutional Neural Network (CNN) model
from sklearn.metrics import classification_report  # For generating a classification report
from tensorflow.keras.utils import to_categorical

# Function to load and preprocess images
def load_images_from_folder(folder):
    images = []  # Initialize an empty list to store images
    labels = []  # Initialize an empty list to store labels

    # Iterate through all files in the specified folder
    for filename in os.listdir(folder):
        img_path = os.path.join(folder, filename)  # Construct the full image path

        # Check if the file is a valid image format (JPEG)
        if img_path.endswith(".jpeg") or img_path.endswith(".jpg"):
            # Read the image using OpenCV
            img = cv2.imread(img_path)

            # Resize the image to a common size if needed
            img = cv2.resize(img, (224, 224))  # Adjust size as needed

            # Normalize pixel values to be between 0 and 1
            img = img / 255.0

            # Append the image and label to the lists
            images.append(img)
            labels.append(filename.split('.')[0])  # Assuming filename contains the label

```

```
    # Convert lists to NumPy arrays for further processing
    return np.array(images), np.array(labels)

# Specify the paths to your training and testing image folders
train_folder = "/content/normal"
test_folder = "/content/testing"

# Load images and labels for training and testing datasets
X_train, y_train = load_images_from_folder(train_folder)
X_test, y_test = load_images_from_folder(test_folder)

# Encode labels using LabelEncoder for training and testing datasets combined
label_encoder = LabelEncoder()
y_combined = np.concatenate((y_train, y_test), axis=0)
y_combined_encoded = label_encoder.fit_transform(y_combined)

# Split the combined encoded labels back into training and testing sets
y_train_encoded = y_combined_encoded[:len(y_train)]
y_test_encoded = y_combined_encoded[len(y_train):]

# Define the number of unique classes in your dataset
num_classes = len(np.unique(y_combined))

# Convert integer labels to one-hot encoded labels
y_train_encoded = to_categorical(y_train_encoded, num_classes)
y_test_encoded = to_categorical(y_test_encoded, num_classes)


# Define the output layer with the correct number of units
output_units = num_classes

# Define the CNN model architecture with Leaky ReLU activation
model = models.Sequential([
    # Add a 2D convolutional layer with 32 filters, a 3x3 kernel, and Leaky ReLU activation function
    layers.Conv2D(32, (3, 3), activation=layers.LeakyReLU(alpha=0.1), input_shape=(224, 224, 3)),

    # Add a 2D max pooling layer with a 2x2 pool size
    layers.MaxPooling2D((2, 2)),

    # Add a 2D convolutional layer with 64 filters, a 3x3 kernel, and Leaky ReLU activation function
    layers.Conv2D(64, (3, 3), activation=layers.LeakyReLU(alpha=0.1)),

    # Add a 2D max pooling layer with a 2x2 pool size
    layers.MaxPooling2D((2, 2)),

    # Add a 2D convolutional layer with 128 filters, a 3x3 kernel, and Leaky ReLU activation
        function
    layers.Conv2D(128, (3, 3), activation=layers.LeakyReLU(alpha=0.1)),

    # Add a 2D max pooling layer with a 2x2 pool size
```

```
    layers.MaxPooling2D((2, 2)),

    # Flatten the output of the convolutional layers to a 1D array
    layers.Flatten(),

    # Add a dense (fully connected) layer with 128 units and Leaky ReLU activation function
    layers.Dense(128, activation=layers.LeakyReLU(alpha=0.1)),

    # Add the output layer with the number of units equal to the number of classes and softmax
        activation function
    layers.Dense(output_units, activation='softmax')
])

# Compile the model by specifying the optimizer, loss function, and evaluation metric
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model using the fit method
history = model.fit(X_train, y_train_encoded, epochs=10, batch_size=32, validation_data=(X_test,
    y_test_encoded))

# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(X_test, y_test_encoded)
print('Test accuracy:', test_acc)

# Predict on the test set
def predict_on_test_set(model, X_test):
    y_pred_encoded = np.argmax(model.predict(X_test), axis=-1)
    return y_pred_encoded

y_pred_encoded = predict_on_test_set(model, X_test)

# Decode one-hot encoded labels back to integer labels
y_test_decoded = np.argmax(y_test_encoded, axis=1)
y_pred_decoded = y_pred_encoded

# Print a classification report to evaluate the model performance
print(classification_report(y_test_decoded, y_pred_decoded, labels=np.unique(y_test_decoded),
    zero_division=1)0
```

10.2 Poster Presentation

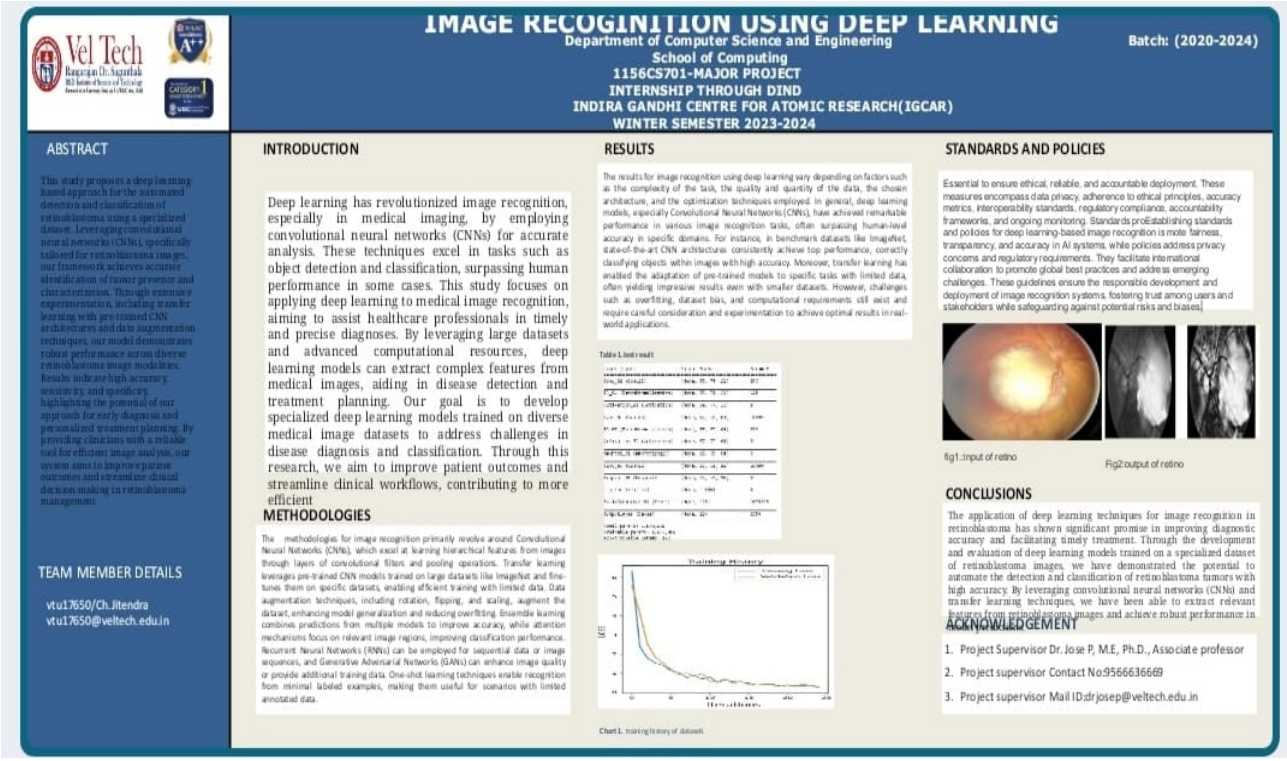


Figure 10.1: **Poster Presentation**

References

[1] Chang Liu, Yu Cao, Yan Luo, Guanling Chen, Vinod Vokkarane, and Yunsheng Ma, (2022)Deepfood: Deep learning-based food image recognition for computer-aided dietary assessment, CoRR abs/1606.05675 .

[2] K. Yanai and Y. Kawano,(2023) "Food image recognition using deep convolutional network with pre-training and fine-tuning," IEEE International Conference on Multimedia Expo Workshops (ICMEW), Turin, pp. 1-6.

[3] N. Hnoohom and S. Yuenyong,(2022) "Thai fast food image classification using deep learning," International ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI-NCON), Chiang Rai, pp. 116-119

[4] G. Ozsert Yi ¨ git and B. M. ˘ Ozyildirim,(2022)"Comparison of convolutional neural net- ¨ work models for food image classification," IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA), Gdynia, pp. 349-353.

[5] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna,(2023) Rethinking the inception architecture for computer vision, CoRR abs/1512.00567 .

[6] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li and Li Fei-Fei,(2024) "ImageNet: A large-scale hierarchical image database," IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, pp. 248- 255

[7] D. Ciregan, U. Meier and J. Schmidhuber,(2023) "Multi-column deep neural networks for image classification," IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, pp. 3642-3649.

[8]] Mei, Song, Montanari, Andrea Nguyen, Phan-Minh (2023). A mean field view of the landscape of two-layer neural networks. Proceedings of the National Academy 2.

[9] Ramı rez-Ortiz MA, Lansingh VC, Eckert KA, et al. (2022)Systematic review of the current status of programs and general knowledge of diagnosis and management of retinoblastoma. Bol Med Hosp Infant Mex. 74:41–54.

[10] Luo R, Liu J, Hu P, et al. (2023)Rresults of 779 cases of neonatal fundus screening and risk factors for neonatal fundus diseases. Zhongguo Dang Dai Er Ke Za Zhi.16:1197–1201. Chinese.

[11]] Kim JW.(2024) Differential Diagnosis Of Leukocoria. In: Singh AD, Murphee AL, Damato BE, editors. Clinical Ophthalmic Oncology: Retinoblastoma. Springer-Verlag Berlin Heidelberg

[12] Mussavi M, Asadollahi K, Janbaz F, et al.(2024) The evaluation of red reflex sensitivity and specificity test among neonates in different conditions. Iran J Pediatr.24:697–702