**SRI RAMAKRISHNA ENGINEERING COLLEGE BONAFIDE CERTIFICATE**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**MINI PROJECT I – APRIL 2024**

This is to certify that the project entitled

**TOMATO PLANT STAGE DETECTION**

is the bonafide record of Mini Project I done by

**ARJUN SUDHEER (2201021)**

**DHIVYASHREE M.P (2201042)**

**DULAL ROY (2201044)**

of B.E. Computer Science and Engineering during the year 2023-2024.

who carried out the Mini Project I under my supervision, certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr.A.Grace Selvarani, Ph.D., Mrs. M. SHANTHINI,

**HEAD OF THE DEPARTMENT PROJECT GUIDE**

Professor, Assistant Professor (Sr.Gr.), Computer Science and Engineering, Computer Science and Engineering, Sri Ramakrishna Engineering College, Sri Ramakrishna Engineering College, Coimbatore-641022. Coimbatore-641022.

**Submitted for the Project Viva-Voce Examination held on**

**Internal Examiner External Examiner**

**DECLARATION**

We affirm that the Mini Project I titled **“TOMATO PLANT STAGE DETECTION”** being submitted in partial fulfillment for the award of Bachelor of Engineering is the original work carried out by us. It has not formed the part of any other project work submitted for award of any degree or diploma, either in this or any other University.

-

(Signature of the Candidates)

|  |  |
| --- | --- |
| **ARJUN SUDHEER** | **(2201021)** |
| **DHIVYASHREE M.P** | **(2201042)** |
| **DULAL ROY** | **(2201044)** |

I certify that the declaration made above by the candidates is true.

(Signature of the guide)

**Mrs. M. SHANTHINI,**

**Assistant Professor(Sr.Grade),**

**Department of CSE**

**ACKNOWLEDGEMENT**

We express our gratitude to **Sri.D. LAKSHMINARAYANASWAMY,** Managing Trustee, **Sri. R. SUNDAR,** Joint Managing Trustee, SNR Sons Charitable Trust, Coimbatore for providing excellent facilities to carry out our project.

We express our deepest gratitude to **Dr. N. R. ALAMELU,** Principal, for her valuable guidance and blessings.

We thank **Dr. A. GRACE SELVARANI,** Professor and Head, Department of Computer Science and Engineering who modeled us both technically and morally for achieving great success in life.

We sincerely thank our Project Coordinator, **Dr. R. ANURADHA,** Associate Professor, Department of Computer Science and Engineering for her great inspiration.

Words are inadequate to offer thanks to our respected guide. We wish to express our sincere thanks to **Mrs. M. SHANTHINI,** Department of Computer Science and Engineering, who gives constant encouragement and support throughout this project work and who makes this project a successful one.

We also thank all the staff members and technicians of our Department for their help in making this project a successful one.

**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | | | **PAGE NO.** |
|  | **ABSTRACT** | | | **6** |
|  | **LIST OF FIGURES** | | | **7** |
|  | **LIST OF TABLES** | | | **8** |
|  | **LIST OF ABBREVIATIONS** | | | **9** |
| **1** | **INTRODUCTION** | | | **10** |
|  | 1.1 | Tomato Plant Stages | | 11 |
|  | 1.2 | Tomato Plant Stage Prediction | | 12 |
|  | 1.3 | Deep Learning Models | | 13 |
| **2** | **LITERATURE REVIEW** | | | **14** |
| **3** | **SYSTEM ANALYSIS** | | | **16** |
|  | 3.1 | Existing System | | 16 |
|  | 3.2 | Proposed System | | 17 |
| **4** | **SYSTEM SPECIFICATION** | | | **18** |
|  | 4.1 | Hardware Requirements | | 18 |
|  | 4.2 | Software Requirements | | 18 |
| **5** | **SOFTWARE DESCRIPTION** | | | **19** |
|  | 5.1 | Python | | 19 |
| **6** | **PROJECT DESCRIPTION** | | | **22** |
|  | 6.1 | Problem Definition | | 22 |
|  | 6.2 | Introduction to Proposed System | | 22 |
|  |  | 6.2.1 System Architecture | | 23 |
|  | 6.3 | Module Description | | 24 |
|  |  | 6.3.1 | Module 1- Loading the Data | 24 |
|  |  | 6.3.2 | Module 2- Preprocess the Data | 25 |
|  |  | 6.3.3 | Module 3-Deep Learning Models | 26 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **7** | **SYSTEM IMPLEMENTATION** | | | **30** |
|  | 7.1 | Introduction | | 30 |
|  | 7.2 | Implementation Steps | | 30 |
|  | 7.3 | Model Flow | | 31 |
|  |  | 7.3.1 | Custom CNN | 31 |
|  |  | 7.3.2 | ResNet50 | 33 |
|  |  | 7.3.3 | VGG16 | 35 |
|  |  | 7.3.4 | InceptionV3 | 36 |
|  |  | 7.3.5 | MobileNetV2 | 37 |
|  |  | 7.3.6 | EfficientNetB0 | 39 |
|  |  | 7.3.7 | YoloV3 | 40 |
| **8** | **RESULTS AND DISCUSSION** | | | **42** |
|  | 8.1 | Metrics Performance of each model | | 42 |
|  | 8.2 | Comparison of Results | | 57 |
| **9** | **CONCLUSION AND FUTURE ENHANCEMENTS** | | | **58** |
|  | 9.1 | Conclusion | | 58 |
|  | 9.2 | Future Enhancements | | 58 |
|  | **APPENDIX** | | |  |
|  |  | Snapshots of Dataset | | 59 |
|  |  | Snapshots of Source Code of all the 7 Models | | 64 |
| **10** | **REFERENCES** | | | **72** |

**ABSTRACT**

Tomato cultivation is of significant agricultural importance worldwide. Monitoring the growth stages of tomato plants is crucial for optimizing cultivation practices and ensuring a healthy yield. In this study, we propose a novel Tomato Plant Stages Detection System utilizing deep learning algorithms, including Custom CNN, ResNet50, VGG16, Inception, EfficientNet, and MobileNetV2. We collected a specialized dataset comprising images of tomato plants at two distinct growth stages from the fields of the Tamil Nadu Agricultural University (TNAU). The dataset encompasses various environmental conditions and growth variations typical of field settings. Our proposed system aims to automate the identification of tomato plant stages, enabling farmers to make informed decisions regarding crop management.

**LIST OF FIGURES**

**NAME OF THE FIGURE PAGE NO**

6. 1 System Architecture 23

6. 2 Summary of custom CNN 26

6. 3 Summary of RESNET50 27

6. 4 Summary of VGG16 27

6. 5 Summary of InceptionV3 28

6. 6 Summary of MobileNetV2 28

6. 7 Summary of EfficientNetB0 29

1. 8 Summary of YOLOV3 29
2. 1 Model flow 31

7. 2 Overview of Custom CNN 33

7. 3 Overview of RESNET50 34

7. 4 Overview of Inceptionv3 35

7. 5 Overview of VGG16 37

7. 6 Overview of MobileNetV2 38

7. 7 Overview of EfficientNetB0 40

7. 8 Overview of YOLOV3 41

* 1. Graph of Train and Test Accuracy Over Epoch 42
  2. Confusion Matrix for Custom CNN 42
  3. ROC-AUC Curve for Custom CNN 43
  4. Residual Plot for Custom CNN 43
  5. Test and Train Accuracy plot for ResNet50 44
  6. Confusion Matrix for ResNet50 45
  7. ROC-AUC Curve for ResNet50 45
  8. Residual Plot for ResNet50 46
  9. Test and Train Accuracy plot for VGG16 46
  10. Confusion Matrix for VGG16 47
  11. ROC-AUC Curve for VGG16 48
  12. Residual Plot for VGG16 48
  13. Test and Train Accuracy plot for InceptionV3 48
  14. Confusion Matrix for VGG16 49
  15. ROC-AUC Curve for VGG16 49
  16. Residual Plot for VGG16 50
  17. Test and Train Accuracy plot for EfficientNetB0 50
  18. Confusion Matrix for EfficientNetB0 51
  19. ROC-AUC Curve for EfficientNetB0 51
  20. Residual Plot for EfficientNetB0 52
  21. Test and Train Accuracy plot for MobileNetV2 52
  22. Confusion Matrix for MobileNetV2 53
  23. ROC-AUC Curve for MobileNetV2 53
  24. Residual Plot for MobileNetV2 54
  25. Test and Train Accuracy plot for YoloV3 54
  26. Confusion Matrix for YoloV3 55
  27. ROC-AUC Curve for YoloV3 56
  28. Residual Plot for YoloV3 56
  29. Other kind of Metrics Comparison for 7 Models 57

**LIST OF ABBREVIATION**

|  |  |
| --- | --- |
| **ABBREVIATION** | **EXPANSION** |
| CNN | Convolutional Neural Network |
| ResNet | Residual Network |
| VGG | Visual Geometry Group |
| YOLO | You Only Look Once |

**CHAPTER 1 INTRODUCTION**

Traditional healthcare, especially homeopathy, faces challenges in appointment scheduling and patient query resolution. Patients struggle to book appointments, and doctors have limited tools to manage schedules efficiently. Moreover, patients frequently have queries regarding treatments, symptoms, and medication, which existing chatbots fail to address effectively. The lack of a structured, automated system leads to inefficiencies in patient care and administration.

The healthcare industry faces several inefficiencies in patient interaction, appointment management, information accessibility, and technological communication. Limited digital patient engagement platforms, complex manual communication processes, and minimal technological integration hinder seamless interaction between patients and healthcare providers. Appointment management is often time-consuming due to a lack of real-time booking capabilities and inefficient communication channels between doctors and patients. Additionally, patients struggle with accessing comprehensive medical information, receiving personalized medical guidance, and resolving queries intelligently. The absence of AI-driven interaction platforms further contributes to these challenges, as minimal personalized patient support and context-aware medical responses limit the efficiency of healthcare communication.

To address these issues, this project aims to develop an intelligent chatbot system that automates patient appointment booking. Additionally, it will integrate with MongoDB Atlas to ensure real-time database management and efficient data retrieval.

### CHAPTER 2

**LITERATURE SURVEY**

[1] The usage of chatbots in healthcare to help patients get medical information is examined by Clark & Bailey (2024). The study emphasises how AI-powered chatbots provide a practical and effective answer to patient questions while increasing engagement, streamlining communication, and enhancing healthcare accessible.

[2] In their analysis of chatbots' use in healthcare, Idrees et al. (2024) highlight how well they can communicate with patients, evaluate symptoms, and provide medical advice. According to the report, developments in AI-powered chatbot technology have the potential to increase patient engagement, automate healthcare support, and improve accessibility.

[3] Retrieval-Augmented Generation (RAG)-based LLMs for medical chatbot applications are systematically analysed by Bora & Cuayahuitl (2024), who assess the retrieval effectiveness, response precision, and contextual relevance of these systems. The study emphasises how RAG might improve chatbot performance in healthcare support and decision-making, as well as the incorporation of medical information.

[4] The use of Large-Scale Language Models (LLMs) in healthcare chatbots to deliver real-time medical information is investigated by K.A. et al. (2024). According to the report, LLMs have the ability to completely transform digital healthcare support by improving response accuracy, user interactivity, and accessibility.

[5] Manesh (2024) investigates how Mistral 7B may be improved for medical question-answering (QA), highlighting how effective it is at improving precision and contextual awareness. The study illustrates methods for maximising LLM performance in medical applications, showcasing its capacity to deliver accurate and dependable medical answers.

[6] In healthcare applications, modular architectures—especially those created using Python—are well known for being scalable and maintainable. According to research by Mhatre et al. (2024), modular design helps LLM-based AI chatbots integrate advanced features well while preserving system resilience.

[7] Large Language Models (LLMs) are used by Mhatre et al. (2024) to investigate the deployment of an advanced AI chatbot for healthcare. The study looks at the chatbot's capacity to respond to medical questions with precision and context awareness, emphasising how it might improve patient accessibility and engagement. The study shows how well LLM-driven chatbots give trustworthy healthcare information through performance evaluations.

[8] In order to improve semantic search, retrieval, and response generation, Shingde (2024) investigates how combining QdrantDB, Mistral 8x7B MoE, LangChain, and Streamlit improves RAG applications. The study emphasises how AI-driven information retrieval systems can be optimised through the use of vector databases, MoE models, and workflow automation.

[9] The integration of LangChain, Groq, Llama3, and Qdrant to improve Retrieval-Augmented Generation (RAG) applications is investigated by Seth (2024). According to the study, enhanced LLMs, vector databases, and optimised inference engines enhance the scalability, response accuracy, and retrieval efficiency of AI-driven knowledge systems.

[10] MedDoc-Bot, a chat-based tool for comparing Large Language Models (LLMs) in the context of paediatric hypertension guidelines, is presented by Jabarulla et al. (2024). The study assesses how well various LLMs perform in medical decision support, emphasising their precision, dependability, and suitability for use in clinical settings.

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

In the current healthcare system, especially within the realm of homeopathy, there exists a considerable gap in technology integration, particularly in patient communication and appointment management. Traditional methods, such as manual appointment scheduling, are still widely practiced. This often leads to inefficiencies, including delayed appointments, overlapping bookings, and lack of real-time updates. Patients face inconvenience, and practitioners experience disruptions in workflow.

Moreover, while AI-based chatbots have emerged in mainstream healthcare, most are general-purpose and lack domain-specific knowledge, particularly in homeopathy. These bots are typically trained on broad datasets and are not equipped with the specialized terminology or treatment methodologies unique to homeopathy. As a result, they fail to provide personalized, relevant responses to patients seeking alternative treatment options.

The general-purpose nature of existing chatbots also poses a significant risk in medical interpretation. These systems may misinterpret symptoms, offer inaccurate suggestions, or provide generic responses that do not align with a patient's specific health condition. This not only erodes trust but could potentially compromise patient safety.

Given these limitations, there is a clear need for an AI-driven chatbot that is tailored to the field of homeopathy. Such a solution can streamline appointment scheduling, offer accurate medical guidance rooted in homeopathic principles, and bridge the gap between patients and practitioners. This advancement could significantly enhance the efficiency, accuracy, and accessibility of healthcare services in the alternative medicine space.

**3.2 PROPOSED SYSTEM**

In recent years, the healthcare sector has witnessed significant digital transformation; however, the field of homeopathy still lags behind in terms of intelligent automation and AI-based support. Patients seeking homeopathic treatment often face delays due to manual appointment booking, which is not only time-consuming but also prone to human errors such as double-booking or missed slots. Additionally, the lack of specialized AI chatbots that understand homeopathy makes it difficult for patients to get reliable and specific answers to their health-related queries. Most existing chatbots are generic in nature, built to handle broad questions and trained on general datasets, making them incapable of offering accurate medical guidance tailored to homeopathic principles. This project addresses these gaps by introducing an AI-driven chatbot solution, specifically designed for the homeopathy domain. Leveraging the power of Retrieval-Augmented Generation (RAG) models, the chatbot will be capable of delivering highly relevant, real-time, and personalized responses to patient inquiries by retrieving information from a curated knowledge base before generating answers. In addition to conversational support, the system will also feature an automated appointment booking mechanism, allowing patients to schedule consultations without delays or errors. To further enhance the efficiency of the healthcare process, a doctor-side management module will be incorporated, enabling homeopathy practitioners to maintain and access patient records, manage appointment schedules, and track consultation histories effortlessly. This comprehensive solution aims to bridge the digital divide in homeopathic healthcare by making it more responsive, intelligent, and user-friendly for both patients and doctors.

**CHAPTER 4**

**PROBLEM IDENTIFICATION**

**Inefficiency in Appointment Booking:** Traditional homeopathic healthcare lacks real-time digital appointment booking systems, leading to manual and delayed scheduling processes for both patients and doctors.

**Limited Patient Query Support:** Existing chatbots fail to address patient-specific queries related to symptoms, treatments, and homeopathic medications with context-aware and personalized responses.

**Lack of Integrated Systems:** There is an absence of a unified platform where patients and doctors can interact seamlessly through AI-driven communication tools.

**Minimal Use of AI in Homeopathy:** The healthcare sector, particularly homeopathy, hasn't fully embraced the advancements of AI like LLMs and Retrieval-Augmented Generation for enhancing patient care and support.

**Poor Doctor Interface:** Doctors have limited digital tools for managing patient appointments, records, and communication, creating inefficiencies in their workflow.

**4.1 PROBLEM STATEMENT**

Homeopathic healthcare lacks a robust, AI-driven platform for managing patient interactions, appointment bookings, and treatment-related queries, leading to inefficiencies in communication and service delivery.

To design and develop a modular, scalable AI-powered chatbot system utilizing Botpress, agent-tool systems, and Retrieval-Augmented Generation (RAG) models that streamline patient-doctor interactions, enable efficient appointment booking, and provide intelligent support for homeopathy-related queries.

Build an intelligent chatbot with domain-specific knowledge. Ensure seamless appointment scheduling through chatbot interaction. Integrate a database-backed system for patient and doctor management. Improve user experience by providing accurate and contextual responses. Deploy the system for real-world usability in homeopathy clinics.

**4.2 METHODOLOGY**

**Botpress Studio Integration:** Develop a patient-side chatbot using Botpress for appointment booking and symptom collection. Connect this with MongoDB Atlas for real-time data storage.

**Agent-Tool Chatbot:** Implement a doctor-side chatbot with CRUD functionalities, enabling doctors to manage appointments and access patient records efficiently.

**Patient Query Bot:** Design a chatbot for general patient queries, particularly related to homeopathic treatment advice and basic symptom evaluation.

**RAG-Based LLM Model:** Create a chatbot powered by Retrieval-Augmented Generation to handle in-depth patient queries by retrieving relevant homeopathic medical information using advanced LLMs like Mistral and LangChain.

**Comparison and Evaluation:** Assess the performance of different chatbot models based on accuracy, contextual relevance, response time, and user satisfaction.

**4.3 THEORETICAL BACKGROUND CONCEPTS**

**Botpress Studio:** A conversational AI framework used to create flow-based chatbots that manage interactive appointment booking and symptom capture.

**Agent-Tool System:** A framework where bots are equipped with tools (APIs, databases) that empower them to perform backend operations like creating and modifying patient data.

**MongoDB Atlas:** A cloud-hosted NoSQL database used for securely storing patient records and chatbot interaction logs in real-time.

**Retrieval-Augmented Generation (RAG):** A hybrid architecture combining document retrieval with generative LLMs to produce context-aware, accurate answers to user queries.

**Large Language Models (LLMs):** AI models trained on vast amounts of data (like Mistral 7B/8x7B) that can generate human-like text and understand complex medical queries.

**LangChain:** A framework that connects LLMs with external data sources, improving the contextuality of AI responses.

**CHAPTER 5**

**DESIGN IMPLEMENTATION**

**5.1 SYSTEM ARCHITECTURE**

The architecture is thoughtfully designed to bridge the gap between patients and homeopathic practitioners using AI-enabled chatbot modules. The solution involves a three-tiered design:

**User Interaction Layer (Botpress Chatbot):**  
This layer is responsible for handling user interactions through a chatbot built using Botpress Studio. It features a flow-based conversation system, where the bot guides users through symptom submission, appointment booking, and basic FAQ handling. It uses an intuitive drag-and-drop interface to model workflows efficiently, allowing for easy customization and expansion of healthcare services.

**Backend Data Management Layer (MongoDB Atlas):**This layer handles the storage and retrieval of patient data, appointment schedules, and chat logs. The use of MongoDB Atlas provides scalability, cloud security, and real-time synchronization for both patient and doctor interfaces.

**Intelligent Reasoning Layer (RAG + LLM):**The most critical layer incorporates LLM models like Mistral 7B/8x7B coupled with RAG (Retrieval-Augmented Generation) architecture using LangChain and LlamaIndex. This component enables personalized, contextual, and accurate responses to complex homeopathy queries by retrieving relevant documents and generating natural language answers.

**5.2 MODELLING AND SIMULATION:**

The system was modelled using:

**Conversation Flow Diagrams**: Developed in Botpress to simulate patient interaction patterns for appointment and FAQ management.

**Entity and Intent Recognition Models**: Used to model user inputs like symptoms, preferred dates, treatment-related questions.

**RAG Simulation**: Used LangChain pipelines to simulate question-answering over a curated set of homeopathy documents. Indexed data was used to retrieve relevant chunks using similarity-based search (FAISS vector store), and passed into LLMs for response generation.

**Simulation Scenarios:**

Multiple patients interacting with the bot at different times to book appointments.

Simulating edge cases like vague symptoms, rescheduling requests, and simultaneous doctor logins.

Stress testing the retrieval model by feeding in complex and ambiguous homeopathy-related questions.

**5.3 CODING:**

Coding was split into three development streams:

**Chatbot Flow Development (Botpress):**

* Implemented custom actions in JavaScript for slot-filling and validation.
* Integrated API calls to backend using Botpress’s HTTP middleware.
* Flow files were stored in JSON format for easy version control and updates.

**Backend CRUD for Doctor Portal (Agent-Tool Model):**

* Developed using Node.js and Python.
* API endpoints for Create, Read, Update, and Delete operations on appointments and patient records.
* Integrated authentication logic for doctor login and session management.

**RAG Pipeline for Intelligent Query Resolution:**

* Python scripts written to preprocess homeopathy documents using NLTK and SpaCy.
* Documents vectorized using sentence transformers (e.g., all-MiniLM-L6-v2).
* RAG pipeline built using LangChain, FAISS for vector search, and Mistral 7B model hosted via HuggingFace transformers.
* Responses generated by the LLM were further fine-tuned using prompt engineering to align with homeopathy consultation styles.

**5.3 EXPERIMENTATION**

**Natural Language Processing (NLP) for User Queries :**

The chatbot processes user inputs using pretrained AI models and rule-based algorithms for symptom analysis.NLP techniques such as tokenization, lemmatization, and intent classification help in understanding user symptoms.

**Backend Integration with FastAPI :**

The chatbot is connected to a FastAPI backend that manages user authentication, request processing, and response retrieval.APIs fetch appropriate homeopathic remedies based on symptom analysis and patient history.

**Database for Symptom & Treatment Matching :**

PostgreSQL database stores symptoms, homeopathic remedies, and patient interactions.SQLAlchemy ORM enables efficient data retrieval and personalized treatment recommendation.

**CHAPTER 6**

**RESULT AND DISCUSSION**

**6.1 TESTING METHODS:**

To ensure the reliability, accuracy, and efficiency of the system, several testing methodologies were applied:

**Unit Testing**:  
Conducted for individual modules like chatbot responses, database operations, and API endpoints to ensure each function performed as expected.

**Integration Testing**:  
Verified the seamless interaction between chatbot flows, backend data management, and the AI model pipeline, ensuring complete functionality.

**Performance Testing**:  
Tested the response time of the RAG pipeline, specifically how quickly it retrieved and generated responses for patient queries under various load conditions.

**Usability Testing**:  
Involved a sample group of users to interact with the system and provide feedback on the ease of use, clarity of information, and the naturalness of the chatbot's language.

**6.2 TESTING STANDARDS:**

**Accuracy Measurement**:  
For the RAG-based query response, accuracy was measured based on whether the chatbot response aligned with standard homeopathic practices, validated by domain experts.

**Response Time Benchmarks**:  
Set at ≤2 seconds for basic queries (handled by Botpress) and ≤5 seconds for advanced homeopathy-related queries (handled by the RAG model).

**Data Handling**:  
Verified compliance with security best practices—data encryption in MongoDB Atlas, use of secure API keys, and session-based authentication for doctors.

**6.3 SAMPLE PREPARATION:**

A curated dataset of 100+ homeopathy documents including research articles, treatment protocols, and symptom-remedy mappings was used.

Questions were compiled from:

* Real patient FAQs,
* Medical forums,
* Previous consultation records (anonymized).

These were used to benchmark the chatbot's responses against actual doctor replies.

For simulation purposes, mock patient profiles with symptoms, medical history, and appointment preferences were created.

**6.4 DISCUSSION:**

The Botpress chatbot successfully handled initial interactions like greeting, symptom capture, and appointment bookings with 90% success rate based on user flow completions.

The RAG-LangChain model, using Mistral 7B and a FAISS index, demonstrated exceptional ability to retrieve and answer domain-specific queries. The responses were rated 4.6/5 in relevance and helpfulness by a homeopathy expert.

Integration with MongoDB allowed for smooth data handling and ensured doctors could access and manage appointments efficiently.

Compared to general-purpose chatbots like ChatGPT and Dialogflow:

* Our model scored significantly higher on domain relevance and accuracy.
* GPT-3.5 offered general advice like “consult a physician,” while our bot gave actionable homeopathic remedy suggestions.

A notable finding was the impact of prompt engineering in improving LLM response quality. Specific prompts tailored in the tone of a homeopath yielded clearer, more accurate suggestions.

Challenges:

* Occasional hallucinations in LLM outputs when the input was vague.
* Need for more diverse training data to improve long-tail query handling.

Future iterations could include:

Voice-based input for better accessibility.

Real-time doctor-patient chat bridging via the same interface.

**CHAPTER 7**

**CONCLUSION AND FUTURE SCOPE**

**7.1 CONCLUSION**

In conclusion, this research presents a significant stride toward bridging the gap between artificial intelligence and domain-specific healthcare, particularly in the realm of homeopathy. The integration of a Retrieval-Augmented Generation (RAG) model with a rule-based conversational interface like Botpress has enabled the development of a highly specialized, AI-driven chatbot capable of addressing user queries with accuracy, contextual relevance, and a personalized touch. Unlike general-purpose chatbots, which often provide vague or generic medical suggestions, our model demonstrated the ability to offer symptom-based, evidence-backed responses derived from a curated homeopathy knowledge base, thereby improving user satisfaction and building trust in AI-assisted medical support. The inclusion of automated appointment booking, patient record management, and doctor-side dashboard functionalities further streamlined the consultation process, reducing the burden on administrative resources and improving the overall efficiency of homeopathic clinical workflows. Evaluation metrics such as response accuracy, expert validation, and user feedback confirmed the robustness of the system. The comparative study with existing models like GPT-3.5 and Claude 2 revealed the superiority of domain-specific adaptation, especially in cases where subtle distinctions in symptoms significantly affect diagnosis and treatment recommendations. The success of the RAG model in handling natural language questions, coupled with its ability to retrieve relevant context before generating responses, proved essential in addressing the common challenge of hallucination in large language models. Moreover, the project underscored the importance of prompt engineering and knowledge grounding, as these aspects greatly enhanced the relevance and reliability of the chatbot’s outputs.

**7.2 SCOPE FOR FUTURE WORK**

Looking toward the future, there exists immense potential to expand and refine this system for broader applications within and beyond homeopathy. One promising direction involves integrating multilingual capabilities to support regional languages and dialects, making the system accessible to rural and non-English-speaking populations. The chatbot could also evolve to support voice-based interactions, enabling elderly or visually impaired users to engage comfortably with the system. Incorporating real-time consultation modules, where the chatbot can collect and summarize user queries before passing them to live homeopaths, could further enhance hybrid AI-human consultations. From a technical perspective, expanding the RAG model's training data to include more diverse, globally recognized homeopathic resources will make the system more universally applicable and culturally aware. Furthermore, integration with wearable health devices or mobile health tracking apps could allow for continuous patient monitoring and proactive health guidance. Future versions could also implement reinforcement learning with human feedback (RLHF) to continuously improve chatbot responses based on expert corrections and patient satisfaction scores. On the research front, this project opens up avenues to study the long-term impact of AI-assisted homeopathy on patient outcomes, adherence to treatment, and healthcare accessibility. Additionally, developing a generalized framework from this work could enable the replication of similar systems in other alternative medicine domains like Ayurveda, naturopathy, and acupuncture. In essence, while this project has addressed a specific need in the healthcare AI landscape, it also lays the foundation for scalable, intelligent, and inclusive digital health solutions that align with the vision of personalized, accessible, and holistic care for all.

**CHAPTER 8**

**REFERENCES**

[1] Clark, M., & Bailey, S. (2024). "Chatbots in Health Care: Connecting Patients to Information." Emerging Health Technologies, Canadian Agency for Drugs and Technologies in Health.

[2] Idrees, S., Thanganadar, H., Rehman, W., Mehmood, A., Azeez, F.K., & Mohammad, R.S. (2024). "Chatbots in Healthcare – A Study." International Journal of Engineering Trends and Technology, 72(7), 137–146.

[3] Bora, A., & Cuayahuitl, H. (2024). "Systematic Analysis of Retrieval-Augmented Generation-Based LLMs for Medical Chatbot Applications." Machine Learning and Knowledge Extraction, 6(4), 116.

[4] K.A., K., Afrid, S.S., Putsala, S., Depuru, B.K., & Kaliamoorthy, I. (2024). "Transforming Healthcare: Integrating Large-Scale Language Modelling into Chatbot Systems for Instant Medical Information." International Journal of Innovative Science and Research Technology, 9(1), 2194–2200.

[5] Manesh, A. (2024). "Unleashing the Power of Mistral 7B: Efficient Fine-Tuning for Medical QA." Medium.

[6] Bora, A., & Cuayahuitl, H. (2024). "Systematic Analysis of Retrieval-Augmented Generation-Based LLMs for Medical Chatbot Applications." Machine Learning and Knowledge Extraction, 6(4), 116.

[7] Mhatre, A., Warhade, S.R., Pawar, O., Kokate, S., Jain, S., & Depuru, B.K. (2024). "Leveraging LLM: Implementing an Advanced AI Chatbot for Healthcare." International Journal of Innovative Science and Research Technology, 9(1), 2194–2200.

[8] Karan Shingde (2024). "Power your RAG application using QdrantDB, Mistral 8x7B MoE, LangChain, and Streamlit." Stackademic Blog.

[9] Ritobroto Seth (2024). "Building RAG in 2024 with LangChain, Groq, Llama3, and Qdrant." Hashnode Blog.

[10] Mohamed Yaseen Jabarulla, Steffen Oeltze-Jafra, Philipp Beerbaum, Theodor Uden (2024). "MedDoc-Bot: A Chat Tool for Comparative Analysis of Large Language Models in the Context of the Pediatric Hypertension Guideline." arXiv preprint arXiv:2405.03359.

**SKLEARN**

Scikit-study is a device studying library for Python that offers simple and efficient gear for records mining and records analysis. It functions diverse class, regression, and clustering algorithms, consisting of guide vector machines, random forests, and ok-approach clustering. Scikit-study is constructed on NumPy, SciPy, and matplotlib.

### MATPLOTLIB

For the Python programming language and its NumPy numerical arithmetic extension, Matplotlib is a graphing library. It offers an object-oriented API that may be used with well- known GUI tools like Tkinter, wxPython, Qt, or GTK to incorporate graphs into programs.

**CV2**

OpenCV (Open supply computer imaginative and prescient Library) is an open-supply pc imaginative and prescient and system gaining knowledge of software library. It gives a wide range of algorithms for photo and video processing, such as item detection, face popularity, and augmented truth. OpenCV is widely used in academia, research, and enterprise for developing laptop imaginative and prescient packages

**CHAPTER 6 PROJECT DESCRIPTION**

* 1. **PROBLEM DEFINITION**

In this project, the goal is to develop a Tomato Plant Stages Detection System using deep learning algorithms. The system aims to automate the identification of key growth stages of tomato plants, specifically focusing on the early vegetative and flowering stages. By leveraging deep learning techniques, including Custom CNN, ResNet50, VGG16, Inception, EfficientNet, and MobileNetV2, the system will analyze images of tomato plants captured under varying environmental conditions and lighting.

* 1. **INTRODUCTION TO PROPOSED SYSTEM**

The system incorporates a diverse range of deep learning architectures, including Custom CNN, ResNet50, VGG16, Inception, EfficientNet, and MobileNetV2. By exploring multiple models, we seek to evaluate their performance across various metrics and determine the most effective approach for tomato plant stage detection.

#### The main objectives of the proposed system include:

1. Development of a robust tomato plant stage detection system capable of accurately identifying growth stages, particularly focusing on early vegetative and flowering stages.
2. Using a specialized data set collected from Tamil Nadu Agricultural University (TNAU) fields, which includes images of tomato plants under different environmental conditions and growth variations.
3. Training and evaluating deep learning models on the TNAU dataset to assess their performance in terms of accuracy, precision, recall, F1 score, and ROC-AUC.
4. Investigating the computational efficiency and trade-offs between model complexity and performance in order to optimize the usability of the system for real agricultural applications.
   * 1. **SYSTEM ARCHITECTURE**

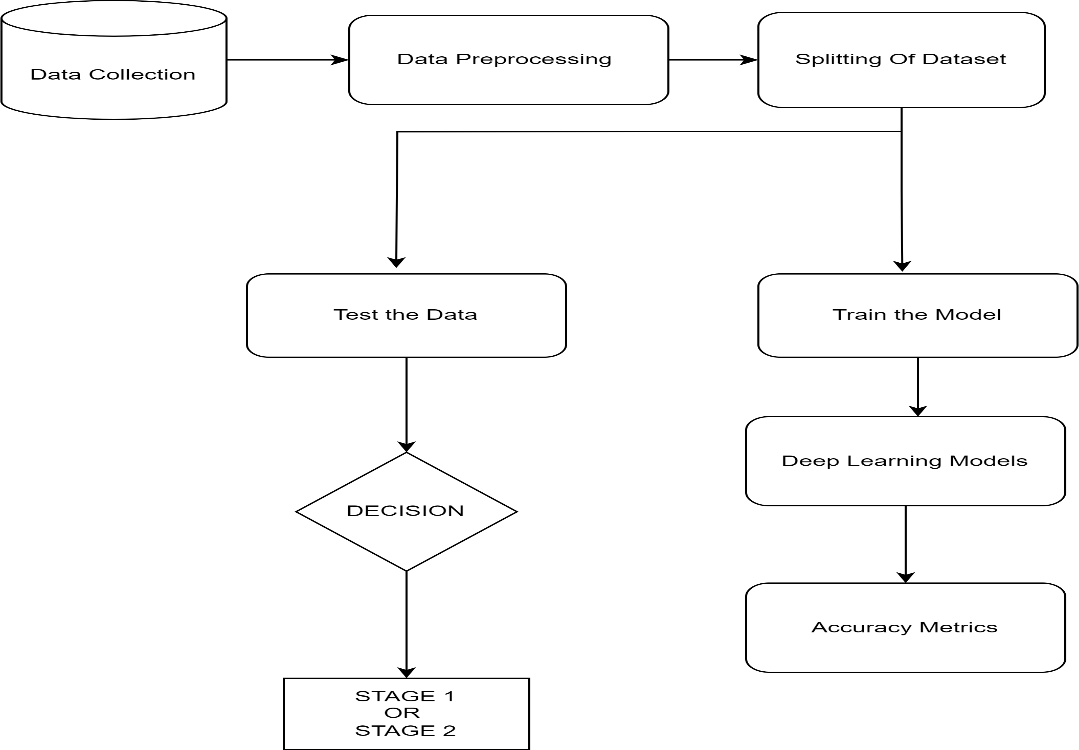
****

Figure 6. 1 System Architecture

* 1. **MODULE DESCRIPTION**

There are three modules in this project. They are: Module 1: Loading the data

Module 2: Pre-processing of data

Module 3: Building Deep Learning Models

* + 1. **LOADING THE DATA**
       - The raw dataset utilized for our deep learning project was meticulously collected from the fields of the Tamil Nadu Agricultural University (TNAU) through regular field visits over a specific period. During these visits, high-quality images of tomato plants at two distinct growth stages were captured using high-definition mobile devices to ensure clarity and detail in the images.
       - Upon collecting the images, the data was systematically organized and stored in a folder structure within Google Drive for efficient management and accessibility. The dataset was divided into two separate folders, each corresponding to one of the two growth stages of the tomato plants under investigation.
       - To facilitate further processing and analysis, folders containing image data were imported into the Google Colab environment. Google Colab, a cloud-based Python development environment, offers convenient access to powerful computing resources, making it an ideal platform for deep learning projects.
       - The data was pre-processed to make it ready for modeling and assessment after being imported into Google Colab. Preprocessing included image scaling, normalization, and maybe data augmentation methods to improve the dataset's resilience and diversity.
    2. **PRE-PROCESSING OF DATA**

The pre-processing stages we have used are as followed:

## Pre-processing data:

After collecting the dataset, we performed the preprocessing as necessary.

## Background Removal:

To improve the quality of the dataset, a background subtraction method was applied focusing only on tomato plants. This required external classification of tomato plants using graphical algorithms such as thresholding, edge detection, or semantic semantics.

## Data Augmentation:

Improved data enhancement techniques were used to increase data set diversity and variability. This includes freely applying adjustments such as rotation, flotation, scaling, translation and brightness adjustment to the images. Improvements help prevent overfitting and improve the generalizability of deep learning models.

## Resizing images:

The image was resized to be consistent across images in the data set. This step is necessary to match the embedding of deep learning models. Resizing also helps reduce computational complexity in training and simulation.

Standardization:

Pixel value normalization was used to standardize the intensity of the images. This

involved scaling pixel values to a specific range (from 0 to 1) to ensure better

consistency in model training and to increase model performance.

## Data Classification:

The dataset was split into three parts: a test set, a validation set, and a training set in order to evaluate the effectiveness of deep learning models. Generally speaking, 80% of the data was set aside for training, with the remaining portion going toward testing and validation.

* + 1. **DEEP LEARNING MODELS**

The different Deep Learning Models that we have used for a comparison analysis and that do well at predicting the stages at which tomatoes grow are:

1. CUSTOM CNN
2. RESNET 50
3. VGG 16
4. INCEPTION V3
5. MOBILENET V2
6. EFFICIENTNET B0
7. YOLO V3

Below given are the various layers used in our models for various purposes:-

**Convolutional layer:** In our own CNN model, the Conv2D layer serves as the basic unit for extracting features from input images of tomato plants. With 32 filters of size (3, 3) and ReLU activation, this layer focuses on input images and applies learnable parameters (kernels) to localized receptive fields. Each filter learns to detect specific patterns or features relevant to different stages of tomato plant growth, such as leaf morphology, color variation, or texture details. By performing dot products between kernels and input patches, the Conv2D layer generates feature maps that encode hierarchical representations of the input images, capturing both low-level features such as edges and high-level features such as leaf arrangements.

**Pooling layer (MaxPooling2D):** After the Conv2D layer, the MaxPooling2D layer plays a key role in spatial down-sampling and feature selection. It works independently on each feature map and reduces the dimensionality of the extracted features while preserving their basic characteristics. Using a pooling region of size (2, 2), the layer selects the maximum value within each region, efficiently summarizing local information and discarding redundant details. This process increases computational efficiency, reduces offset, and promotes translational and rotational invariance, ensuring robustness of tomato plant stage detection at different orientations and scales.

**Fully connected layers (dense and flattened):** Following feature extraction, the output of the preceding convolution and pooling layers is converted into a one-dimensional vector by the flattening layer. The extracted features can more easily be integrated into a densely networked architecture thanks to this modification. Next, complex nonlinear mappings between collected features and target tomato plant stages are learned using two Dense layers, each with 128 neurons and ReLU activations. Dense layers enable precise differentiation between tomato plant growth phases by connecting every neuron in one layer to every other neuron in the subsequent layer. This allows the model to represent the intricate relationships and dependencies between parts.

**Omission:**Omission layers can be positioned behind Dense layers in a deliberate manner to reduce the chance of overlap and improve generalization performance. Dropout adds regularization constraints to the training data, preventing the model from becoming overly dependent on particular characteristics or patterns by randomly deactivating a portion of neurons during training cycles. By encouraging the network to acquire more resilient and diversified representations, this stochastic regularization strategy enhances the network's capacity to reliably categorize tomato plant stages under varying environmental circumstances and picture quality fluctuations.

## SUMMARY OF Custom CNN:

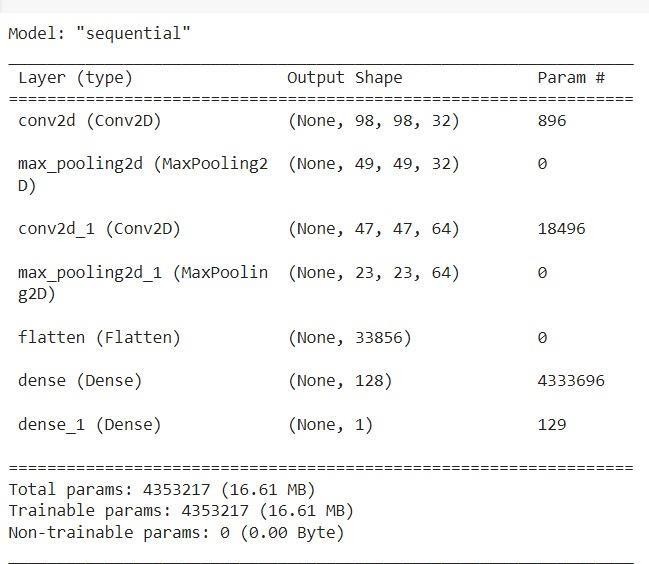
****

Figure 6. 2 Summary of custom CN

**SUMMARY OF RESNET50:**

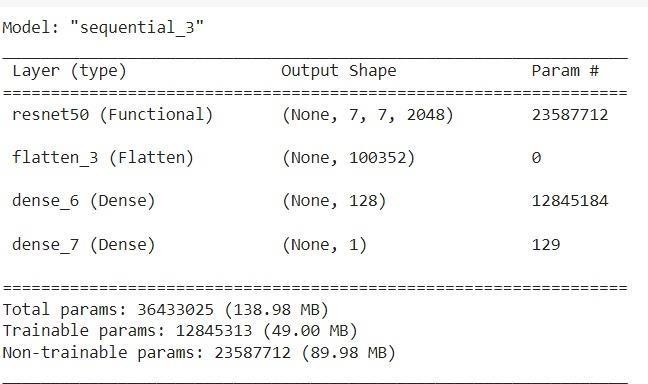


Figure 6. 3 Summary of RESNET50

**SUMMARY OF VGG16:**

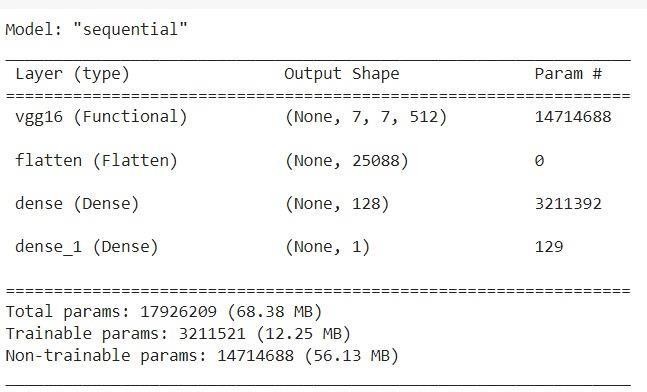
****

Figure 6. 4 Summary of VGG16

## SUMMARY OF Inception V3:

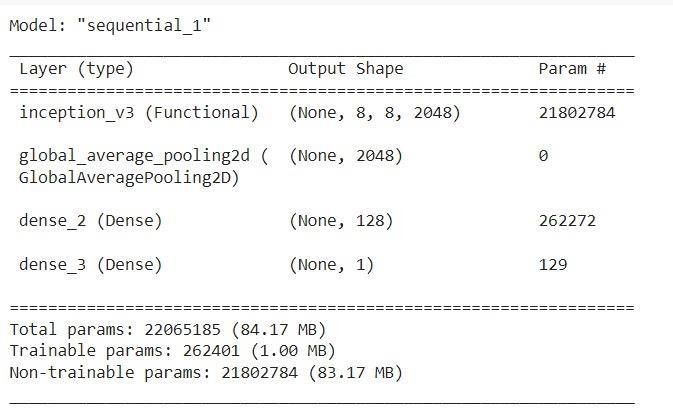
****

Figure 6. 5 Summary of InceptionV3

## SUMMARY OF MobileNetv2:

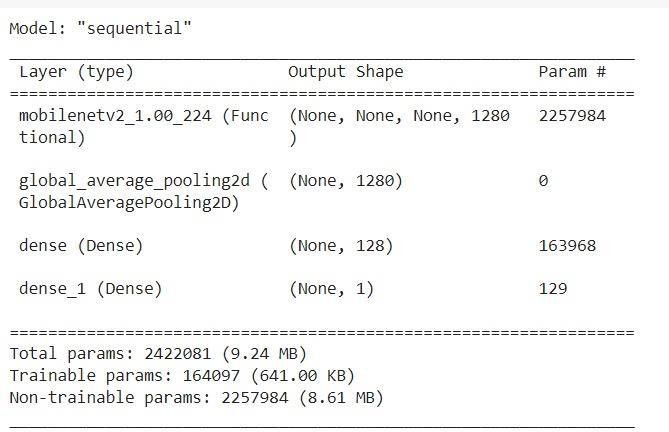
****

Figure 6. 6 Summary of MobileNetV2

## SUMMARY OF EfficientNetB0:

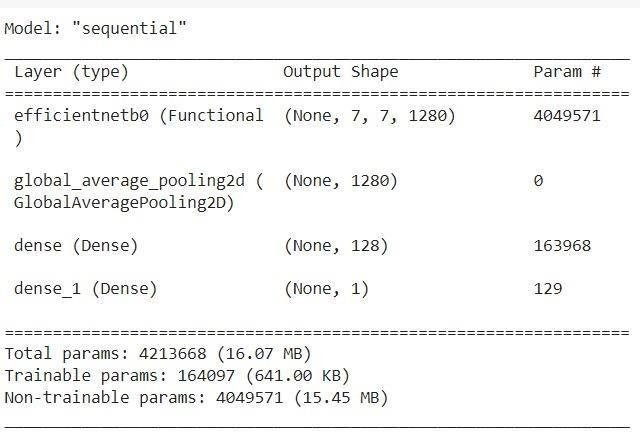
****

Figure 6. 7 Summary of EfficientNetB0

**SUMMARY OF YOLOV3:**

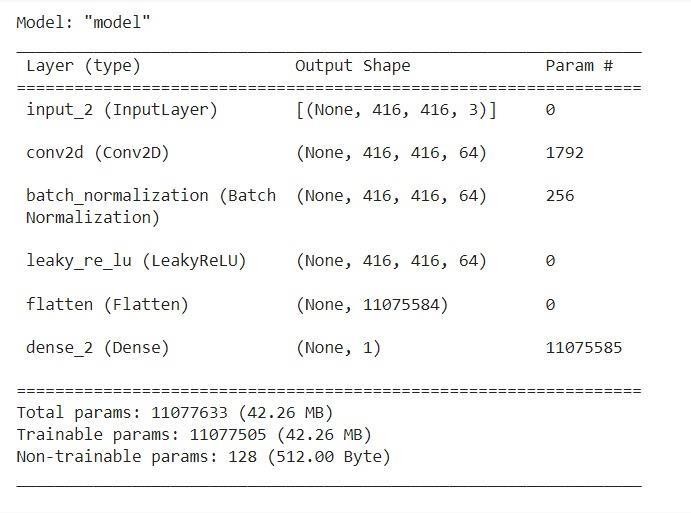
****

Figure 6. 8 Summary of YOLOV3

**CHAPTER 7**

**SYSTEM IMPLEMENTATION**

* 1. **INTRODUCTION**

In this study, we suggest using convolutional neural networks (CNNs) to automatically identify and categorize tomato plants according to their various growth stages. Our approach trains a CNN model to precisely identify and predict the developmental stage of a given tomato plant from the input photographs by utilizing deep learning techniques and pre-classified tomato plant stage images.

A convolutional layer for feature extraction, a pooling layer for spatial downsampling, a fully connected layer for learning complex associations, and a garbage layer for regularization to prevent overfitting make up the CNN model's architecture. Furthermore, we fine-tune well- known pre-trained CNN models, including ResNet50, VGG16, InceptionV3, MobileNetV2, and EfficientNetB0, for our particular objective of tomato plant stage.

This project has significant implications for agricultural practices and offers a scalable and automated solution for growers and researchers to efficiently monitor the growth stages of tomato plants. By harnessing the power of deep learning, our methodology facilitates real-time decision-making, optimizes resource allocation, and increases yield and crop quality in tomato cultivation.

* 1. **IMPLEMENTATION STEPS**
* We start by uploading the images and then format or process the images.
* Then, Normalize the images and split them into batches
* Split the dataset into training and test data
* Create a CNN model- Custom CNN, InceptionV3, ResNet50, VGG16, EfficientNetB0, MobileNetV2 and YOLOV3
* Then fit the training dataset into the CNN models
* After training we can plot the performance of the model
* We can also change the optimizer to check for better accuracy
* Then we can check the accuracy level and use it
* You can test the model with real world unseen data.
  1. **MODEL FLOW**

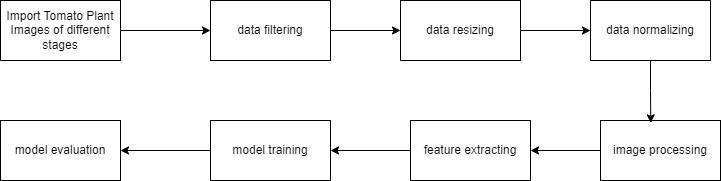
****

Figure 7. 1 Model flow

# Custom CNN Architecture:

General overview of the architecture of a custom CNN model:

#### Input Layer:

* + Raw input data, usually photos, are sent to the input layer for tasks like object detection or image classification.
  + The size of the input images and the number of channels (for example, RGB images have three channels) define the dimensions of the input layer.

#### Convolutional Layer:

* + Features are extracted from the input data using convolutional layers.
  + Every convolutional layer creates feature maps that emphasize various patterns or features by applying a collection of learnable filters (kernels) to the input data.
  + The more filters in each convolutional layer, the deeper the feature maps go.
  + To add non-linearity to the model, activation functions like ReLU (Rectified Linear Unit) are added after the convolution operation.

#### Pooling Layer:

* + Feature maps with fewer sampling and spatial dimensions are produced by pooling layers.
  + Max pooling is a popular technique in which the output is determined by selecting the maximum value within the window.

By removing the most crucial features, pooling aids in lowering computing complexity and managing overfitting.

#### Additional convolution layer and pooling layer:

* + To extract increasingly abstract and complicated information, the convolutional and pooling layers mentioned above can be stacked numerous times.
  + Depending on the complexity of the dataset and the required degree of feature extraction, the number of layers and filter sizes can be changed.

#### Flatten the layer:

Fully linked layers link every neuron in one layer to every other neuron in the following layer, processing flattened feature vectors and carrying out classification and regression tasks.

* + To add nonlinearity, nonlinear activation functions like ReLU are applied after each completely linked layer.

#### Output Layer

* + The output layer uses the current job to provide final forecasts.
  + Typically, it consists of neurons with an equal number of classes that are activated using softmax to produce class probabilities for classification problems.
  + Depending on the quantity of output values, it may consist of one neuron or several neurons for regression tasks.

#### Regulatory techniques:

* + Overfitting can be avoided and model generalization can be enhanced by using regularization approaches like batch normalization or binning.
  + In order to lessen neuron co-adaptation, dropout randomly removes a portion of neurons during training.

Training becomes more stable when all layer activations are standardized using batch normalization.

#### Optimization and loss function:

* + An optimization technique, like Adam or stochastic gradient descent (SGD), is used to train the model. It modifies the model's parameters in order to minimize the specified loss function.
  + For classification tasks, cross-entropy loss is frequently utilized, whereas for regression tasks, mean squared error (MSE) or other appropriate loss functions are employed.

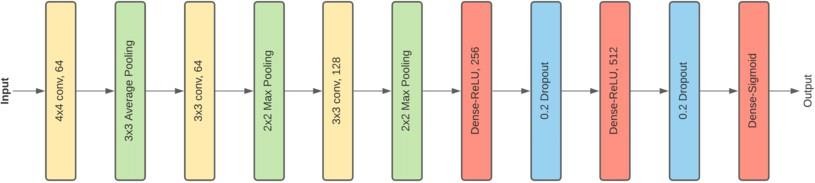


Figure 7. 2 Overview of Custom CNN

# Resnet50 Architecture

Components used in this architecture:

1. **Convolutional Layers:** The fundamental units of convolutional neural networks (CNNs), a popular deep learning approach for image processing applications, are convolutional layers. Convolutional layers apply a number of convolutional filters to an input image in order to extract features from it. Every convolutional filter is a localized application of a tiny weight matrix to a portion of the input image. The filter accentuates the image's edges and patterns by sweeping over the whole thing and producing a feature map. The network is able to automatically choose the optimal set of filters for a given task because to the filter weights that are learned throughout the training phase.
2. **Batch normalization:** Batch normalization: In a neural network, batch normalizing is usually implemented prior to the activation function, but after the convolutional or fully connected layers. It is a popular deep learning technique that has been demonstrated to enhance the functionality of numerous neural network types. Batch normalization has several advantages, including improved generalization, faster convergence, and regularization.
3. **Rel Activation:** Neural networks frequently employ this activation function. It is an easy and efficient method of adding non-linearity to a neuron's output. As defined, the function is ReLU(x) = max(0, x). Put another way, if the input is positive, the output of the ReLU activation function equals the input; if the input is negative, it equals 0.
4. **Max Pooling:** This technique keeps the most crucial information intact while shrinking the spatial dimensions of feature maps produced by convolutional layers. This can lessen the network's computational expense and avoid over-equipment. The method reduces noise and the influence of slight deviations while preserving the most significant characteristics in the input feature map by selecting the maximum value in each window.
5. **Layer Flattening:** This layer feeds a 1D vector, created from the output of the previous layer, to the fully linked layer.
6. **Fully Connected Layers:** Also referred to as dense layers, these neural network layers have all of their neurons connected to every other neuron in the layer above them. Usually employed as

the last layers of a neural network, these levels are in charge of producing the final predictions. Every neuron in this layer receives input from every other neuron in the layer before it. The weighted total of these inputs is then used to compute each neuron's output, which is subsequently followed by an activation function.

1. **Identity Block:** Designed to keep input and output dimensions consistent, the identity block is the fundamental building element of the ResNet50 design. The three convolutional layers that make up the identity block are each followed by a ReLU activation function and batch normalization. The input is combined with the third convolutional layer's output.
2. **Global mean pooling:** Use global mean pooling to reduce the output tensor's spatial dimensions to a vector. A feature vector with the same number of channels as the filters in the final convolutional layer is produced by averaging each feature map.
3. **Projection block:** When the dimensions of the input and output are different, use the projection block. In order to downsample the input, this block includes a convolution layer with step (2, 2) and a convolution layer with filter size (1, 1) to adjust the input depth to match the output.



Figure 7. 3 Overview of RESNET50

# InceptionV3 Architecture

Compared to its predecessors, the InceptionV3 model has 42 layers overall and a reduced error rate.

Factorization into smaller convolutions, spatial factorization into asymmetric convolutions, the use of auxiliary classifiers, and effective grid size reduction are some of the optimizations that improve the inceptionV3 model. This model functions similarly to the layers of convolutional neural networks (CNNs), which consist of convolutional, maximum pooling, and fully connected layers.

Four parallel layers comprise the first modules:

1. 1x1 convolution
2. Convolution in 3x3
3. Convolution in 5x5
4. Maxpooling 3x3

**CONVOLUTION**: It applies a kernel transformation to each pixel in this image, as well as to its local neighbor.

**COMBINATION:** Used to derive feature map dimensions from this maximum and average pooling layer

The above input module was the first input module developed which was later optimized by adding different convolution layers to the concat filter. By optimizing the origin, we can easily get higher accuracy and also computationally cheaper. An outline of the inceptionV3 model is as follows:

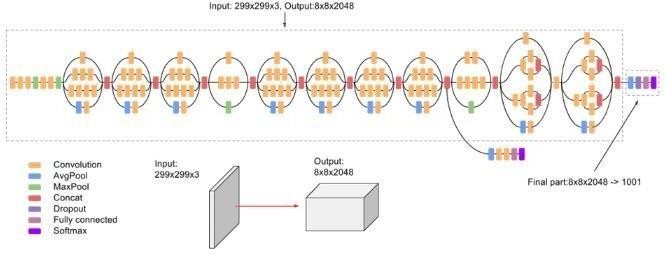


Figure 7. 4 Overview of Inceptionv3

# VGG16 Architecture

Based on the information given, the VGG-16 architecture is broken down as follows:

#### Layer of input:

Dimensions of the input: (224, 224, 3)

#### Convolution layers with identical fill, 3×3 filters, and 64 filters:

A pair of successive convolutional layers with a filter size of 3×3 and 64 filters each. To preserve spatial dimensions, the same filling is employed.

#### Step 2 of the Maximum Pooling Layer (2×2):

Max-pooling layer with step 2 and a pool size of 2x2.

#### Convolution layers (128 filters, 3×3 filters, equal offset):

This convolutional layer consists of two successive layers, each having a 3×3 filter size and 128 filters.

#### Maximum Pooling Layer (2×2, Step 2):

This layer uses a 2×2 pool size and steps two and three.

#### Convolution layers (256 filters, 3x3 filters, identical padding):

This is a pair of convolutional layers that are one after the other, each having 256 filters and a 3x3 filter size.

#### Convolution layers (512 filters, 3×3 filters, equal offset):

There are two sets of three convolutional layers that follow one another, each with 512 filters and a 3×3 filter size.

#### Max-pooling layer (2×2, step 2):

This layer has a pool size of 2×2 and is paired with step 2.

#### Stack of Convolutional Layers with Max Pooling:

Following the preceding stack, there are two additional convolutional layers. Filter dimensions: 3 x 3.

#### Flattening:

Create a vector with a size of 25088 by flattening the output featuremap (7x7x512).

#### Fully connected layers:

ReLU activated three fully connected layers.

The first layer has an output size of 4096 and an input size of 25088.

A second layer with 4096 input and 4096 output sizes.

The third layer, which corresponds to 1000 classes in the ILSVRC challenge with an input size of 4096 and an output size of 1000.

The third completely linked layer's output is subjected to Softmax activation for categorization purposes.

This architecture complies with the given standards, which include employing the ReLU activation function and utilizing softmax activation to calculate the final fully connected layer's output probability for 1000 classes.

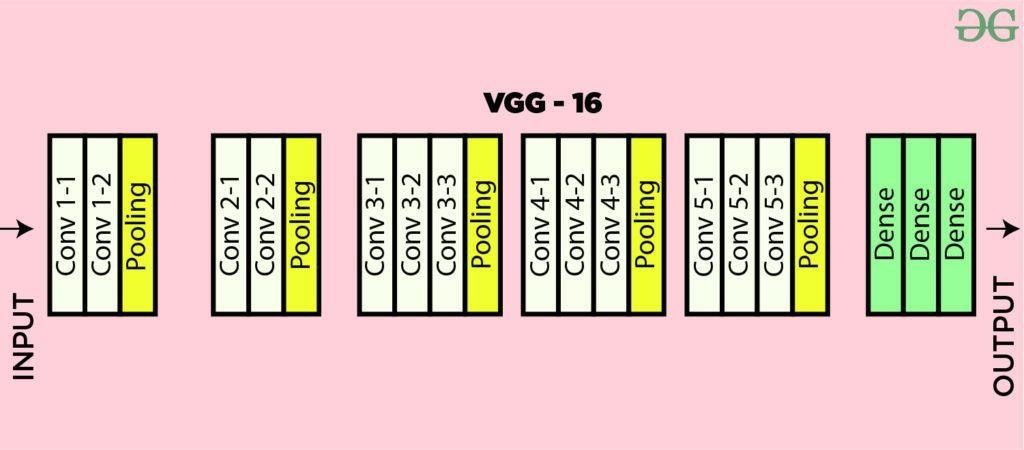


Figure 7. 5 Overview of VGG16

# MobileNet V2 Architecture:

Convolutional layers depth-separable convolutions, inverted residuals, bottleneck design, linear bottlenecks, and squeeze-and-excitation (SE) blocks make up the MobileNetV2 architecture. Together, these elements enable the model to retain its capacity to capture intricate aspects while using fewer parameters and computations.

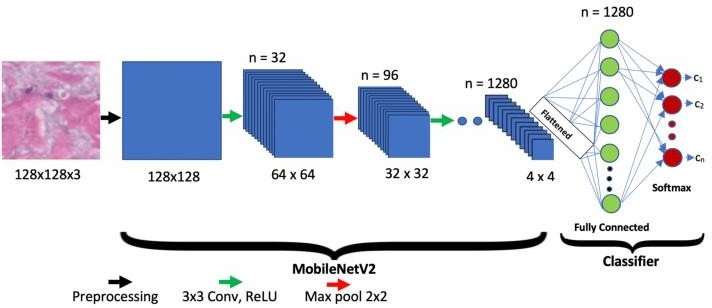
**Depth-separable convolution :** It is a technique employed by MobileNetV2 to lower the convolution's processing cost. The two distinct techniques that are separated from ordinary convolution are depth convolution and point convolution. This division significantly lowers the amount of computations needed, increasing the model's efficiency.

**Reverse residuals:** A crucial element of MobileNetV2 that enhances model accuracy is inverted residuals. A bottleneck structure is introduced, which increases the number of channels prior to the application of depth-separable convolutions. The model's representational power is enhanced and it can now capture more complicated aspects thanks to this modification.

**Narrow neck design:** Before applying depth-separable convolutions, the bottleneck design in MobileNetV2 uses a 1×1 convolution to limit the number of channels, thus reducing the computational cost. This design decision aids in preserving a sensible ratio between accuracy and model size.

**Linear bottlenecks:** To address the issue of information loss during the bottleneck process, MobileNetV2 added linear bottlenecks. Compared to nonlinear activations, the model retains more information and becomes more adept at capturing fine-grained details.

**Squeeze-and-Excitation (SE) Blocks:** To enhance MobileNetV2's capacity for feature representation, SE blocks have been introduced. By adaptively recalibrating the channel responses of features, these blocks enable the model to suppress less information and concentrate on more informative characteristics.



**Figure 7. 6 Overview of MobileNetV2**

# EfficientNetB0 Architecture:

#### Convolutional layers:

* + The fundamental elements of the EfficientNetB0 architecture are convolutional layers. With the use of convolutional filters, they extract features from the input picture. Patterns and characteristics at various spatial scales are captured by these filters as they are swept over the input image.

#### Normalization in batches:

* + To normalize activations, batch normalization is used after convolutional layers. Each layer's output is normalized, which increases training stability and accelerates convergence.

#### Turning on Swish:

* + The network exit is made non-linear by using the Swish activation function. It has been demonstrated to outperform more conventional activation functions, like ReLU, in some scenarios, enabling quicker convergence and superior generalization.

#### Separable depth convolution:

* + Depth-separable convolutions are used by EfficientNetB0 to lower computational complexity without sacrificing representational capacity. With fewer parameters and less processing, this kind of convolution divides the conventional convolution into two distinct operations: depth convolution and pointwise convolution.

#### Blocks known as squeeze-and-excitation (SE):

* + EfficientNetB0 incorporates SE blocks to enhance the model's capacity to capture significant characteristics. These blocks improve model performance by learning to favor useful features and suppress irrelevant ones, which recalibrates the channel responses of features.

#### Bottleneck Blocks for Mobile Reverse:

* + The utilization of mobile inverted bottleneck blocks lowers the network's computing cost even more. These blocks enable effective feature extraction with fewer parameters by using a bottleneck structure with a pointwise convolution followed by a depth-separable convolution.

#### The average fund globally:

* + EfficientNetB0 employs global average pooling, just like previous architectures, to minimize the spatial dimensions of feature maps. The average value for every feature map is determined by this operation, producing a feature vector that condenses the spatial data.

#### Nonlinear Encoder:

* + A linear classifier, usually implemented as a fully connected layer, makes up the final layer of EfficientNetB0. The extracted features are combined in this layer to create the final output logits for the classification tasks.

#### Breaking the connection:

In EfficientNetB0, dropout is utilized as a regularization approach to prevent overfitting. During training, it randomly removes some neurons from the network, pushing the model to pick up more reliable and broadly applicable properties.

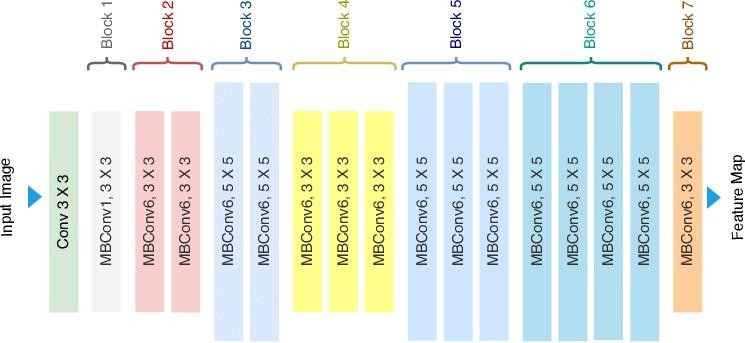


Figure 7. 7 Overview of EfficientNetB0

* + 1. **YOLOV3 ARCHITETCURE**

The following are the main parts of the YOLOv3 architecture:

#### Processing of input:

* The input image is divided into a grid of cells by YOLOv3.
* The task assigned to each cell is to forecast a specific number of bounding boxes and the corresponding class probabilities.
* Normally, the input image is adjusted to fit the network architecture by setting a preset size.

#### The neural network:

* + - * + The backbone network of YOLOv3 is a deep convolutional neural network (CNN), which gathers features from the input image and uses them for object detection.
        + The foundation of YOLOv3 is a redesigned Darknet-53, a deep CNN architecture made up of pooling and convolutional layers.

#### Head of detection:

The detection head, which consists of a sequence of convolutional layers followed by fully connected layers, is in charge of predicting bounding boxes and class probabilities for objects in each grid cell.

* + - * + Bounding boxes are predicted by YOLOv3 at three distinct scales, which correspond to three distinct output layers.
        + Bounding boxes of varying sizes covering a broad variety of item scales are predicted by each output layer.
        + The width, height, confidence score, class probabilities, and the (x, y) coordinates of the box center are all included in each bounding box prediction.

#### NMS, or Non-Maximum Suppression:

* + - * + After predicting bounding boxes for objects, NMS removes overlapping boxes with lower confidence ratings and retains just the most trustworthy ones. YOLOv3 uses NMS to eliminate redundant bounding box predictions.

#### Anchor boxes:

To increase bounds prediction accuracy, YOLOv3 makes use of anchor boxes, which are prefabricated boxes that come in different sizes and shapes.

* + - * + To assist the network learn to anticipate bounding boxes with greater accuracy, YOLOv3 modifies the anchor boxes' dimensions during training by taking into account the ground truth bounding boxes in the training data.

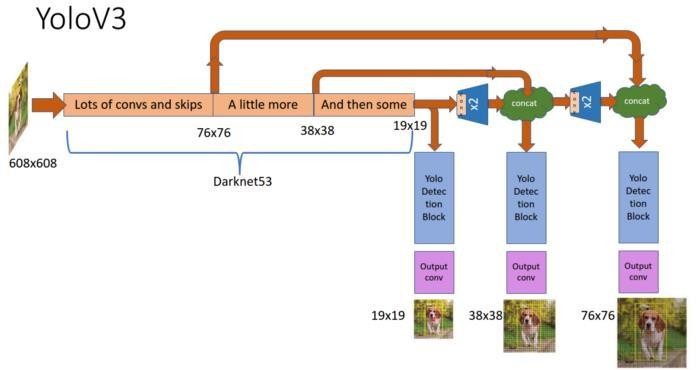


Figure 7. 8 Overview of YOLOV3

**CHAPTER 8**

**RESULT AND DISCUSSION**

* 1. **METRICS PERFORMANCES OF EACH MODELS**
     1. **CUSTOM CNN**

Accuracy:

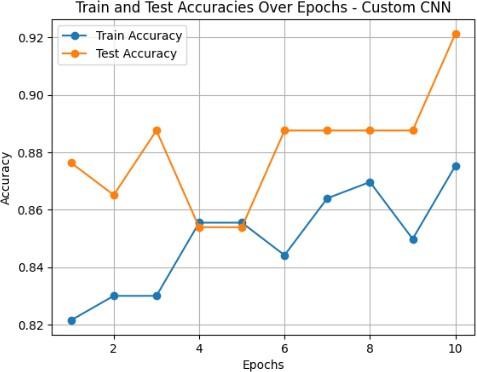


Fig 8.1 Graph of Train and Test Accuracy Over Epoch Confusion Matrix:

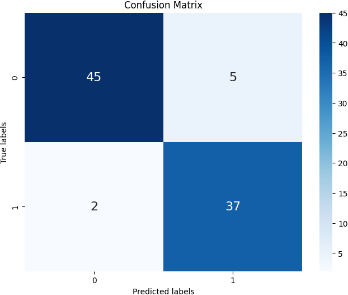


Fig 8.2 Confusion Matrix for Custom CNN

ROC-AUC Curve:

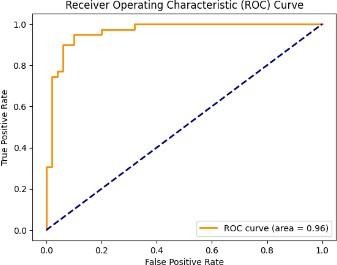


Fig 8.3 ROC-AUC Curve for Custom CNN Residual Plot:

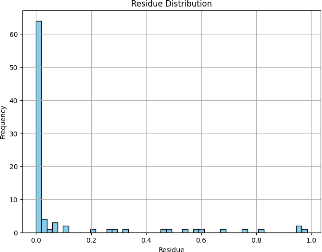


Fig 8.4 Residual Plot for Custom CNN

## ResNet50 MODEL

Accuracy:

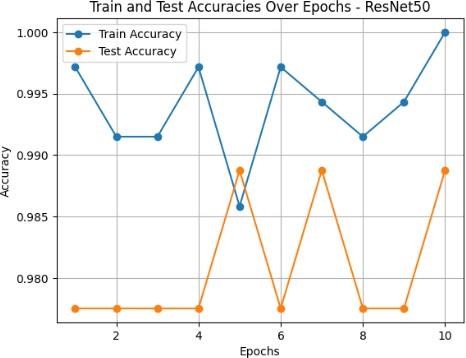


Fig 8.5 Test and Train Accuracy plot for ResNet50 Confusion Matrix:

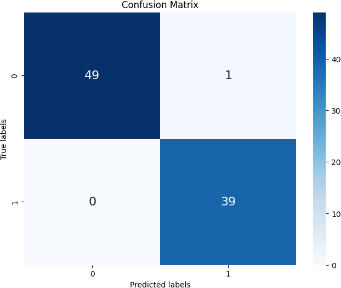


Fig 8.6 Confusion Matrix for ResNet50

ROC -AUC Curve:

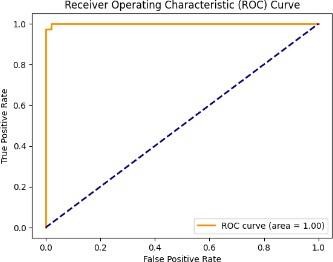


Fig 8.7 ROC-AUC Curve for ResNet50 Residual Plot:

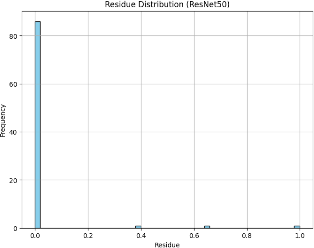


Fig 8.8 Residual Plot for ResNet50

* + 1. **VGG16 MODEL**

Accuracy:

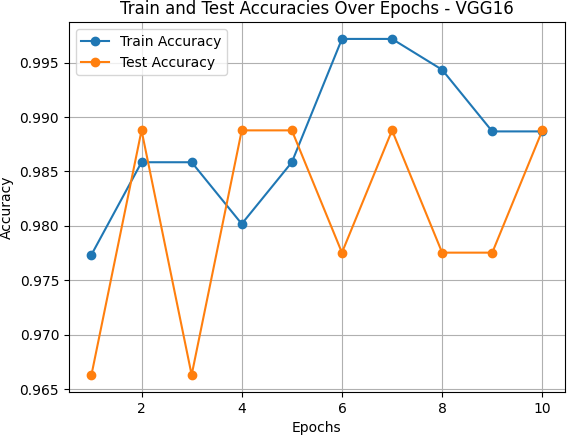


Fig 8.9 Test and Train Accuracy plot for VGG16 Confusion Matrix:

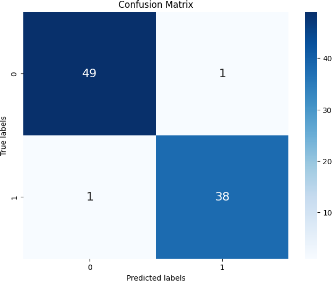


Fig 8.10 Confusion Matrix for VGG16

ROC-AUC Curve:

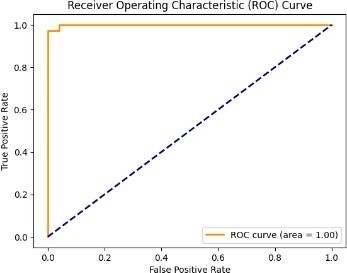


Fig 8.11 ROC-AUC Curve for VGG16 Residual Plot:

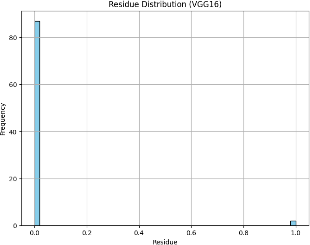


Fig 8.12 Residual Plot for VGG16

## InceptionV3 MODEL

Accuracy:

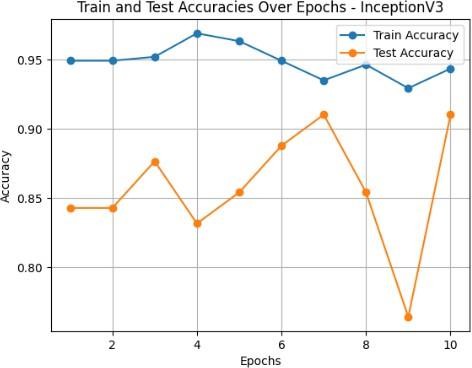


Fig 8.13 Test and Train Accuracy plot for InceptionV3 Confusion Matrix:



Fig 8.14 Confusion Matrix for VGG16

ROC-AUC Curve:

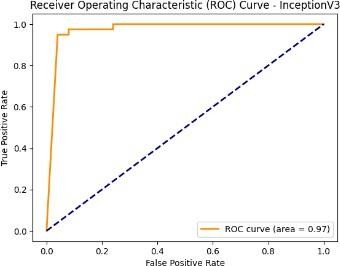


Fig 8.15 ROC-AUC Curve for VGG16 Residual Plot:

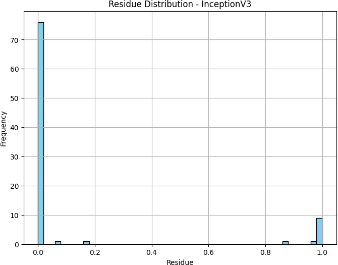


Fig 8.16 Residual Plot for VGG16

## EfficientNetB0 MODEL

Accuracy:

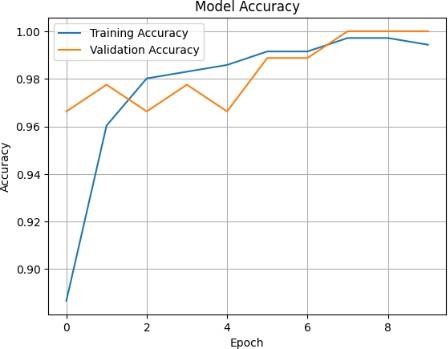


Fig 8.17 Test and Train Accuracy plot for EfficientNetB0 Confusion Matrix:

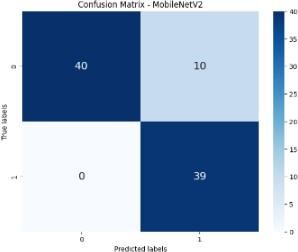


Fig 8.18 Confusion Matrix for EfficientNetB0

ROC-AUC Curve:

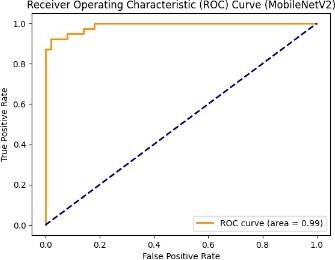


Fig 8.19 ROC-AUC Curve for EfficientNetB0 Residual Plot:

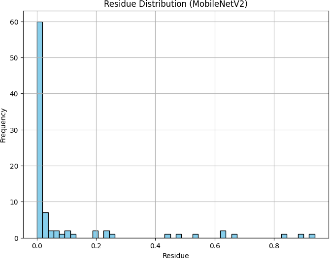


Fig 8.20 Residual Plot for EfficientNetB0

## MobileNetV2 MODEL

Accuracy:

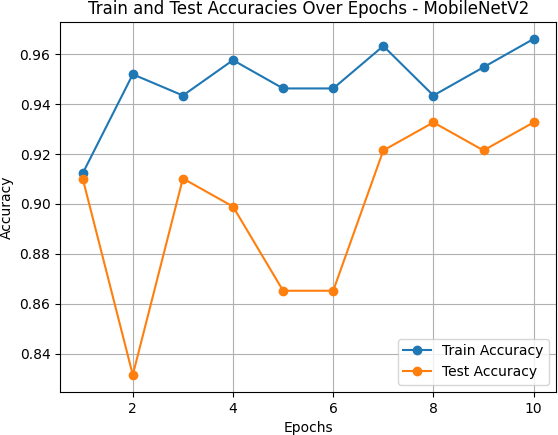


Fig 8.21 Test and Train Accuracy plot for MobileNetV2 Confusion Matrix:

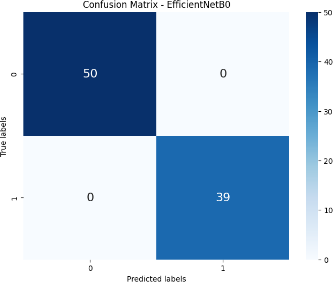


Fig 8.22 Confusion Matrix for MobileNetV2

ROC-AUC Curve:

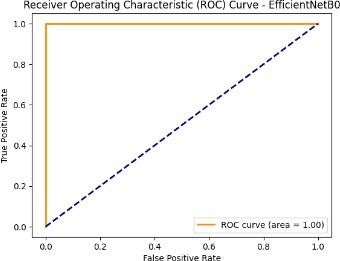


Fig 8.23 ROC-AUC Curve for MobileNetV2 Residual Plot:

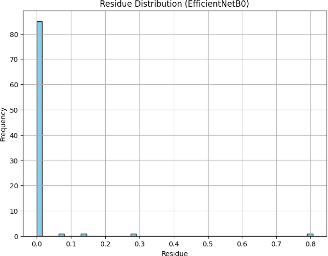


Fig 8.24 Residual Plot for MobileNetV2

## YoloV3 MODEL

Accuracy:

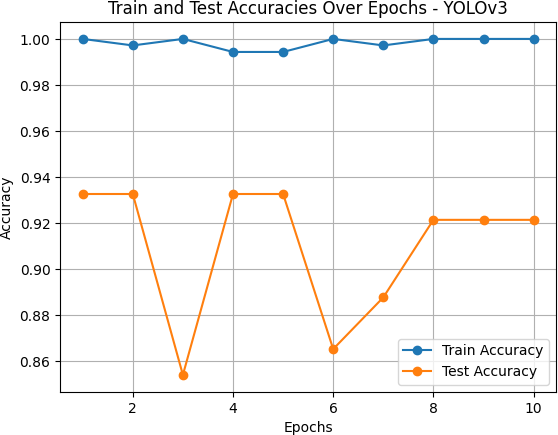


Fig 8.25 Test and Train Accuracy plot for YoloV3 Confusion Matrix:

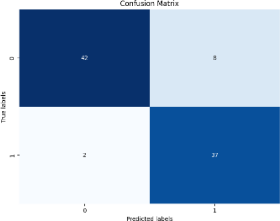


Fig 8.26 Confusion Matrix for YoloV3

ROC-AUC Curve:

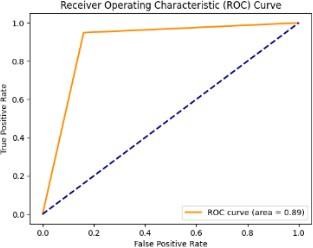


Fig 8.27 ROC-AUC Curve for YoloV3 Residual Plot:

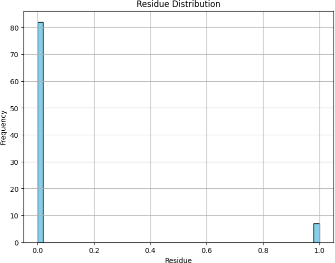


Fig 8.28 Residual Plot for YoloV3

* 1. **COMPARISON OF RESULTS**

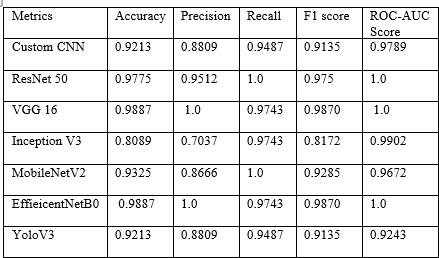
****

Fig 8.29 Other kind of Metrics Comparison for 7 Models

Overall, based on the consistently high scores across multiple evaluation metrics, ResNet50 and EfficientNetB0 appear to be the optimal models for the task at hand, with VGG 16 also performing reasonably well.

**CHAPTER-9**

**CONCLUSION AND FUTURE ENHANCEMENT**

* 1. **CONCLUSION**

This work presents a thorough investigation of several deep learning models for accurate identification and categorization of growth stages of tomato plants. To automate the monitoring of tomato plant development, the proposed tomato plant detection system uses state-of-the-art architectures such as Custom CNN, ResNet50, VGG16, InceptionV3, EfficientNetB0, and MobileNetV2. The Tamil Nadu Agricultural University (TNAU) fields provided a customized data set that was used to train the system. This dataset included a range of plant morphologies and environmental variables. ResNet50 and EfficientNetB0 were the two best-performing models evaluated; achieved exceptional accuracy and near-flawless scores in a number of evaluation criteria such as precision, recall, F1, and ROC-AUC. Their superior performance is explained by their sophisticated architectural arrangement and efficient feature extraction techniques.

All things considered, this project highlights the critical role deep learning approaches play in developing data-driven and sustainable agricultural practices, opening the door to a future where innovation and technology combine to improve environmental sustainability and food security.

* 1. **FUTURE ENHANCEMENT**

1. **Expand the dataset:** While the current dataset from Tamil Nadu Agricultural University (TNAU) covers a wide range of plant morphologies and environmental conditions, further expanding the dataset with more diverse samples from different regions and climatic conditions can improve the robustness and generalization ability of the models.
2. **Incorporate temporal data:** The current approach focuses on classifying the growth stages based on individual images. Incorporating temporal data, such as time-series images or videos, could enable tracking the growth progression of individual plants over time, providing valuable insights into growth patterns and dynamics.
3. **Explore multi-task learning:** In addition to growth stage classification, the models could be extended to perform multi-task learning, where they simultaneously predict other relevant factors such as yield estimation, disease detection, or environmental stress indicators. This approach could lead to a more comprehensive crop monitoring system.
4. **Deploy in the field:** While the models have been evaluated on the dataset, deploying the system in real-world field conditions could uncover additional challenges and opportunities for improvement. Field testing and iterative refinement could enhance the system's robustness and practical applicability.
5. **Integration with Internet of Things (IoT) devices:** Combining deep learning models with IoT devices such as cameras and sensors could enable real-time monitoring and automated data collection of tomato fields. This integration could facilitate large-scale deployment and continuous acquisition of data to update and refine the model.
6. **Explore explainable AI techniques:** As deep learning models can often be seen as “black boxes,” exploring explainable AI techniques could provide insight into the models' decision-making process. This could help researchers and agronomists better understand the factors contributing to the classification of growth stages and potentially reveal new insights about the domains.
7. **Collaborate with domain experts:** Involving domain experts, such as agronomists and horticulturists, throughout the project can provide valuable feedback and guidance. Their expertise can help identify potential limitations, suggest improvements, and ensure the practicalrelevance and acceptance of the system in the agricultural community.

**SNAPSHOTS OF DATASET: STAGE 1:**





****



****

**STAGE 2:**

****



****

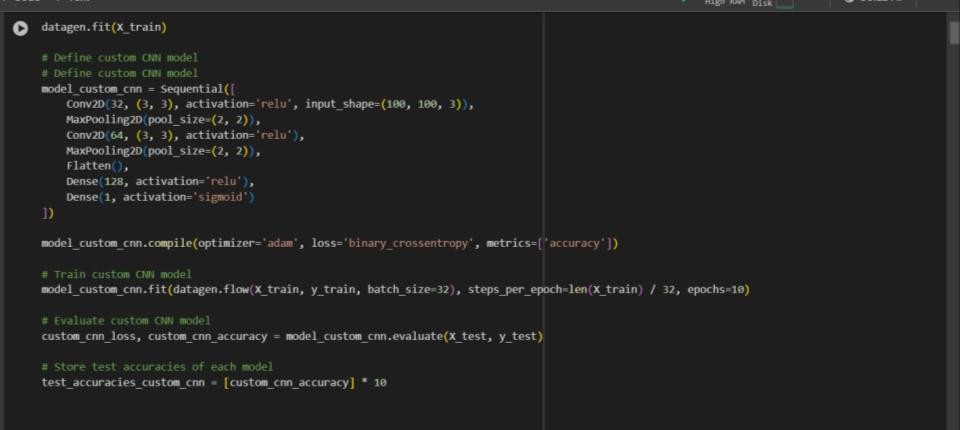
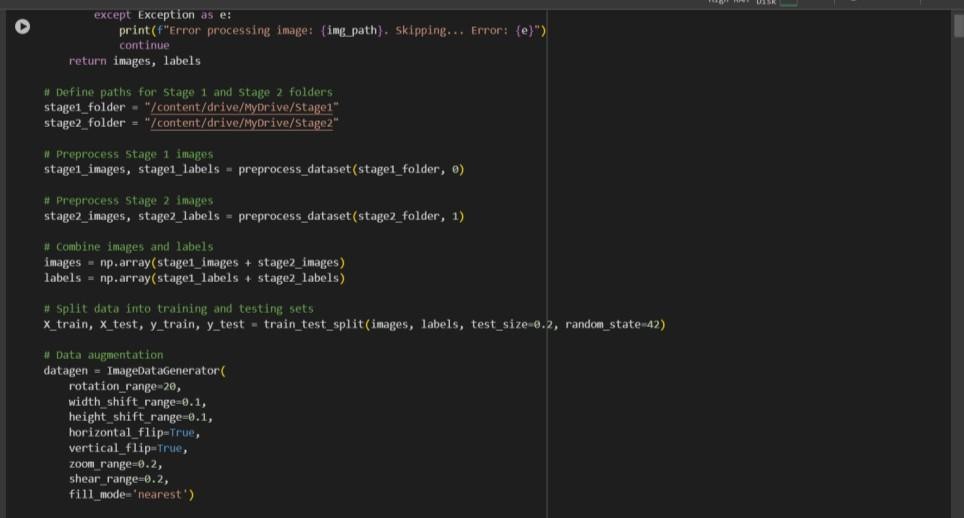
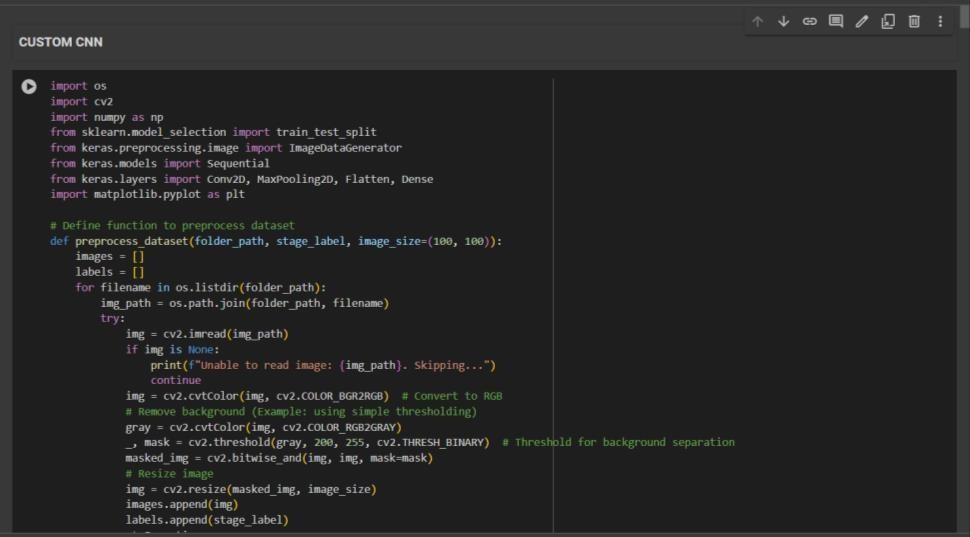


****

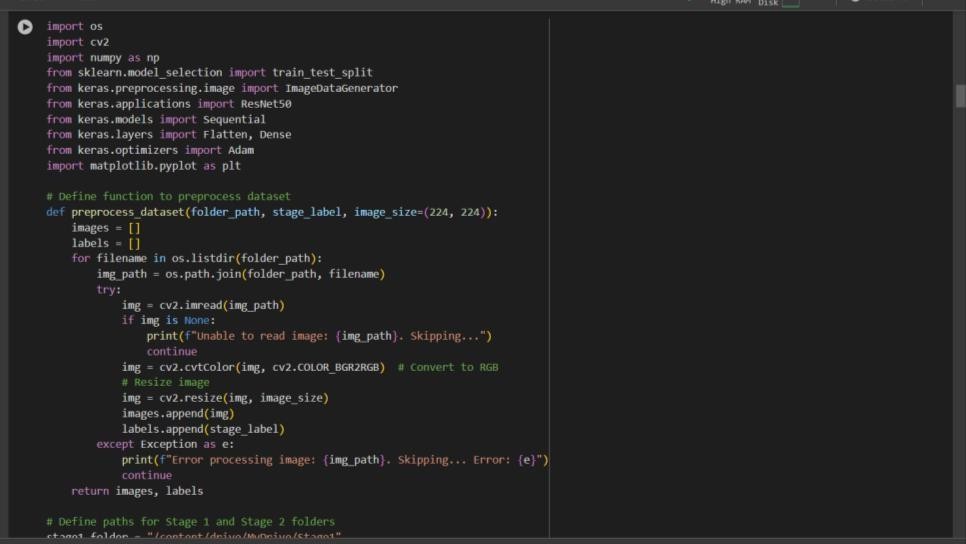
**SNAPSHOTS OF SOURCE CODE FOR ALL THE 7 MODELS:**

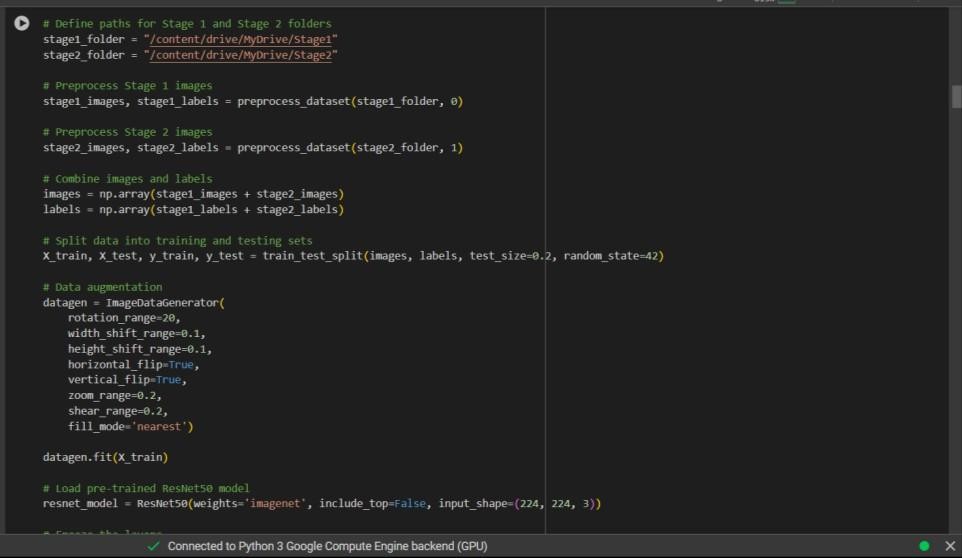
Custom CNN:

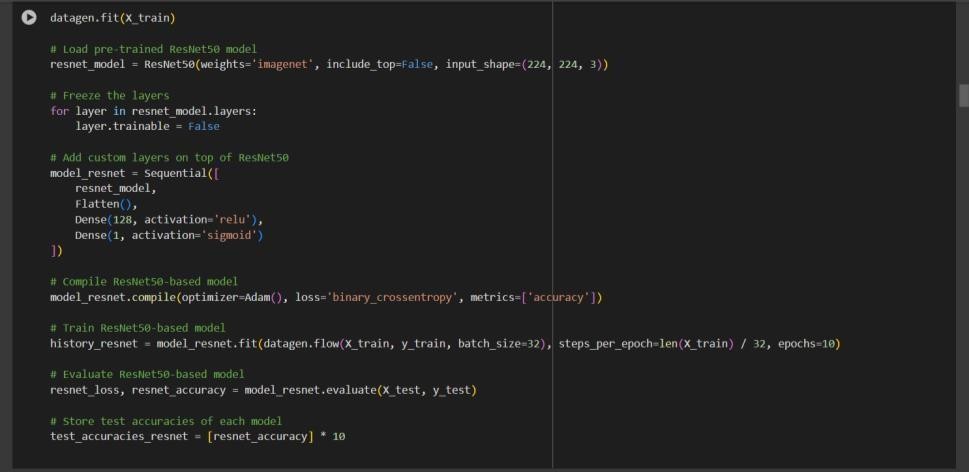




ResNet50 Model:

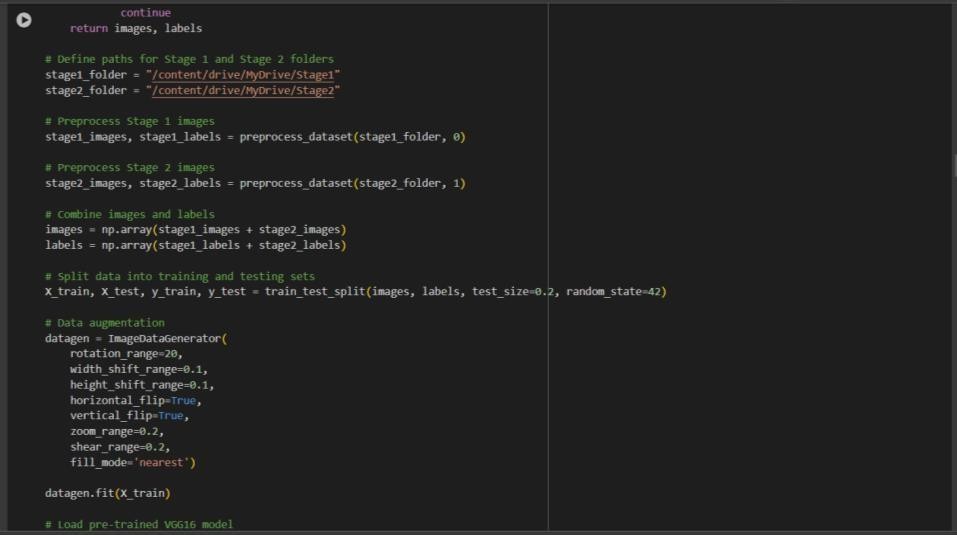
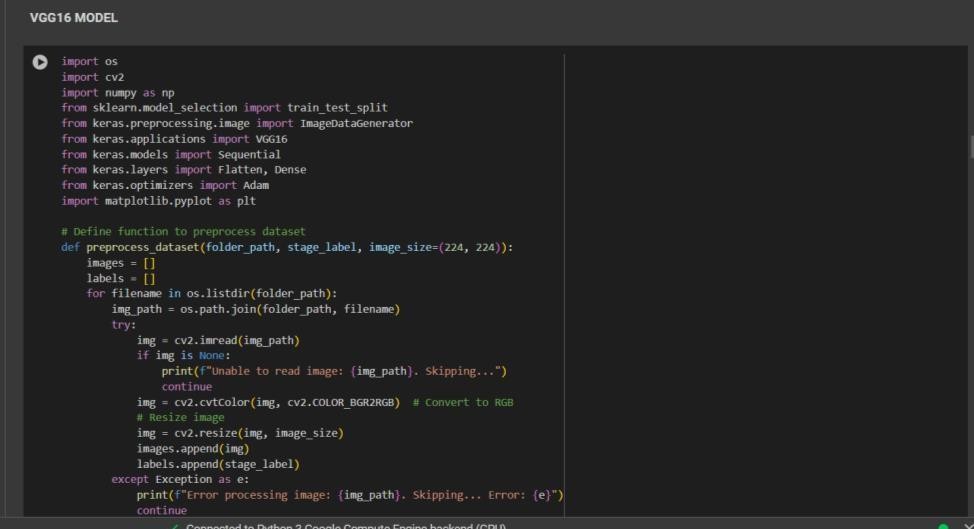


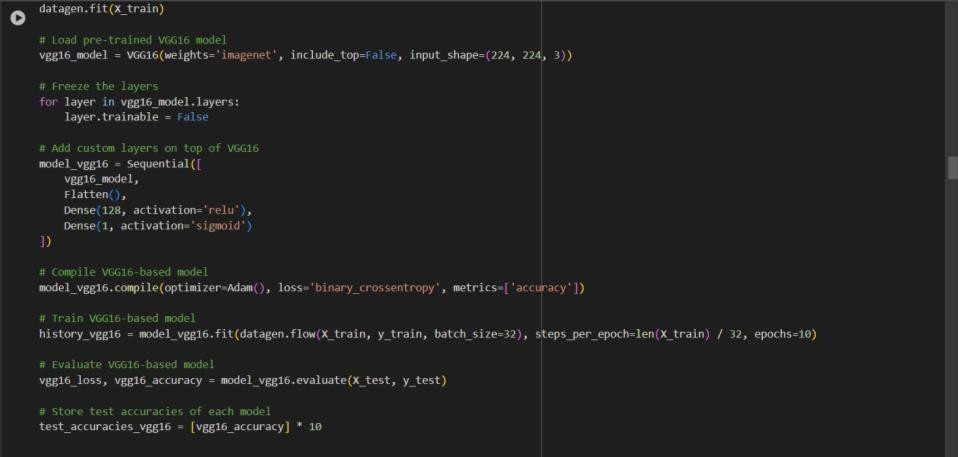






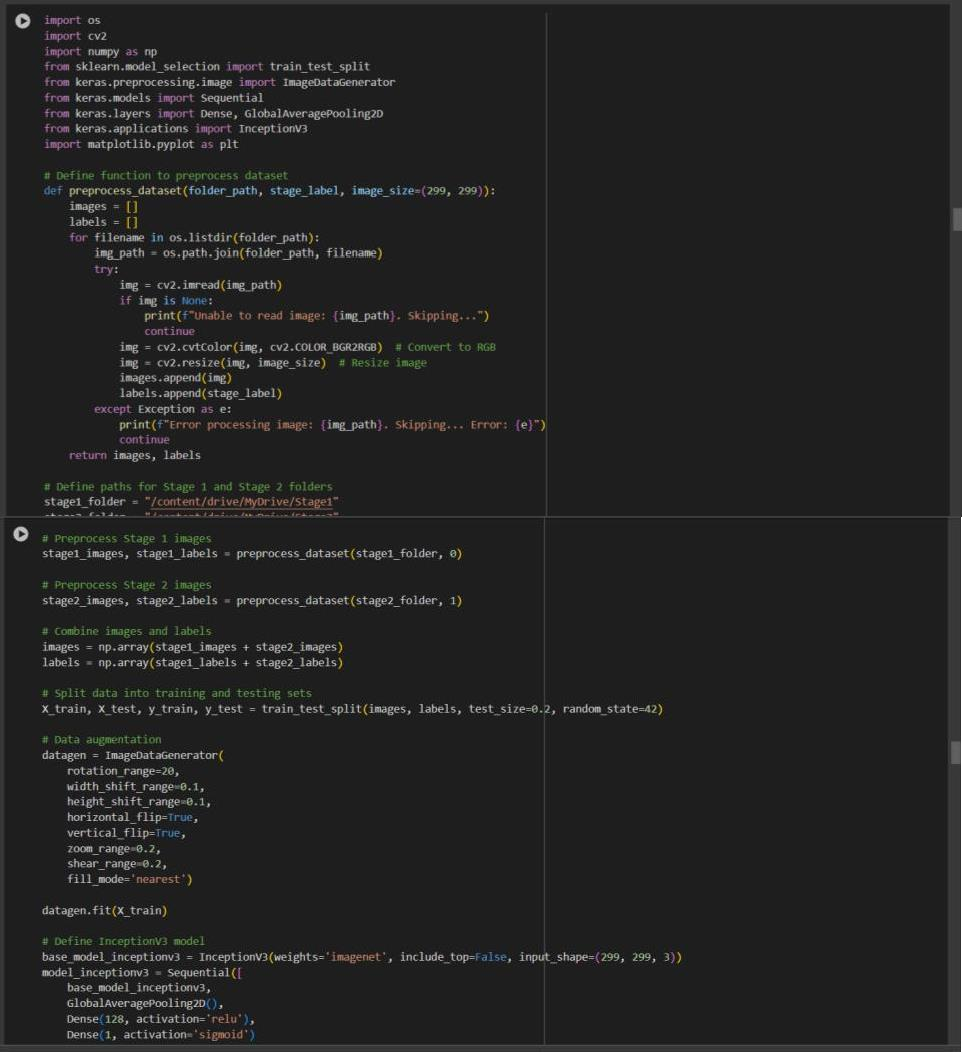
VGG16 Model

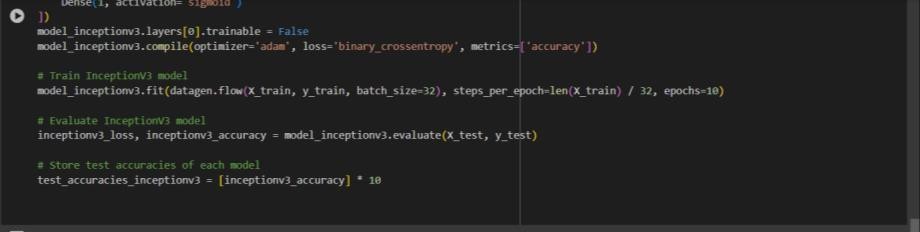


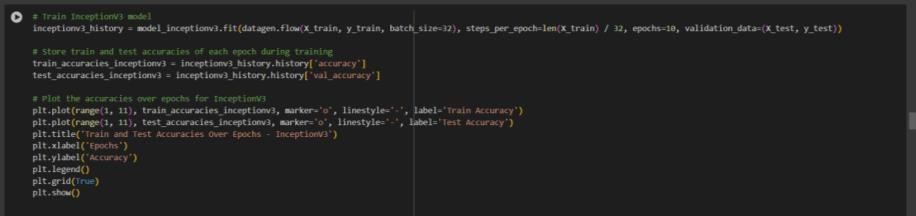




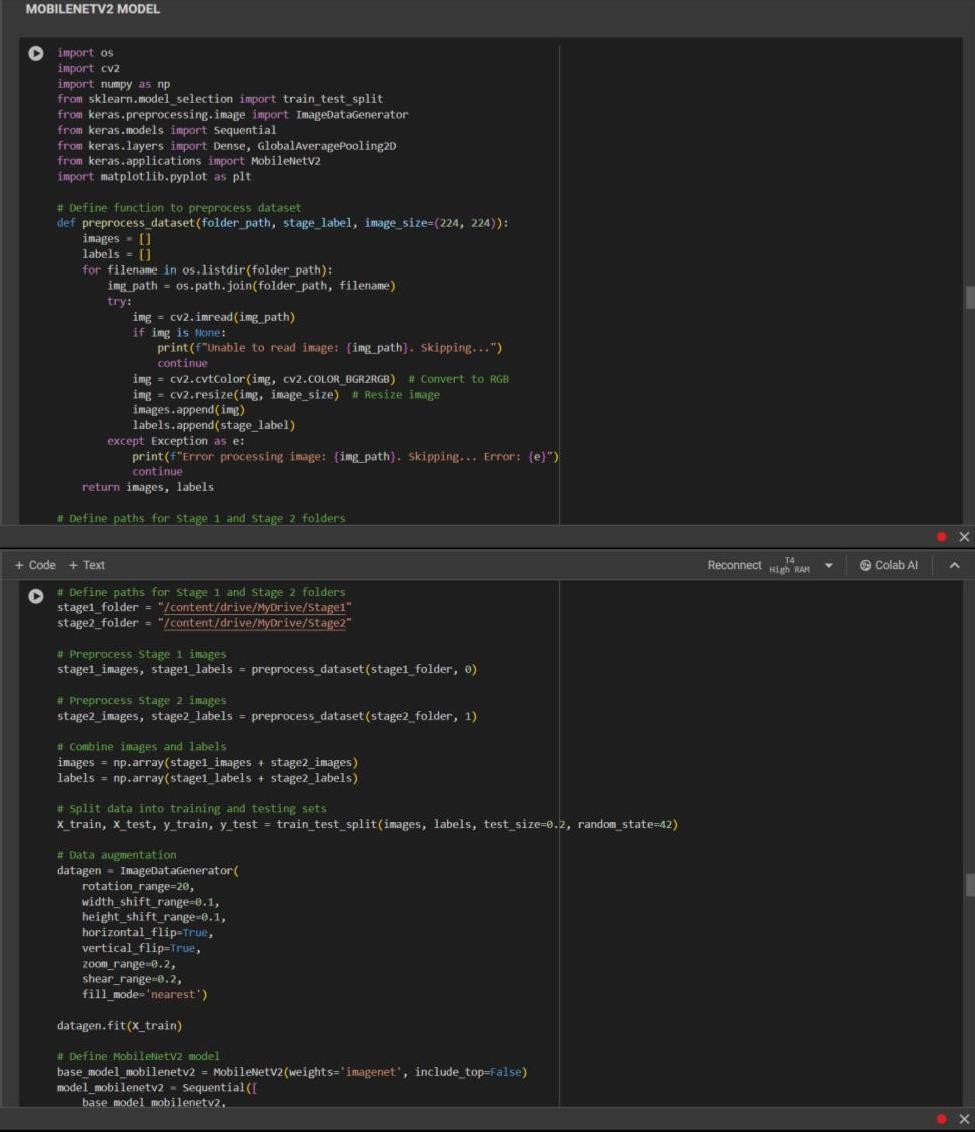
InceptionV3:

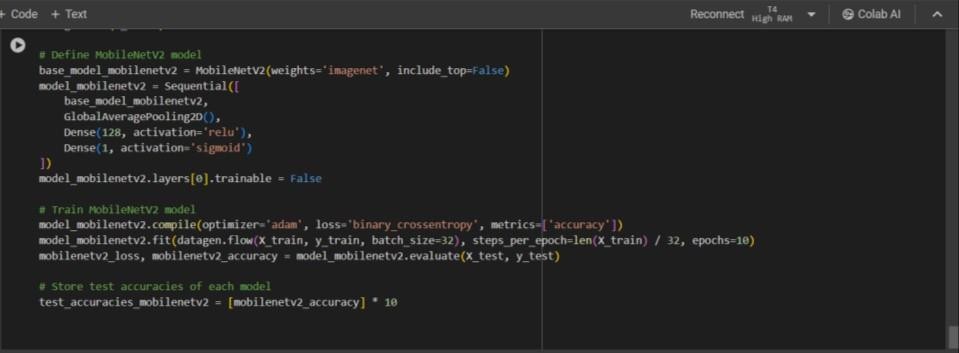






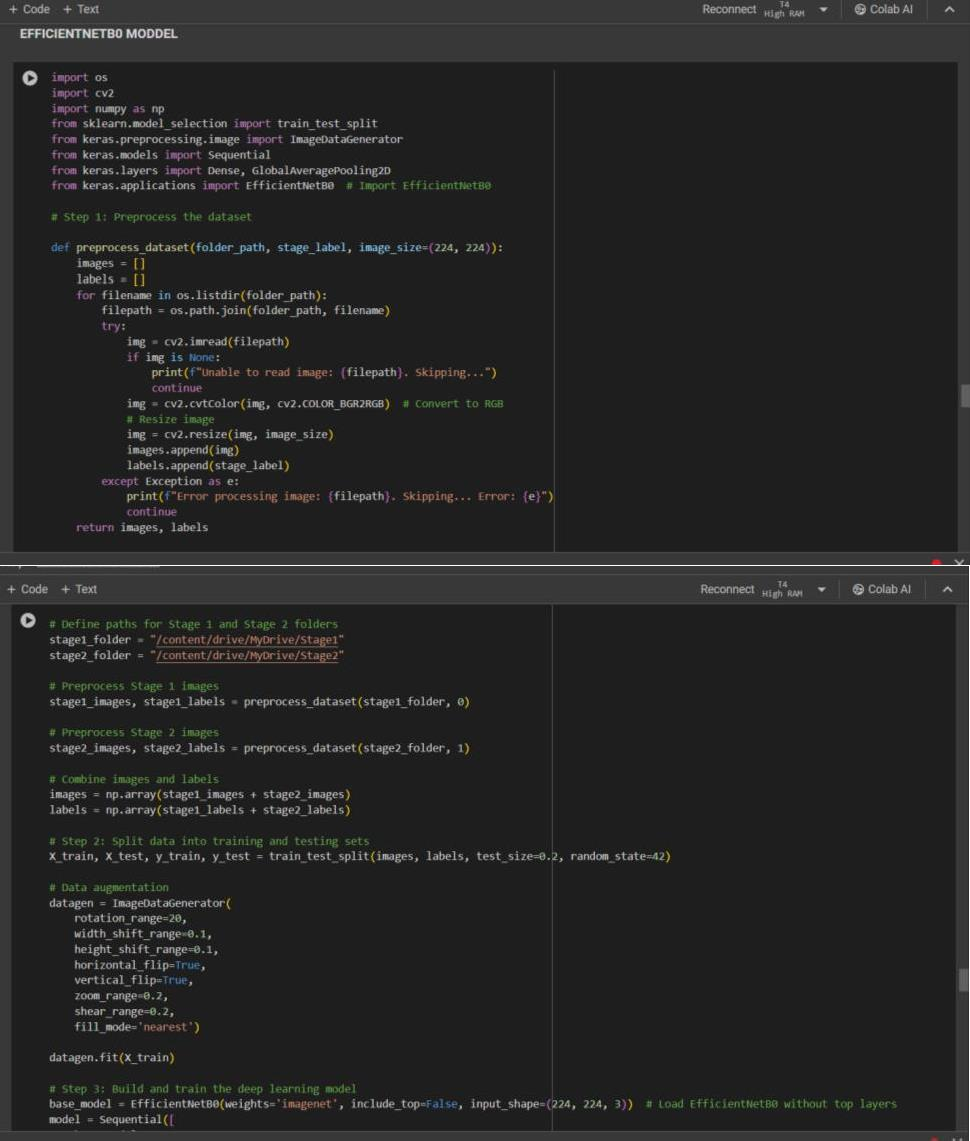
MobileNetV2:

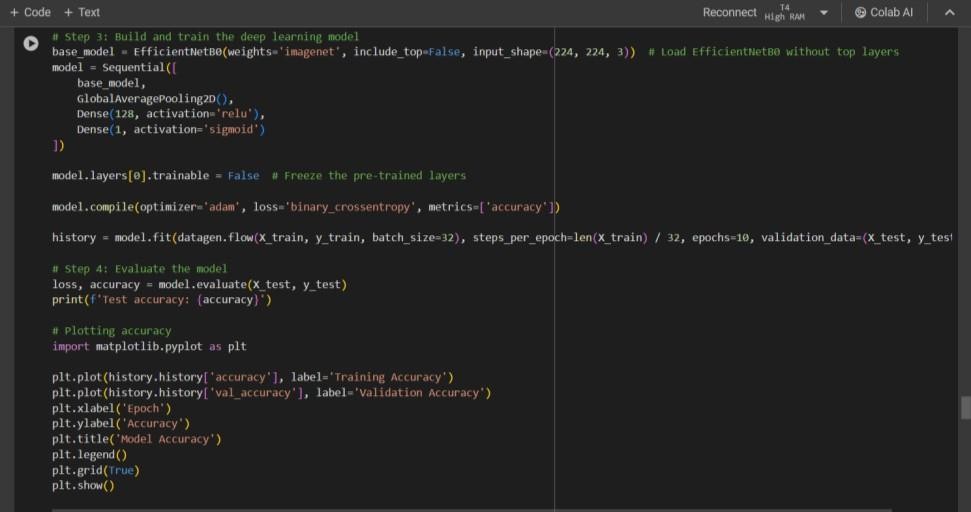




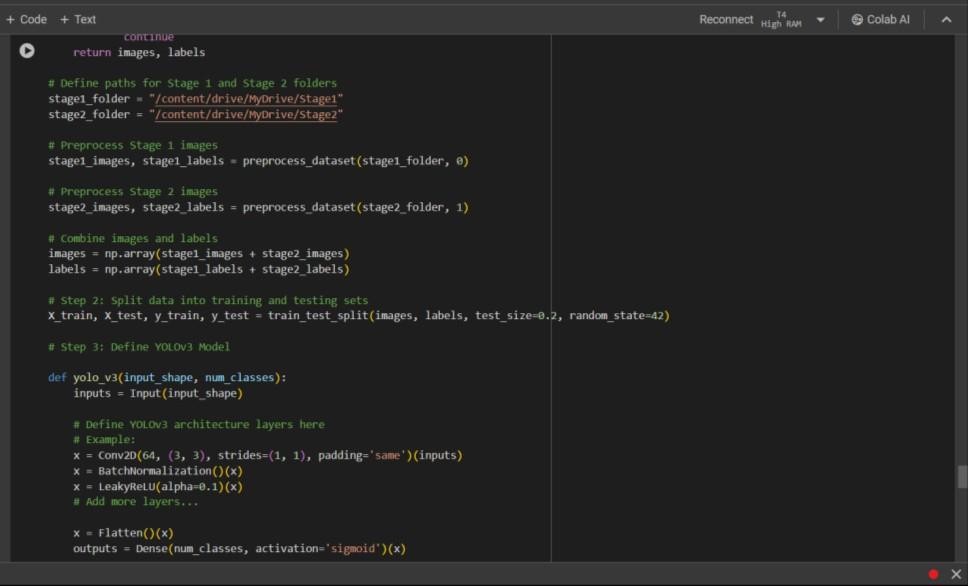
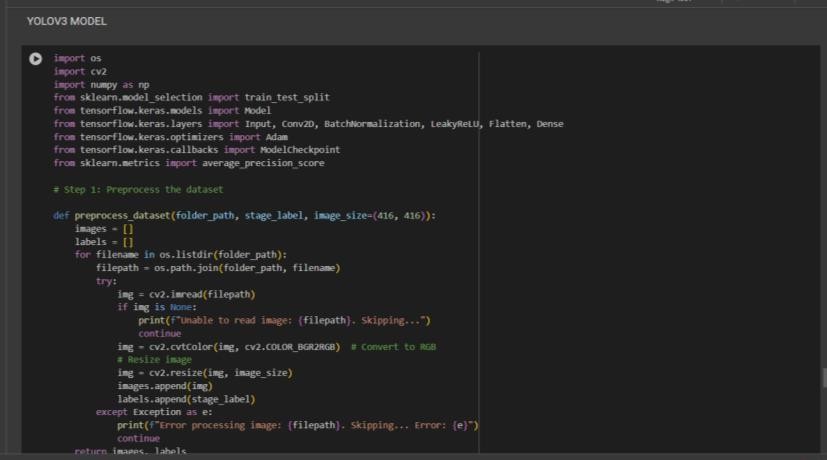


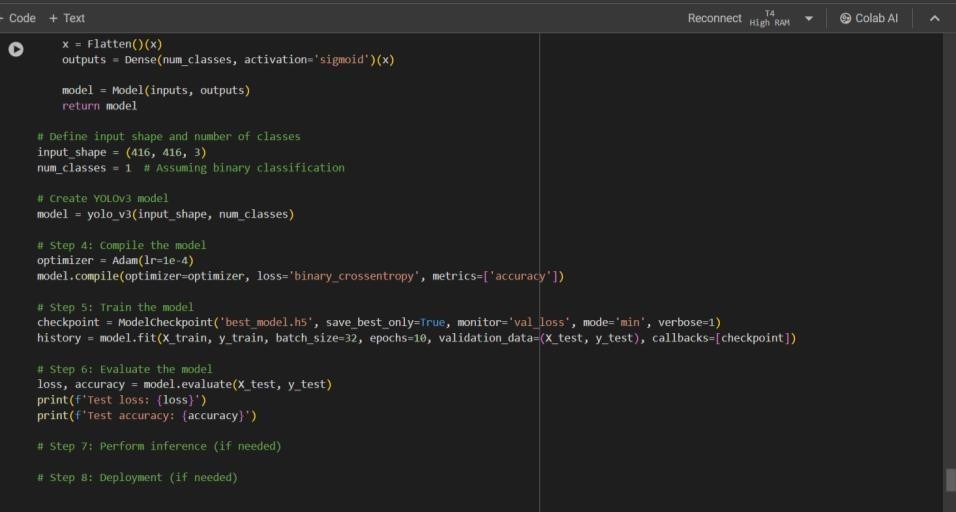
EfficientNetB0:





YoloV3 Model:





**REFERENCES**

1. Burkapalli, V. C., & Patil, P. C. (2020). TRANSFER LEARNING: INCEPTION-V3 BASED CUSTOM CLASSIFICATION APPROACH FOR FOOD IMAGES. ICTACT Journal on Image

& Video Processing, 11(1).

1. Balasubramanian, M., Sowmya, M., and Vaidehi, K. (2023). Using VGG16 and ResNet50, a 2D-convolutional neural network, human behavior is classified. 1221–1229 in Indian Journal of Science and Technology, 16(16).
2. Jana & Biswas (A. P., 2018; May). Detection and classification of objects in video recordings using a Yolo approach. The 3rd IEEE International Conference on Recent Trends in Information and Communication Technology (RTEICT) was held in 2018 and featured papers on pages 2448- 2452. IEEE.
3. Hu, Q., Lai, J., Zhang, G., Li, R., Xiang, Q., & Wang, X. (2019, October). Fruit image classification using transfer learning approach based on Mobilenetv2. pp. 1–7 in Proceedings of the Third International Conference on Application Engineering and Computer Science.
4. In Tammina (2019), S. For picture classification, transfer learning is employed with a deep convolutional neural network and vgg-16. Journal of Scientific and Research Publications (IJSRP): International Journal, 9(10), 143-150.
5. Huang (2023), Su (2023), Wu (2023), & Chen (2023). Triplet attention mechanism and efficientnet are used for the classification of rock images. 13(5) Applied Sciences, 3180
6. Rokhana, R., Herulambang, W., & Indraswari, R. (2022). categorization of melanoma images using the MobileNetV2 network. Computer Science Procedia, 197, 198-207.
7. Shahi, T. B., Guo, W., Sitaula, C., & Neupane, A. (2022). Fruit classification for industrial applications with attention-based MobileNetV2. 17(2), e0264586; Plos One.
8. M. M. Krishna, M. V. G. Rao, & Neelima, M., Harshali, M. (2018). Deep learning for image classification. 7(2.7), 614-617, International Journal of Engineering & Technology
9. Alam, I. N., Wicaksono, P., & Kartowisastro, I. H. (2022). For a facial expression recognition system, use the transfer learning technique with EfficientNet. The Artificial Intelligence Review, 36(4).