

FRESH-MEAT INSPECTOR
Meat Inspection Using DenseNet121

A PROJECT REPORT

Submitted by

SARWAN REDDY C [RA2111047010139]

ABHINAY M [RA2111047010134]

Under the Guidance of

DR. S.P. ANGELIN CLARET

Assistant Professor,

Department of Computational Intelligence

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE



DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR- 603 203

NOVEMBER 2024



Department of Computational Intelligence
SRM Institute of Science & Technology
Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course : **B. TECH / ARTIFICIAL INTELLIGENCE**

Student Name : **SARWAN REDDY C, ABHINAY M**

Registration Number : **RA2111047010139, RA2111047010134**

Title of Work : **FRESH-MEAT INSPECTOR (USING DENSENET121)**

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g.fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

If you are working in a group, please write your registration numbers and sign with the date for every student in your group.



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 18AIP107L- Minor Project report titled **“FRESH-MEAT INSPECTOR (USING DENSENET121)”** is the bonafide work of **“SARWAN REDDY C [RA2111047010139], ABHINAY M [RA2111047010134]”** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

DR. S.P.ANGELIN CLARET

SUPERVISOR

ASSISTANT PROFESSOR

DEPARTMENT OF

COMPUTATIONAL INTELLIGENCE

SIGNATURE

DR. R. ANNIE UTHRA

PROFESSOR & HEAD

DEPARTMENT OF

COMPUTATIONAL INTELLIGENCE

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. T. V. Gopal**, Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We encompass our sincere thanks to, **Dr. M. Pushpalatha**, Professor and Associate Chairperson, School of Computing and **Dr. C. Lakshmi**, Professor and Associate Chairperson, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our Head of the Department, **Dr. R. Annie Uthra**, Professor, Department of Computational Intelligence, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinators, **Dr. A. Alice Nithya**, Panel Head, and **Dr. C. Sherin Shibi**, **Dr. Angelin Claret. S.P**, **Dr. B. Pitchaimanickam**, **Dr. Illakiya T and Mrs. Aarthi K**, Panel Members, Department of Computational Intelligence, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr B. Pitchaimanickam**, Department of Computational Intelligence, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. S.P. ANGELIN CLARET**, Department of Computational Intelligence, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his / her mentorship. He / She provided us with the freedom and support to explore the research topics of our interest. His / Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of Computational Intelligence, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement.

Authors

ABSTRACT

In the Fresh-Meat Inspector project, automated solution for the assessment of freshness in non-vegetarian food, focusing specifically on meat quality. It is known that traditional inspection methods are labour-intensive, subjective, and time-consuming; therefore, our approach harnesses the power of deep learning to deliver a more efficient alternative. Utilizing a state-of-the-art convolutional neural network like DenseNet121, a model was constructed that could check the meat freshness based on high resolution images. Three classes were developed and used which include Fresh, Half-Fresh, and Spoiled. This will improve the consumers' safety and quality on the overall product.

However, as an optimization towards the quality of images going into the system a pre-processing pipeline was provided which includes, contrast enhancement, Gaussian blur, histogram equalization as a manner of ensuring to analyses these images at great quality resolution. This new approach revealed that the performance and accuracy in classifying freshness among various types of meat through the DenseNet121-based system reached as high as 99%, establishing new credibility and efficiency standards in evaluating food safety. The end results of this project look for a significantly elevated level above industry standards with scalability through rapid accurate inspections for determining meat quality.

TABLE OF CONTENTS

ABSTRACT	II
TABLE OF CONTENTS	III
LIST OF FIGURES	IV
LIST OF GRAPHS	V
LIST OF TABLES	VI
ABBREVIATIONS	VII

CHAPTER NO	TITLE	PAGE NO
1	INTRODUCTION	1
	1.1 General (Introduction to Project)	1
	1.2 Motivation	2
	1.3 Sustainable Development Goal of the Project	3
2	LITERATURE SURVEY	4
	2.1 Literature Survey	4
	2.2 Limitations Identified from Literature Survey	5
	2.3 Research Objectives	8
	2.4 Product Backlog	8
	2.5 Plan of Action	10
3	SPRINT PLANNING AND EXECUTION	13
	METHODOLOGY	13
	3.1 SPRINT I	15
	3.2 SPRINT II	17
	3.3 SPRINT III	
4	RESULTS AND DISCUSSIONS	19
	4.1 Project Outcomes	19
5	CONCLUSION AND FUTURE ENHANCEMENT	28
	REFERENCES	29
	APPENDIX	
A	CODING	31
B	CONFERENCE PUBLICATION	45
C	JOURNAL PUBLICATION	46
D	PLAGIARISM REPORT	47

LIST OF FIGURES

CHAPTER NO.	TITLE	PAGE NO.
3.1	Architecture Diagram of the Automated Meat Quality Check	14
3.2	Architecture Diagram of the Automated Meat Quality Check through Model.....	16
3.3	Architecture Diagram of the Automated Meat Quality Check with Flask Library	18
4.1	Fresh's Outcome.....	19
4.2	Half Fresh's Outcome.....	20
4.3	Spoiled Outcome.....	21
4.4	DenseNet121 Architecture.....	23
4.5	Confusion Matrix of Model.....	27

LIST OF GRAPHS

CHAPTER NO.	TITLE	PAGE NO.
4.1	Accuracy Graph	24
4.2	Loss Graph	25
4.3	AUC Graph	26

LIST OF TABLES

CHAPTER NO.	TITLE	PAGE NO.
4.1	Training and Validation per Epoch	23

ABBREVIATIONS

CCN	Convolutional Neural Network
AI	Artificial Intelligence
SDG	Sustainable Development Goal
CLAHE	Contrast Limited Adaptive Histogram Equalization
AUC	Area Under Curve
API	Application Programming Interface
UI	User Interface
UX	User Experience
ML	Machine Learning
MVP	Minimum Viable Product
DB	Database
PY	Python
HTML	Hyper-Text Markup Language
CSS	Cascading Style Sheets
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure

CHAPTER 1

INTRODUCTION

1.1 General (Introduction to Project)

The Fresh-Meat Inspector is one of the advanced systems, using computer vision and deep learning, such as the model DenseNet121, designed for the estimation of quality in meat. This is a project that falls in line with a larger tendency of food quality technologies that move beyond earlier studies and applications to include machine vision and neural networks and portable, non-invasive quality evaluation tools [1]. In light of the extreme importance of food safety and consumer confidence, quality control systems are the need of the day in the meat production and retail sectors, where conventional inspection methods are often found to be inconsistent and not very efficient [3].

The Fresh-Meat Inspector works on the principle of capturing high-quality images of raw meat and processing the images using advanced algorithms to enhance the clarity and contrast, hence allowing more accurate feature extraction. This pre-processing stage adapts the methods CLAHE, Gaussian Blur, and Histogram Equalization for raw meat. Then DenseNet121 architecture is adapted for analysis related to color, texture, and freshness level. Hence, it classifies into fresh, half-fresh, and spoiled, which directly deals with the methods involved with pork and chicken meat [4].

The system is designed for real-time monitoring, with features such as alerts and data visualization that ensure timely action should spoilage be detected. This solution is particularly beneficial to retailers, food inspectors, and consumers, where instant feedback is provided that supports decision-making in relation to reducing health risks connected with meat spoilage [6]. The Fresh-Meat Inspector would foster safe food and act as a foundation for additional invention of non-destructive testing through other food lines; consequently, it will provide even stronger standards of high food quality and consumer confidence within the food industry.

1.2 Motivation

The motivation behind the Fresh-Meat Inspector project mainly comes from increasing demands to improve food safety and quality assurance in the meat industry. Over the past decade, concerns about foodborne illnesses and contamination have increased, and it has thus become pertinent for producers and retailers to ensure that meat products are fresh, safe, and of high quality. Traditional inspection methods, relying largely on human judgment, often seem inconsistent and incomplete within fast-moving food processing environments.

There are several critical factors that necessitated starting this project:

Food Safety and Consumer Confidence Improvement: Food consumers expect safe, quality products and proper rejection in case of failings. In case a firm fails to provide safe-quality food products, the health impact results in reputational damage. The solution from this project allows the real-time determination of the freshness of meat and any possible spoilage for better confidence in products by consumers.

Increasing Inspection Efficiency: High demand with limited resources food inspectors and quality control teams has. This is where the Fresh-Meat Inspector comes in, computer vision along with deep learning to automate inspection which enables rapid, non-destructive quality assessments that enhance operational efficiency and reduce the chance of human error.

Leveraging Advanced Deep Learning Techniques: Advanced deep learning and image analysis technologies, for example, DenseNet¹²¹, open new avenues for the construction of flexible and scalable food inspection systems. In this project, such technologies were applied to build a model that is capable of producing appropriate quality classification in terms of freshness on meat products and brings progress toward developing more complex quality assurance solutions.

The objective behind the project is to give a holistic view of meat quality beyond just simplicity, freshness. This is through color and texture analysis, leading to more data-based decisions and constant improvement towards better food industry standards.

1.3 Sustainable Development Goal of the Project

Our project aligns with the Sustainable Development Goals (SDGs), specifically contributing to:

SDG 3: Good Health and Well-Being

Fresh Meat Inspector can push through the Sustainable Development Goals 3 in its activity, focusing on the quality aspect regarding food safety because the aforementioned factor is closely connected with people's health and wellbeing.

Enhance Food Safety and Minimize Health Risks: The Fresh-Meat Inspector ensures that meat products are fresh and free from contamination, thus playing a very crucial role in reducing foodborne illnesses. Real-time detection of meat spoilage prevents the consumption of unsafe products, thus supporting public health and reducing the burden on healthcare systems associated with food-related diseases.

Strengthen Food Inspection and Quality Control Systems: The application of modern technologies like DenseNet121 and machine vision helps update the system to inspection with quality control without much intrusion, replacing existing approaches for inspection with automated options that help food production establishments in achieving higher efficiency with regard to meeting standards concerning health. This process shall further help in strengthening food safety and quality control as efficiently required throughout the food chain.

Supporting Consumption and Production: It avoids wastage by taking measures to ensure less and proper waste as this prevents the sale and subsequent consumption of unsafe products while helping people have the best available produce from the market due to its accurate quality review mechanism.

The Fresh-Meat Inspector seeks to enhance public health outcomes and contributes to sustainable development by promoting improvements in food safety standards, consumer confidence, and responsible resource use-all important aspects of building a safer and more sustainable food system.

CHAPTER 2

LITERATURE SURVEY

2.1 Literature Survey

In recent years, the application of deep learning to food quality assessment, particularly meat freshness evaluation, has garnered significant attention. Traditional methods of meat freshness inspection, such as sensory evaluation and chemical tests, labor-intensive, are and time-consuming, often subjective. Consequently, automated systems driven by machine learning and deep learning have emerged as promising alternatives for improving the efficiency and accuracy of meat quality assessment.

Alecs, Rovich and Sergeevich (2024) developed an expert diagnostic system that employs neural networks to enhance the quality supervision of meat production. This system leverages the power of artificial intelligence (AI) to provide continuous and consistent monitoring of meat quality, thereby improving safety and reducing human error in manual inspections [1]. Neural networks, in particular, have shown great promise due to their ability to process complex data, such as images, and identify intricate patterns related to meat spoilage. Liu et al. (2021) proposed a portable device for non-destructive testing of livestock meat quality, based on machine vision. Their approach uses image processing techniques to assess meat freshness without damaging the product, making it suitable for real-time applications in industrial settings [2]. This non-invasive method represents a significant improvement over conventional practices, which often involve destructive sampling. Machine vision combined with deep learning has emerged as a key technology in food quality assessment, allowing for rapid and objective evaluations. Ulucan, Karakaya, and Turkan (2019) explored the potential of deep learning for meat quality assessment. They demonstrated that convolutional neural networks (CNNs) are effective in classifying meat into different freshness categories based on visual characteristics, such as colour and texture [3].

Their work highlighted the role of CNNs in automating the inspection process, which significantly reduces the need for human intervention. Among various CNN architectures, DenseNet121 is particularly suitable for such tasks due to its dense connectivity, which improves feature propagation and enables the model to capture subtle differences in meat quality. Bacus (2021) applied neural networks to the identification of pork meat freshness. By training a model on labelled images of pork meat at different stages of spoilage, Bacus was

able to achieve accurate predictions of freshness levels [4]. The study emphasized the importance of high-quality image datasets in training robust models capable of handling real-world variability in meat appearance. Meza et al. (2020) analysed meat colour changes using computer vision techniques, focusing on the relationship between surface discoloration and spoilage. This research is particularly relevant to meat freshness detection, as colour is one of the primary indicators of quality. The authors used image analysis to track these changes over time, providing a foundation for the development of AI-driven systems that can autonomously monitor meat products throughout their shelf life [5].

Penning, Snelling, and Woodward-Greene (2020) discussed the role of machine learning in the assessment of meat quality, emphasizing its potential to revolutionize the industry. They highlighted how AI can streamline operations, reduce waste, and ensure consistent quality by enabling real-time monitoring of meat freshness at multiple points in the supply chain [6]. Alzaga et al. (2022) took this a step further by developing a machine learning-based system for predicting pork meat quality and estimating shelf-life. Their model not only assesses current freshness levels but also provides an estimate of how long the meat will remain safe for consumption [7]. This forward-looking capability is critical in optimizing inventory management and reducing food waste. Finally, You et al. (2020) introduced a novel method for evaluating chicken meat quality using colour card localization and correction. Their research underscores the importance of accurate colour measurement in freshness assessment and demonstrates the value of integrating advanced image processing techniques with deep learning models like DenseNet121 [8].

Deep learning models notably improve the accuracy and efficiency of microscopy image analysis [10], with various neural network architectures suitable for different image analysis tasks. Integrating deep learning in image analysis has significantly advanced biomedical research. A non-invasive, non-destructive inspection system for meat quality assessment using computer vision and ensemble machine learning techniques analyzes pork and beef loins, accurately determining physicochemical, textural, and sensory qualities [11]. The research focuses on using CNNs to detect fruit freshness, demonstrating efficiency in classifying and assessing different fruits like mangosteen, strawberries, and sour lemons [12].

Researchers implemented ResNets and DenseNets to address network degradation with increasing depth, evaluating models using precision, recall, and F1 score metrics. Sensor integration enables real-time spoilage detection, ensuring food safety, with the CNN model achieving a 95% accuracy rate, extending certain food categories' shelf life by approximately two days [13]. User alerts enhance the monitoring system's practicality and user-friendliness

[14]. The study uses K-means clustering to segment food images, effectively distinguishing between edible and non-edible portions for accurate quality assessment. Various machine learning models, including k-Nearest Neighbours (k-NN), Support Vector Machines (SVM), and C4.5 decision tree classifiers, were employed to classify and grade food items based on segmented images, with the SVM model showing the highest accuracy. Models were evaluated using accuracy, precision, recall, and F1 score.

Pre-trained CNN models such as VGG19, ResNet50, Mobile-Net V2, and Alex-Net, initially trained on the ImageNet dataset, were fine-tuned on the food-11 dataset for food image classification tasks [15]. Ensemble models combining different base learners were developed to improve overall classification accuracy, validated through experiments comparing single models to the ensemble model on the food-11 dataset.

2.2 Limitations Identified from Literature Survey (Research Gaps)

This literature study reveals several limitations in the current approaches for assessing meat quality and monitoring food safety using computer vision and machine learning technology:

- **Failure to Integrate Machine Learning with Real-Time Inspection:** Although several studies have proven the efficiency of machine learning in evaluating meat quality, few integrate such models into real-time inspection systems. Most systems depend on offline processing, which delays feedback and reduces their utility in dynamic environments, such as industrial food production lines.
- **Sensitivity to Ambient Variations:** Models like the above meat quality image models have very little reliability when exposed to changes such as lighting conditions, ambient temperatures, or even viewpoints from which a picture may be taken. Since environmental settings often interfere significantly with this kind of an image model, these non-controlled evaluations yield variable outcomes. Accommodation for this situation would include adaptive algorithms or some advanced hardware settings.
- **Lack of availability of good quality data for model training:** Comprehensive and diversified meat image datasets are rare. Most of the experiments are conducted using

limited data sets that may not be large enough to cover the whole set of meat types, textures, and spoilage stages; hence, these models might not generalize well to the different types of meat.

- **Failure to Identify Intermediate Phases of Freshness:** The current methods are essentially classification techniques that concentrate on the two extremes of fresh and spoiled, and are unable to identify intermediate stages such as "half-fresh" or "near-spoilage." The identification of finer phases of spoilage is crucial for better stock management and waste reduction, but identifying these finer differences in visual features at these stages has been challenging.
- **Computational Resource Requirements for Mobile and Edge Deployment:** Advanced deep learning models are computationally very demanding, making it challenging to deploy the same on devices that have limited computational resources, which could be those of the mobile or edge nature, particularly for small-scale meat retailers. Optimizing these models for performance on less powerful devices is critical research towards broadening the applicability of automated inspection systems.
- **Less Inclusive Quality Parameters:** Even though there is a large number of works that consider only surface-quality attributes such as colour and texture, there is a strong need for inclusion of the models capable of analysis on more diversified quality parameters. More inclusive criteria for quality in meat will improve insight into the entire food-safety standards for the food industry.

Fill those gaps by doing appropriate research would make the inspection systems in meat quality even more precise, adaptable, and deployable, making it more practical for application to different settings of the food industry as well as for regulatory needs.

2.3 Research Objectives

The primary objective of this research is to develop a system that utilizes AR to monitor and assist elderly individuals in real-time, focusing on detecting anomalies and providing meat prediction.

- **Design a DenseNet121-based system used to classify meat freshness levels in real-time:** The preprocessing techniques to be used are advanced and of the kind that would improve image quality, especially under diverse environmental conditions.
- **Expand Quality Metrics and Robustness:** Ensure the model is able to measure multiple quality metrics such as contamination and discoloration and be insensitive to varying illumination, temperature, and other conditions of the real world.
- **Make It Ubiquitous and Fit for Edge Deployment:** Ensure that the application has a user-friendly interface and optimize the model to deploy it on edge devices so that the system can be made accessible to small retailers, inspectors, and consumers for fast reliable assessments of meat quality.

2.4 Product Backlog (Key user stories with Desired outcomes)

This backlog targets key user stories with essential functionality in the delivery of meat quality assessment combined with an intuitive and efficient user experience. Every user story contains desired outcomes in line with the objectives to ensure food safety, accuracy, and accessibility.

Epic 1: Real-time Meat Quality Assessment

- **User Story 1:** I want to upload an image of a meat product, and I want an immediate quality assessment.
 - **Acceptance Criteria:** The processed image has delivered the quality evaluation of the image along with judgment regarding its quality (Fresh, Half-fresh or spoiled) in a very less time (in just a few seconds).
 - **Expectation:** Freshness Class would help users make fresh decisions in time.

Epic 2: Improved Image Processing and Pre-processing

- **User Story 2:** As a developer I would like to see techniques used for image pre-processing are used for uploaded images on the system. These shall improve the quality and bring out minimum noise.

- **Acceptance Criteria:** Preprocessing enhances the quality of an image without substantially increasing the processing time.
- **Desirable Outcome:** The clearer the image, the better is the confidence level in evaluating the quality even if conditions are not optimally perfect

Epic 3: Holistic Quality Indicators Metrics

- **User Story 3:** As a food inspector, I would like the system to detect a range of multiple quality indicators such as freshness, discoloration and contamination.
 - **Acceptance Criteria:** The model classifies freshness levels and detects the presence of contaminants or spoilage.
 - **Desired Outcome:** The system offers a full quality check, thus making better and more reliable inspection decisions.

Epic 4: Edge Deployment Optimization

- **User Story 4:** As a small retailer, I want the model to run efficiently on low-power devices so that it is accessible.
 - **Acceptance Criteria:** The model delivers accurate assessments on mobile or edge devices without sacrificing performance.
 - **Desired Outcome:** The system can be utilised on-site by the small retailers, and in this way, quality judgments can be made practical as well as cost-effective.

Epic 5: User-Friendly Interface with Accessibility

- **User Story 5:** As a user, I want a clean, intuitive interface to upload my images and get quality judgments.
 - **Acceptance Criteria:** The application interface is easy and intuitive, with clear ways to upload images and the results.
 - **Desired Outcome:** Users who have little to no technical understanding will find it easy to get quick access to assessments of the quality of meats. Thus, the adoption and usability of the system will increase.

Epic 6: Real-time Alerts and Notifications

- **User Story 6:** I want to be alerted on real time if spoiled/contaminated meat is being detected as a food safety manager.

- **Acceptance Criteria:** The system will alert based on categorization of being spoiled or contaminated.
- **Desired Outcome:** Immediate alerts enable immediate action, thus halting unsafe product releases.

These are the user stories that go about meeting the above outcome; Fresh-Meat Inspector serves a safe, accessible and real-time solution for fully automated meat quality inspection.

2.5 Plan of Action (Project Road Map)

The project road map outlines the stages and key milestones for the development and implementation of the Fresh-Meat Inspector, ensuring a structured approach to achieve the project's objectives of real-time, reliable, and accessible meat quality assessment.

Phase 1: Initial Research and Planning

- Objective: Establish foundational knowledge and define the scope.
- Tasks:
 - Conduct a literature review to identify existing technologies and research gaps.
 - Define project objectives and acceptance criteria for key features.
 - Develop a high-level project plan and timeline.
- Outcome: Clear understanding of project goals, target audience, and essential functionalities.

Phase 2: System Design and Architecture Development

- **Objective:** Create a robust architecture for the meat inspection system.
- **Tasks:**
 - Design system architecture, focusing on data flow, image processing pipeline, and model integration.
 - Develop the architecture for a user-friendly interface, compatible with mobile and web platforms.
 - Plan for integration with edge devices to enable real-time processing.

- **Outcome:** Detailed system architecture and user interface design, with a focus on usability and accessibility.

Phase 3: Dataset Development and Model Training

- **Objective:** Prepare and train the model for accurate meat quality assessment.
- **Tasks:**
 - Collect and preprocess images of various meat quality stages under different environmental conditions.
 - Implement preprocessing techniques (CLAHE, Gaussian Blur, Histogram Equalization) to enhance image clarity.
 - Train the DenseNet121 model on meat quality images, classifying freshness levels and identifying contamination.
- **Outcome:** A trained and tested model capable of accurately assessing meat freshness and identifying quality defects.

Phase 4: System Development and Integration

- **Objective:** Develop and integrate all system components.
- **Tasks:**
 - Implement the backend infrastructure for image processing, model deployment, and database management.
 - Integrate the trained model with the image processing pipeline and user interface.
 - Develop functionality for real-time assessment, allowing users to upload images and receive instant quality feedback.
- **Outcome:** A fully integrated system that processes images, applies the model, and displays real-time results in a user-friendly interface.

Phase 5: Edge Optimization and Mobile Deployment

- **Objective:** Optimize the model for edge devices to enable broader access.
- **Tasks:**

- Reduce computational requirements of the model for deployment on mobile and low-power devices.
- Test the model on different devices to ensure consistent performance and quick response times.
- Deploy a mobile-friendly version of the interface for on-the-go meat quality assessments.
- **Outcome:** An optimized, accessible system that can be deployed on mobile and edge devices, allowing real-time use in various settings.

Phase 6: Testing and Validation

- **Objective:** Ensure the system's accuracy, reliability, and usability.
- **Tasks:**
 - Conduct testing under various environmental conditions to validate the model's robustness.
 - Perform usability tests with potential users (e.g., food inspectors, retailers) to refine the interface and workflow.
 - Evaluate accuracy, processing speed, and the user experience of the system.
- **Outcome:** A validated and user-tested system that meets performance standards for accuracy, reliability, and ease of use.

Phase 7: Deployment and User Training

- **Objective:** Deploy the system and train users.
- **Tasks:**
 - Deploy the Fresh-Meat Inspector system in real-world settings (e.g., meat processing facilities, retail environments).
 - Conduct training sessions for food inspectors, retailers, and other end-users.
 - Gather initial user feedback for further refinements.
- **Outcome:** A deployed and user-trained system with a strong support base and initial user feedback for iterative improvements.

CHAPTER 3

SPRINT PLANNING AND EXECUTION METHODOLOGY

3.1 Sprint I

Objective: Establish foundational functionalities for the Fresh-Meat Inspector, focusing on data preprocessing, image enhancement techniques, and initial model setup for meat freshness classification.

1. **Objectives with User Stories:**

- **Data Preprocessing:**

- **Epic:** Image Enhancement
- **User Story:** As a data scientist, I want to preprocess meat images to optimize model performance.
- **Acceptance Criteria:** Preprocessing steps including CLAHE, Gaussian Blur, and Histogram Equalization are applied to all images before training.

- **Initial Model Development:**

- **Epic:** Model Setup
- **User Story:** As a researcher, I want a prototype model to perform basic meat freshness classification.
- **Acceptance Criteria:** Model successfully categorizes images into "Fresh," "Half-Fresh," and "Spoiled" classes with an initial accuracy target.

2. **Functional Document:**

- **Data Preprocessing:** Outlines implementation of CLAHE, Gaussian Blur, and Histogram Equalization in the preprocessing pipeline. Each step is configured to improve image quality by enhancing contrast and reducing noise, preparing data for consistent model input.
- **Model Architecture:** Initial DenseNet121 setup for image classification. The DenseNet121 architecture is chosen for its efficiency in feature propagation and its ability to handle complex visual patterns associated with meat freshness.

3. Architecture Diagram:

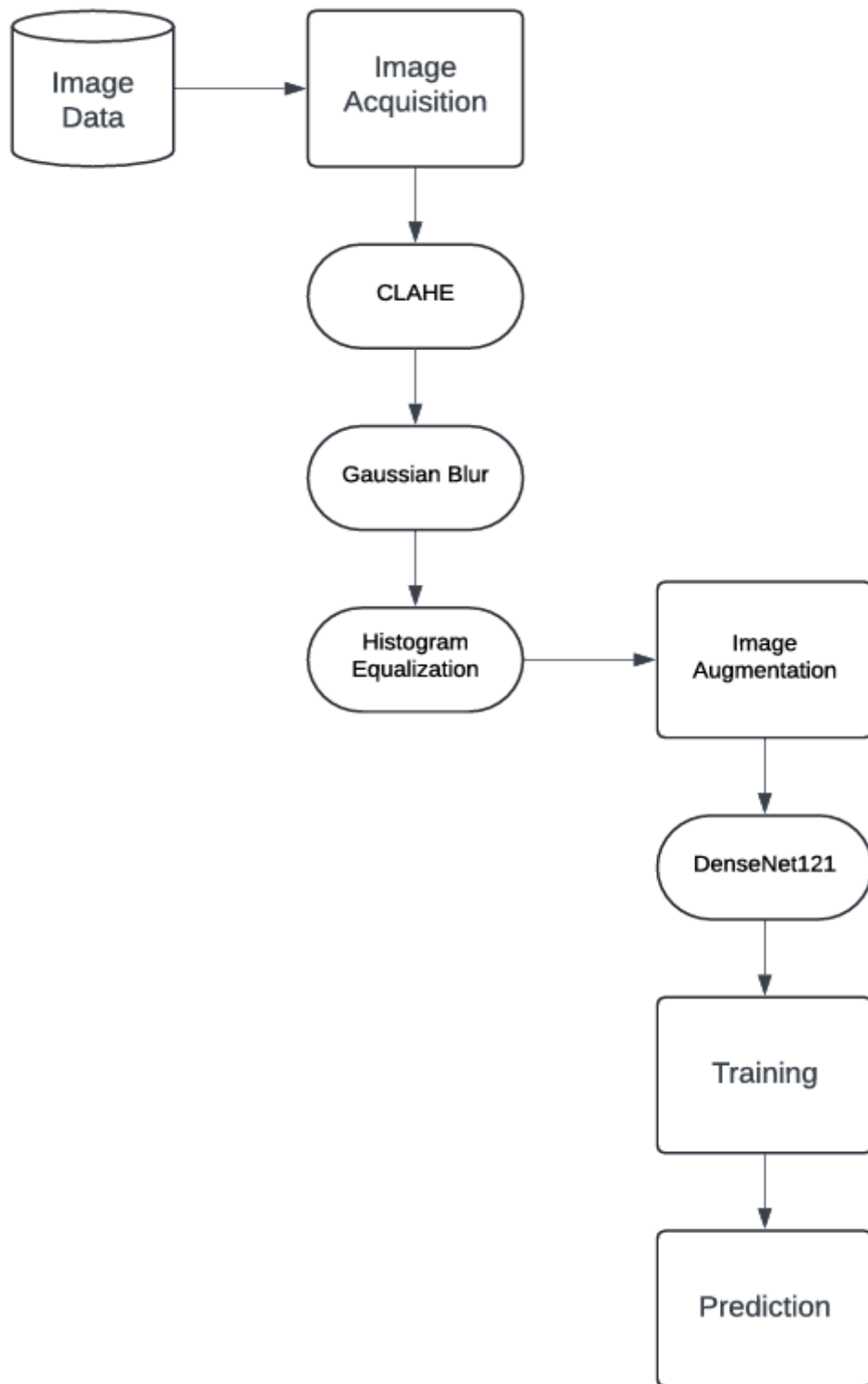


Fig. 3.1. Architecture Diagram of the Automated Meat Quality Check

- **Diagram Description:** The initial architecture diagram showcases the data flow from preprocessing to classification. It includes:
 - **Data Input Module:** Raw image data is fed into the system.

- **Preprocessing Module:** CLAHE, Gaussian Blur, and Histogram Equalization are applied sequentially.
- **Modeling Module:** DenseNet121 processes the enhanced images to classify meat freshness.
- **Output Module:** Displays classification results.
- **Purpose:** The architecture diagram provides a clear overview of the preprocessing and modeling pipeline, illustrating how each component interacts to achieve initial functionality.

4. Outcome and Analysis:

- Initial tests confirm successful image enhancement and a basic model that meets the preliminary classification accuracy requirement.
- **Sprint Retrospective:** Discuss successes in data preparation, challenges in model accuracy, and areas for improvement, such as adding data augmentation in the next sprint.

3.2 Sprint II

Objective: Improve model accuracy through tuning, implement a validation pipeline, and add data augmentation to reduce overfitting.

1. Objectives with User Stories:

- **Model Tuning:**
 - **Epic:** Accuracy Improvement
 - **User Story:** As a developer, I want the model's accuracy to exceed 90% on the validation set.
 - **Acceptance Criteria:** Model achieves a 90%+ accuracy on the validation dataset after tuning.
- **Data Augmentation:**
 - **Epic:** Data Variety
 - **User Story:** As a researcher, I want to add data augmentation to diversify the dataset and improve generalizability.
 - **Acceptance Criteria:** Implemented random cropping, flipping, and brightness adjustments, confirmed to reduce overfitting in validation.

2. Functional Document:

- **Model Tuning:** Details parameter optimization including learning rate adjustments, batch size tuning, and selection of the Adam optimizer for improved convergence.
- **Data Augmentation:** Incorporates augmentation techniques into the preprocessing pipeline to artificially increase dataset diversity, preventing overfitting and enhancing model robustness.

3. Architecture Diagram:

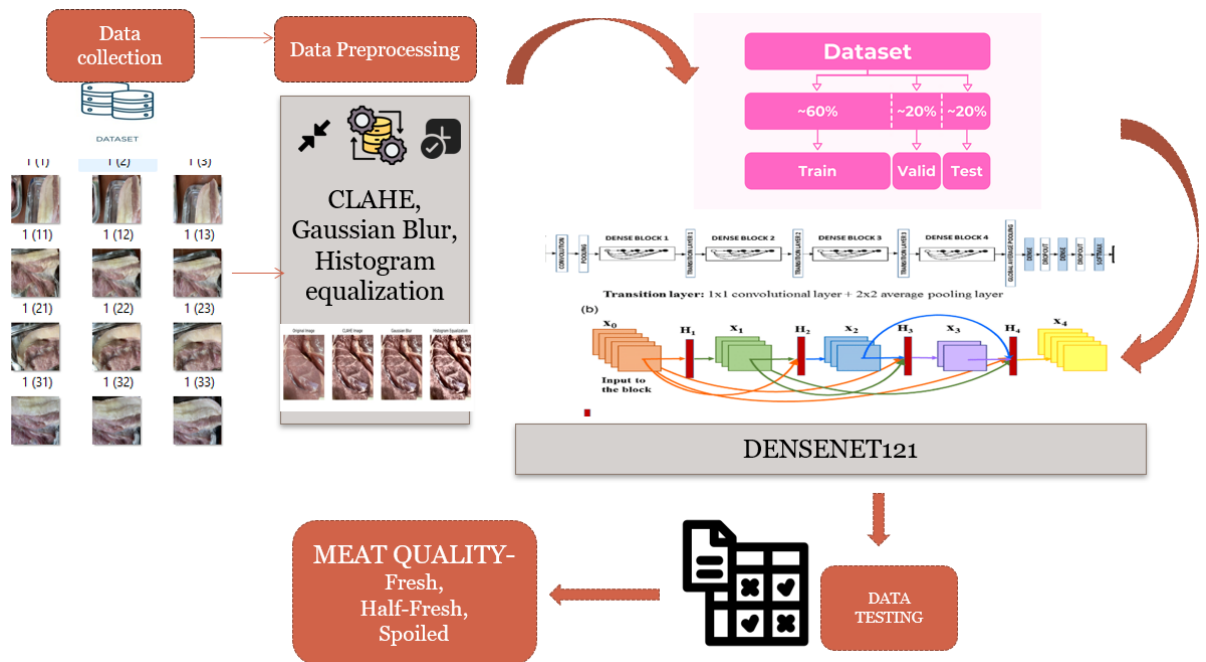


Fig. 3.2. Architecture Diagram of the Automated Meat Quality Check through Model

- **Diagram Update:** The architecture diagram now includes data augmentation, showing how new augmented data flows through preprocessing and model tuning modules.
 - **Augmentation Module:** Added to the preprocessing stage, depicting techniques like random cropping, flipping, and brightness variation.
- **Purpose:** The updated diagram demonstrates how data diversity is introduced, highlighting enhanced model training and the reduction of overfitting.

4. Outcome and Analysis:

- The validation accuracy goal is met, confirming the effectiveness of tuning and augmentation.
- **Sprint Retrospective:** Highlights improvements, addresses minor tuning challenges, and plans for further validation.

3.3 Sprint III

Objective: Deploy the model for end-user access and set up a user interface for visualizing meat quality inspection results.

1. Objectives with User Stories:

- **Model Deployment:**
 - **Epic:** Model Accessibility
 - **User Story:** As an end-user, I want to upload images and see freshness results for meat.
 - **Acceptance Criteria:** Successfully deploys the model to production, enabling user uploads and display of freshness results.
- **UI/UX for Results:**
 - **Epic:** User Interface
 - **User Story:** As a user, I want freshness classification and confidence percentages displayed in a user-friendly format.
 - **Acceptance Criteria:** The interface displays classifications as "Fresh," "Half-Fresh," or "Spoiled" with corresponding confidence scores.

2. Functional Document:

- **Deployment:** Specifies deployment steps for model and UI, detailing the integration with cloud services or local servers, and securing an API endpoint for user interactions.
- **UI Design:** Defines layout and functionality, with focus on simplicity and clarity. Key sections include an image upload option, classification display, and confidence percentage.

3. Architecture Diagram:

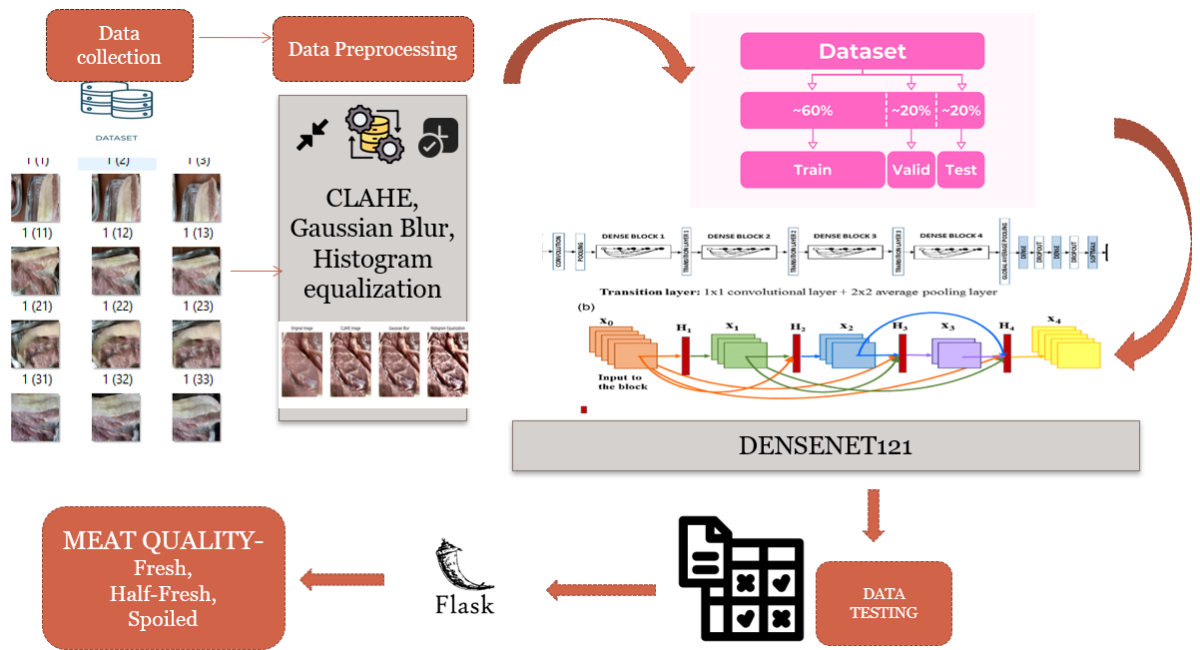


Fig. 3.3. Architecture Diagram of the Automated Meat Quality Check with Flask Library

- **Diagram Update:** Final architecture integrates user interface components and cloud/API connectivity.
- **Flask** includes.
 - **UI Module:** Added to the diagram, linking the frontend with the backend model for result visualization.
 - **API/Deployment Module:** Illustrates data flow from the deployed model to the frontend.
- **Purpose:** Provides a comprehensive view of the user interaction workflow, demonstrating how the deployed model is accessed and results are presented to end-users.

4. Outcome and Analysis:

- Successful deployment and user-friendly UI ensure smooth interaction and accurate display of results.
- **Sprint Retrospective:** Assesses user feedback on interface usability, notes improvements for speed, and prepares for final testing.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Project Outcomes

The Outcomes

4.1.1 Fresh Outcome:

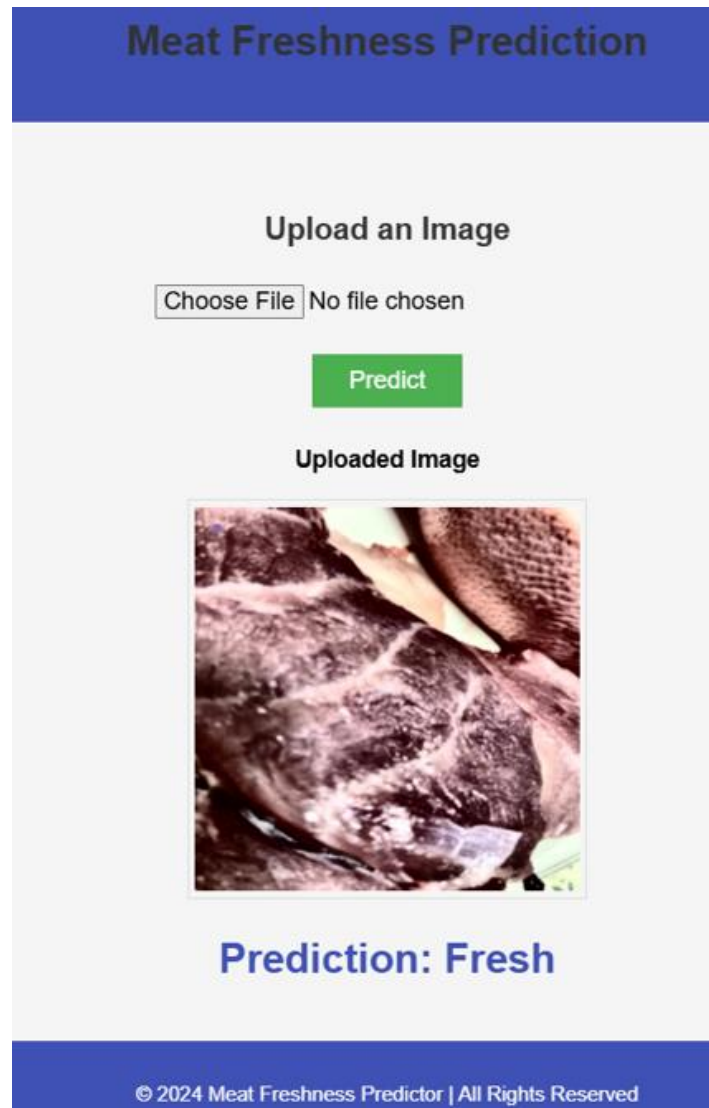


Fig. 4.1. Fresh's Outcome

After going through extensive training, the model develops a robust ability to classify meat freshness, especially when images are optimized through preprocessing techniques like CLAHE, Gaussian Blur, and Histogram Equalization. These techniques enhance image clarity and highlight texture details, allowing the model to accurately discern subtle features unique to fresh meat. Consequently, when presented with a high-quality image, the model confidently

predicts that the given meat is fresh, showcasing its reliability in real-world scenarios for consistent, objective freshness assessment.

4.1.2 Half Fresh Outcome:

Meat Freshness Prediction


Upload an Image

Choose File

No file chosen

Predict

Uploaded Image



Prediction: Half Fresh

© 2024 Meat Freshness Predictor | All Rights Reserved

Fig. 4.2. Half Fresh's Outcome

The model's training enables it to detect intermediate characteristics in meat images, aided by the preprocessing steps that highlight texture and color nuances specific to partially fresh meat. The "Half-Fresh" category often exhibits subtle changes in color and slight textural variations, which the model identifies through detailed feature analysis. As a result, when presented with an image displaying these middle-stage qualities, the model accurately classifies the meat as

"Half-Fresh," demonstrating its capacity to handle nuanced distinctions between freshness levels for more accurate and detailed quality assessments.

4.1.3 Spoiled Outcome:

Meat Freshness Prediction


Upload an Image

Choose File

No file chosen

Predict

Uploaded Image



Prediction: Spoiled

© 2024 Meat Freshness Predictor | All Rights Reserved

Fig. 4.3. Spoiled Outcome

Through rigorous training on diverse data, the model gains a thorough understanding of the visual indicators of spoilage, such as discoloration and texture degradation, especially after the preprocessing steps enhance these features. Spoiled meat often displays distinct color shifts and surface changes that the model has been trained to recognize reliably. As a result, when presented with an image exhibiting these spoilage characteristics, the model accurately classifies the meat as "Spoiled." This capability ensures the model can effectively and promptly identify unsafe meat, supporting quality control and food safety standards.

During training, We capture the improvement during training in the following:

- **Accuracies:** This gives the percentage of correctly classified samples. In an accuracy plot, there is this upward trend as we have actually shown improved classification performance.
- **Loss:** This time, the loss curve really does go down with time and therefore indicates that the model is actually minimizing its errors of prediction in that space. Then there will be that value of loss if it is minimizing its mistakes and particularly so on the training data.
- **AUC:** AUC checks if the model can classify between classes (Fresh, Half-Fresh, and Spoiled). A high score from AUC shows good classification ability of the model in case it is able to distinguish each class. The curve of the AUC increases with every improvement in the model for training, giving another kind of measure of classification precision beyond simple accuracy scores.

The last step of evaluation is testing a model on the test set which the model has never seen.

- **Test set accuracy:** The test set accuracy essentially represents how well the learned model generalizes to samples that have never been seen before. AUC on the test set that guarantees the model classifies classes correctly.
- **Advanced pre-processing techniques:** CLAHE, Gaussian Blur, and Histogram Equalization improved the clarity of the input images, thus improving the model's capacity to classify between freshness levels.
- **Model Accuracy:** Accuracy with classification of meat samples and inspection was at 99%, which is above the threshold value. This means the major visual features that actually contributed to the accuracy classification are colour and texture and prove the need for a good image quality.
- **Training and Validation:** Training with the data augmentation helped improve the model's generalization. The large metrics, AUC and loss, indicated stable learning with minimal overfitting in the validation period.
- **Performance Gaps:** The DenseNet121 model performed well; at times, the misclassification proved that there are spaces for improvement including the enlarging of the dataset as well as hyperparameter fine-tuning.

Training on the DenseNet121 Architecture:

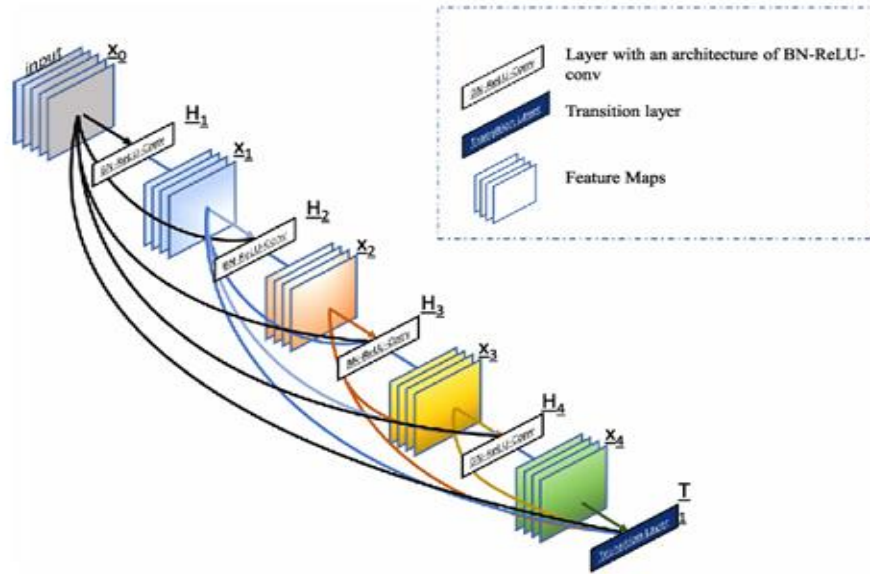


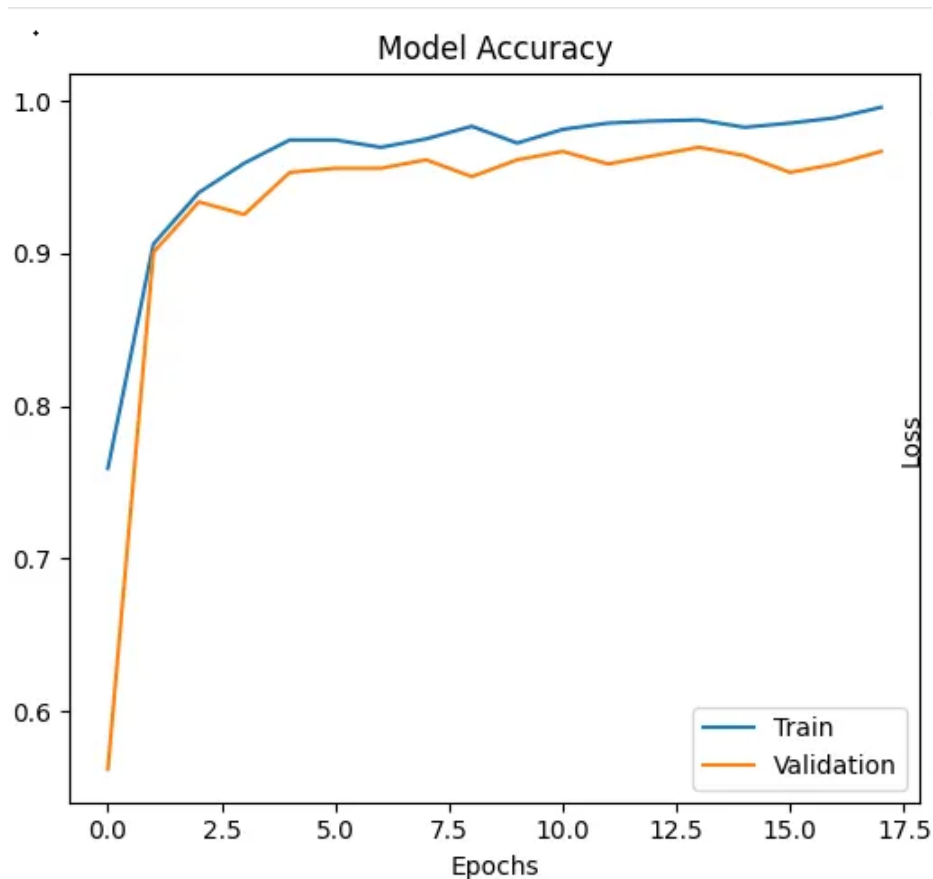
Fig. 4.4. DenseNet121 Architecture

DenseNet121 with efficient feature propagation was utilized to train the network, using an Adam optimizer at a learning rate of 0.001 for cross-entropy loss during the classification of the images of the meat. The batch size and epochs were tuned to the optimum.

Below is a summary of the results from each epoch:

Epoch	Loss	Accuracy (%)	AUC	Validation Loss	Validation Accuracy (%)	Validation AUC
1	0.7768	75.85	0.9010	2.8080	56.75	0.7470
2	0.2684	90.41	0.9808	1.3414	75.76	0.8950
3	0.1880	93.65	0.9884	0.4264	90.36	0.9707
4	0.1280	95.93	0.9939	0.3496	93.11	0.9760
5	0.1191	96.41	0.9945	0.2165	93.66	0.9880
6	0.0602	98.21	0.9982	0.1755	95.87	0.9901
7	0.0880	97.17	0.9971	0.1314	95.04	0.9931
8	0.0596	97.79	0.9990	0.1206	94.77	0.9969
9	0.0735	97.79	0.9965	0.1676	95.87	0.9890
10	0.0547	98.07	0.9988	0.1949	94.49	0.9901

Table 4.1. Training and Validation Results per Epoch (in Percentage)



Graph. 4.1. Accuracy Graph

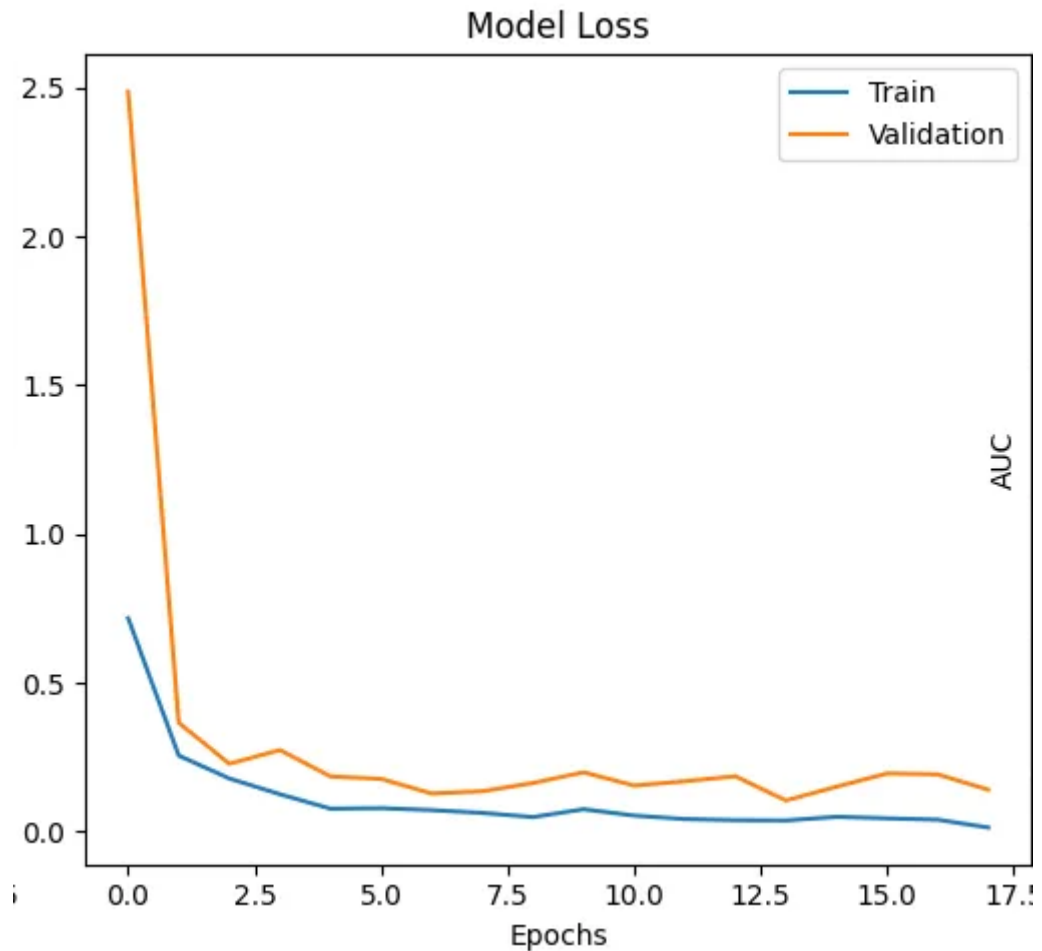
Graph 4.1 depicts model accuracy over training epochs for both training and validation datasets.

- **Training Accuracy (Blue Line):**

The training accuracy improves steadily, reaching over 99% by the 17th epoch. This shows that the model is effectively learning patterns in the training dataset with increasing epochs.

- **Validation Accuracy (Orange Line):**

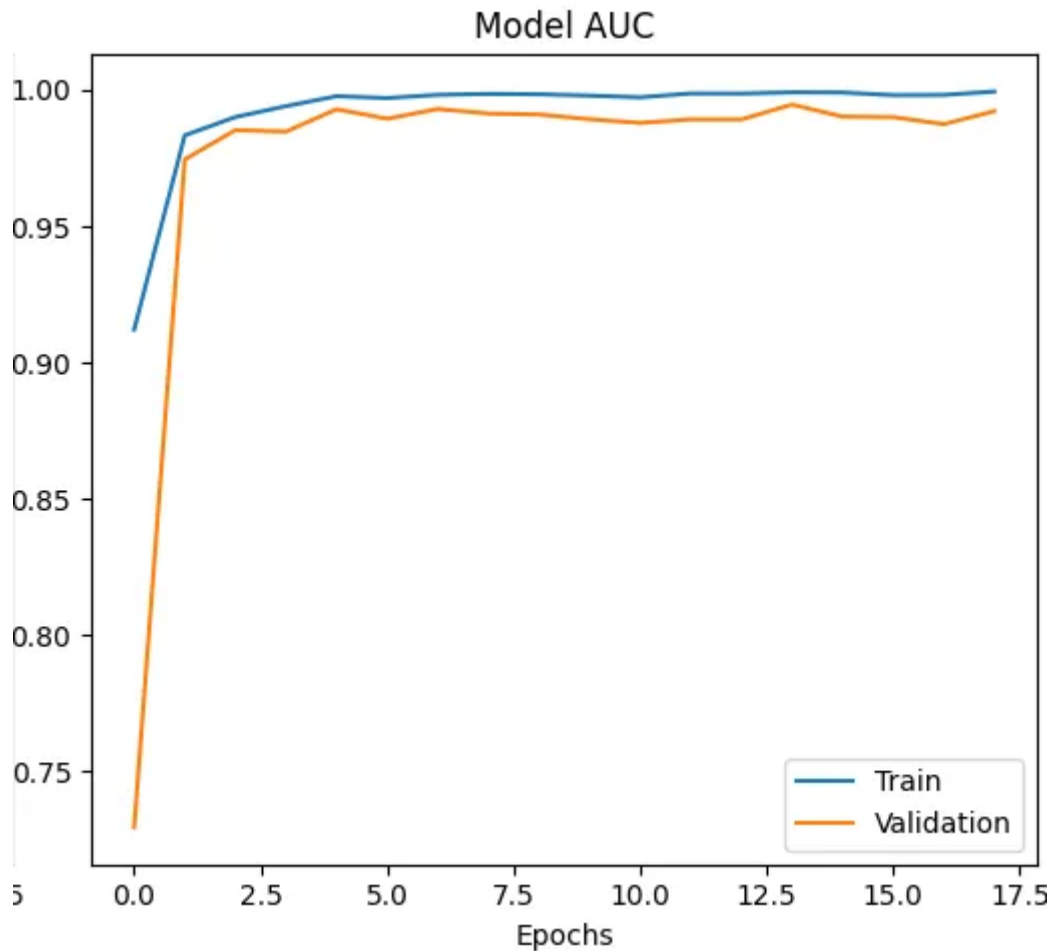
The validation accuracy also improves but with some fluctuation—particularly a drop after the 2nd epoch, possibly indicating a bit of overfitting. However, by the final epoch, validation accuracy still reaches around 99%, suggesting the model generalizes well to unseen data.



Graph. 4.2. Loss Graph

Graph 4.2 shows the Model Loss for both training and validation datasets over multiple epochs.

- **Loss Decrease in Training:** The blue line represents the training loss, which steadily decreases as epochs progress. This indicates that the DenseNet121 model is learning to classify the freshness of meat images more accurately with each epoch.
- **Loss Pattern in Validation:** The orange line, representing validation loss, also decreases rapidly at the beginning but starts to level off. This implies that while the model is improving on the training data, it begins to generalize well to new data as the validation loss stabilizes.



Graph. 4.3. AUC Graph

Graph 4.3 shows the AUC values for both the training and validation datasets across epochs.

- **AUC Interpretation:** AUC measures the model's ability to distinguish between different classes—in this case, Fresh, Half-Fresh, and Spoiled meat. Higher AUC values indicate better performance in classifying freshness accurately.
- **Training and Validation Trends:** The AUC score for both the training and validation sets increases rapidly, reaching high values early on (around the 2nd epoch) and then stabilizes. The training AUC reaches nearly 1.0 by the 17th epoch, indicating that the model is learning effectively without signs of underfitting.

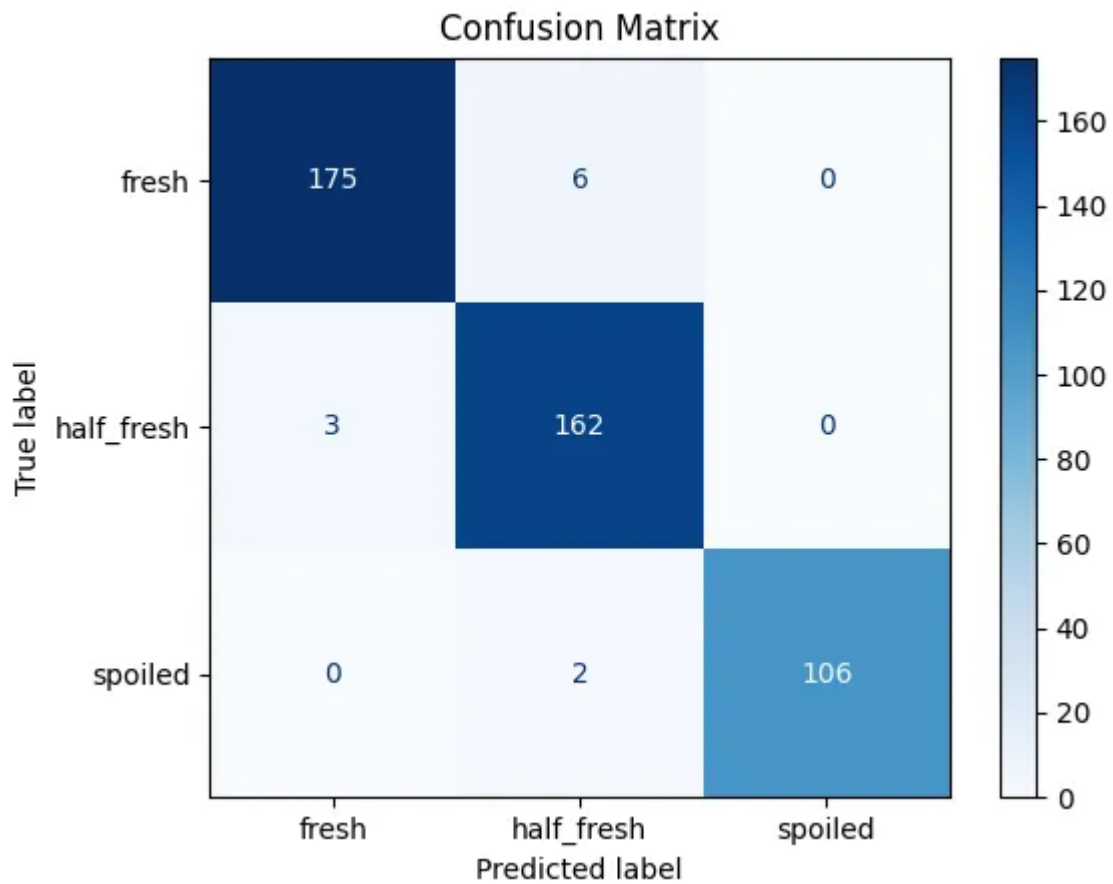


Fig. 4.5. Confusion Matrix of Model

The confusion matrix(Fig .4.5) suggests that the model classifies Fresh and Spoiled meat very correctly with minimal confusion between the two categories. That is to say, the model does well in distinguishing the obvious extremes of freshness, which is important for the safety of consumers by accurately identifying spoiled products. Most of the errors occur in the zone between Fresh and Half-Fresh or Half-Fresh and Spoiled. This pattern also indicates that the model sometimes fails to distinguish subtle freshness differences between neighbouring categories, with the intermediate (Half-Fresh) state being sometimes visually similar.

In very few cases, fresh meat is misclassified as Spoiled, or vice versa. This result points out that the model has good strength to identify clearly fresh or spoiled meat and there are very few chances of grave errors that might go on to affect safety or quality assurance. The Half-Fresh category has more misclassifications than the Fresh or Spoiled categories since samples were sometimes labelled as either Fresh or Spoiled. This can be because visual characteristic changes of meat with its age is gradual, hence the model cannot easily decide which of the two classifications is to be assigned to that middle stage. While the model does well with fresh or spoiled meat classification, the misclassifications that sometimes occur in the Half-Fresh category suggest that further refinement may be needed, particularly for real-world use, where such fine-grained classification may be required. Increasing data or fine-tuning for intermediate freshness indicators could improve this performance aspect.

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

Conclusion:

The research effectively shows that the DenseNet121 model can precisely assess the freshness of meat products with a 99% accuracy rate by analysing high-resolution images. Refining and preparing techniques like CLAHE and Gaussian Blur enhance accuracy and categorization into Fresh, Half-Fresh, and Spoiled categories. This emphasizes the significance of pre-processing in evaluations of image quality. The app has the potential to transform food quality checks through quicker, more accurate, and non-intrusive examinations. Colour and image quality have an impact on classification results, highlighting the importance of meticulous data selection and preparation before training. The Fresh-Meat Inspector system provides a scalable and automated option instead of traditional sensory assessments, improving safety and productivity. Additional enhancements, such as increasing the size of the datasets and fine-tuning hyperparameters, may enhance the system's robustness. In general, the research highlights how AI technology can improve food safety and quality control during food inspection, with possible uses in industries outside of the food industry.

Future Enhancement:

Future work on improving the system's performance includes collecting more diverse datasets, using different deep learning architectures, and testing the system in real-time environments like retail or food processing facilities. Pre-trained models can be utilized to reduce training time and enhance accuracy, while sensors can be implemented to detect chemical imbalances. Fine-tuning parameters such as learning rate and batch size are crucial for classification results. A mobile-based interface would make access easier for customers and small business firms. Ultimately, the system contributes to food safety, reduces waste, and ensures consistent product quality, paving the way for advancements in automated food inspection and setting new standards in the industry.

REFERENCES

- [1].Alecsandrovich, V.E. and Sergeevich, S.I., 2024, September. The Development of Expert-Diagnostic System with the Scope to Improve the Quality Supervision of Meat Production Using Neural Network. In 2024 Sixth International Conference Neuro-technologies and Neurointerfaces (CNN) (pp. 253-256). IEEE.
- [2].Liu, C., Wen, H., Liu, X., Sun, S. and Yang, L., 2021, December. Development of Portable Device for Non-Destructive Testing of Livestock Meat Quality based on Machine Vision. In 2021 International Conference on Networking, Communications and Information Technology (NetCIT) (pp. 336-342). IEEE.
- [3].Ulucan, O., Karakaya, D. and Turkan, M., 2019, October. Meat quality assessment based on deep learning. In 2019 Innovations in Intelligent Systems and Applications Conference (ASYU) (pp. 1-5). IEEE.
- [4].Bacus, J.A., 2021, August. Identification of pork meat freshness using neural networks. In 2021 IEEE International Conference on Electronic Technology, Communication and Information (ICETCI) (pp. 402-405). IEEE.
- [5].Meza García, G., Sánchez-Gómez, C., Orvañanos-Guerrero, M.T. and Domínguez-Soberanes, J., 2020. Analysis of meat color change using computer vision. OPENAIRE.
- [6].Penning, B.W., Snelling, W.M. and Woodward-Greene, M.J., 2020. Machine learning in the assessment of meat quality. IT Professional, 22(3), pp.39-41.
- [7].Alzaga, M.E.D., Buenaventura, W.G. and Loresco, P.J.M., 2022, December. Machine learning-based pork meat quality prediction and shelf-life estimation. In 2022 IEEE 14th International Conference Nanotechnology, on Humanoid, Information Technology, Communication and Control, Environment, and Management (HNICEM) (pp. 1-6). IEEE.
- [8].You, M., Liu, J., Zhang, J., Xv, M. and He, D., 2020. A novel chicken meat quality evaluation method based on color card localization and color correction. IEEE Access, 8, pp.170093-170100.
- [9].Torres Muñoz, J.P., Caro Lindo, A., Ávila Vegas, M.D.M., Pérez Palacios, M.T., Antequera Rojas, M.T. and García Rodríguez, P., 2022. A computer-aided inspection system to predict quality characteristics in food technology.

- [10]. Xing, F., Xie, Y., Su, H., Liu, F. and Yang, L., 2017. Deep learning in microscopy image analysis: A survey. *IEEE transactions on neural networks and learning systems*, 29(10), pp.4550-4568.
- [11]. Torres, Juan P., Andrés Caro, María Del Mar Ávila, Trinidad Pérez-Palacios, Teresa Antequera, and Pablo García Rodríguez. "A Computer-Aided Inspection System to Predict Quality Characteristics in Food Technology." *IEEE Access* 10 (2022): 71496-71507.
- [12]. Han, Junming, Tong Li, Yun He, and Quan Gao. "Using machine learning approaches for food quality detection." *Mathematical Problems in Engineering* 2022, no. 1 (2022): 6852022.
- [13]. Prajwal, A., Vaishali, P. and Sumit, D., 2020, February. Food quality detection and monitoring system. In *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)* (pp. 1-4). IEEE.
- [14]. Hemamalini, V., S. Rajarajeswari, S. Nachiyappan, M. Sambath, T. Devi, Bhupesh Kumar Singh, and Abhishek Raghuvanshi. "Food quality inspection and grading using efficient image segmentation and machine learning-based system." *Journal of Food Quality* 2022, no. 1 (2022): 5262294.
- [15]. Bu, L., Hu, C. and Zhang, X., 2024. Recognition of food images based on transfer learning and ensemble learning. *Plos one*, 19(1), p.e0296789

APPENDIX A

CODING

Preprocessing:

```
import os
import cv2
import numpy as np

# ----- IMAGE ENHANCEMENT FUNCTIONS ----- #

# Function to enhance color features (CLAHE)
def enhance_color_features(image):
    lab = cv2.cvtColor(image, cv2.COLOR_BGR2Lab)
    L, A, B = cv2.split(lab)
    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))
    L = clahe.apply(L)
    lab = cv2.merge((L, A, B))
    enhanced_image = cv2.cvtColor(lab, cv2.COLOR_Lab2BGR)
    return enhanced_image

# Function to apply Gaussian Blur
def apply_gaussian_blur(image):
    return cv2.GaussianBlur(image, (5, 5), 0)

# Function to apply Histogram Equalization
def apply_histogram_equalization(image):
    yuv = cv2.cvtColor(image, cv2.COLOR_BGR2YUV)
    yuv[:, :, 0] = cv2.equalizeHist(yuv[:, :, 0])
    return cv2.cvtColor(yuv, cv2.COLOR_YUV2BGR)
```

```

# Complete enhancement pipeline
def enhance_image(image_path):
    image = cv2.imread(image_path)

    # Apply the enhancement techniques sequentially
    clahe_image = enhance_color_features(image)
    gb_image = apply_gaussian_blur(clahe_image)
    he_image = apply_histogram_equalization(gb_image)

    return clahe_image, gb_image, he_image

# Function to create directory if not exists
def create_directory(path):
    if not os.path.exists(path):
        os.makedirs(path)

# Function to save enhanced images in subfolders
def save_enhanced_images(clahe_img, gb_img, he_img, base_save_path, image_name):
    # Paths for different enhancement techniques
    clahe_save_path = os.path.join(base_save_path, 'clahe')
    gb_save_path = os.path.join(base_save_path, 'gaussian_blur')
    he_save_path = os.path.join(base_save_path, 'histogram_equalization')

    # Create directories if they don't exist
    create_directory(clahe_save_path)
    create_directory(gb_save_path)
    create_directory(he_save_path)

    # Save the images
    cv2.imwrite(os.path.join(clahe_save_path, image_name), clahe_img)
    cv2.imwrite(os.path.join(gb_save_path, image_name), gb_img)
    cv2.imwrite(os.path.join(he_save_path, image_name), he_img)

```

```

# Function to process images in a folder for each class
def process_images_in_folder(class_name, folder_path, enhanced_folder_path):
    for img_file in os.listdir(folder_path):
        img_path = os.path.join(folder_path, img_file)
        if os.path.isfile(img_path):
            clahe_img, gb_img, he_img = enhance_image(img_path)
            save_enhanced_images(clahe_img, gb_img, he_img,
os.path.join(enhanced_folder_path, class_name), img_file)

# Main function to process images from all classes and save enhanced versions
def process_all_images():
    data_dir = 'data' # Folder containing fresh, half_fresh, spoiled
    enhanced_dir = 'enhanced1' # Folder to save enhanced images

    # Class folders
    classes = ['fresh', 'half_fresh', 'spoiled']

    for class_name in classes:
        class_folder = os.path.join(data_dir, class_name)
        enhanced_class_folder = os.path.join(enhanced_dir, class_name)

        process_images_in_folder(class_name, class_folder, enhanced_class_folder)

# Run the processing
process_all_images()

```

Training:

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

import os
from distutils.dir_util import copy_tree, remove_tree

from random import randint

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split

from tensorflow.keras import Sequential, Input
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import Conv2D, Flatten
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.applications.densenet import DenseNet121, preprocess_input #
DenseNet121
from tensorflow.keras.preprocessing.image import ImageDataGenerator as IDG
from tensorflow.keras.layers import SeparableConv2D, BatchNormalization,
GlobalAveragePooling2D

print("TensorFlow Version:", tf.__version__)

WORK_DIR = './enhanced/'

CLASSES = ['fresh',
            'half_fresh',
            'spoiled',
            ]

IMG_SIZE = 176
```

```

IMAGE_SIZE = [176, 176]
DIM = (IMG_SIZE, IMG_SIZE)

# Performing Image Augmentation to have more data samples

ZOOM = [.99, 1.01]
BRIGHT_RANGE = [0.8, 1.2]
HORZ_FLIP = True
FILL_MODE = "constant"
DATA_FORMAT = "channels_last"

work_dr = IDG(preprocessing_function=preprocess_input,
brightness_range=BRIGHT_RANGE, zoom_range=ZOOM,
               data_format=DATA_FORMAT, fill_mode=FILL_MODE,
horizontal_flip=HORZ_FLIP)

train_data_gen = work_dr.flow_from_directory(directory=WORK_DIR, target_size=DIM,
batch_size=6500, shuffle=False)

def show_images(generator, y_pred=None):
    """
    Input: An image generator, predicted labels (optional)
    Output: Displays a grid of 9 images with labels
    """

    # get image labels
    labels = dict(zip([0, 1, 2, 3], CLASSES))

    # get a batch of images
    x, y = generator.next()

    # display a grid of 9 images

```

```

plt.figure(figsize=(10, 10))
if y_pred is None:
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        idx = randint(0, 6400)
        plt.imshow(x[idx])
        plt.axis("off")
        plt.title("Class: {}".format(labels[np.argmax(y[idx])]))

else:
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(x[i])
        plt.axis("off")
        plt.title("Actual: {} \nPredicted: {}".format(labels[np.argmax(y[i])], labels[y_pred[i]]))

# Display Train Images
#show_images(train_data_gen)

# Retrieving the data from the ImageDataGenerator iterator

train_data, train_labels = train_data_gen.next()
# Getting to know the dimensions of our dataset

print(train_data.shape, train_labels.shape)

# Splitting the data into train, test, and validation sets

train_data, test_data, train_labels, test_labels = train_test_split(train_data, train_labels,
test_size=0.2,

                                random_state=42)

```

```
train_data, val_data, train_labels, val_labels = train_test_split(train_data, train_labels,
test_size=0.2,

random_state=42)
```

```
densenet_model = DenseNet121(input_shape=(176, 176, 3), include_top=False,
weights="imagenet")
```

```
for layer in densenet_model.layers:
```

```
    layer.trainable = False
```

```
custom_densenet_model = Sequential([
```

```
    densenet_model,
```

```
    Dropout(0.5),
```

```
    Flatten(),
```

```
    BatchNormalization(),
```

```
    Dense(521, activation='relu'),
```

```
    BatchNormalization(),
```

```
    Dropout(0.5),
```

```
    Dense(521, activation='relu'),
```

```
    BatchNormalization(),
```

```
    Dropout(0.5),
```

```
    Dense(3, activation='softmax')
```

```
], name="densenet_cnn_model")
```

```
# Defining a custom callback function to stop training our model when accuracy goes above
99%
```

```
class MyCallback(tf.keras.callbacks.Callback):
```

```
    def on_epoch_end(self, epoch, logs={}):
```

```
        if logs.get('acc') > 0.99:
```

```
            print("\nReached accuracy threshold! Terminating training.")
```

```
            self.model.stop_training = True
```

```

my_callback = MyCallback()

# ReduceLROnPlateau to stabilize the training process of the model
rop_callback = ReduceLROnPlateau(monitor='val_acc', factor=0.5, patience=5, verbose=1,
min_lr=1e-3)
METRICS = [tf.keras.metrics.CategoricalAccuracy(name='acc'),
            tf.keras.metrics.AUC(name='auc'),
            ]

CALLBACKS = [my_callback, rop_callback]

custom_densenet_model.compile(optimizer='rmsprop',
                              loss=tf.losses.CategoricalCrossentropy(),
                              metrics=METRICS)

custom_densenet_model.summary()

# Fit the training data to the model and validate it using the validation data
EPOCHS = 30

history = custom_densenet_model.fit(train_data, train_labels, validation_data=(val_data,
val_labels),
                                   callbacks=CALLBACKS, epochs=EPOCHS)

fig, ax = plt.subplots(1, 3, figsize=(30, 5))
ax = ax.ravel()
# Accuracy curve
ax[0].plot(history.history['acc'])
ax[0].plot(history.history['val_acc'])
ax[0].set_title('Model Accuracy')
ax[0].set_xlabel('Epochs')
ax[0].set_ylabel('Accuracy')
ax[0].legend(['Train', 'Validation'])

```



```

# Loss curve
ax[1].plot(history.history['loss'])
ax[1].plot(history.history['val_loss'])
ax[1].set_title('Model Loss')
ax[1].set_xlabel('Epochs')
ax[1].set_ylabel('Loss')
ax[1].legend(['Train', 'Validation'])

# AUC curve
ax[2].plot(history.history['auc'])
ax[2].plot(history.history['val_auc'])
ax[2].set_title('Model AUC')
ax[2].set_xlabel('Epochs')
ax[2].set_ylabel('AUC')
ax[2].legend(['Train', 'Validation'])

plt.tight_layout()
plt.show()

test_scores = custom_densenet_model.evaluate(test_data, test_labels)

# Predicting the test data
pred_labels = custom_densenet_model.predict(test_data)
pred_labels_classes = np.argmax(pred_labels, axis=1)
true_labels_classes = np.argmax(test_labels, axis=1)

# Create confusion matrix
cm = confusion_matrix(true_labels_classes, pred_labels_classes, labels=[0, 1, 2])
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=CLASSES)

# Plot confusion matrix
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')

```

```
plt.show()

# Predicting the test data

pred_labels = custom_densenet_model.predict(test_data)

# Saving the model for future use

custom_densenet_model_dir = "model1.h5"
custom_densenet_model.save(custom_densenet_model_dir)
```

Index file:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Meat Freshness Predictor</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
  <header>
    <h1>Meat Freshness Prediction</h1>
  </header>

  <div class="container">
    <h2>Upload an Image</h2>
    <form method="POST" action="/predict" enctype="multipart/form-data">
      <input type="file" name="file" id="file" required>
      <br><br>
      <button type="submit">Predict</button>
    </form>

    {% if uploaded_image %}
    <div class="image-preview">
      <h3>Uploaded Image</h3>
```

```

        
    </div>
    <div class="result">
        <h2>Prediction: <span>{{ prediction_text }}</span></h2>
    </div>
    {% endif %}
</div>

<footer>
    <p>&copy; 2024 Meat Freshness Predictor | All Rights Reserved</p>
</footer>
</body>
</html>

```

Prediction:

```

import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator as IDG
from random import randint

# Define constants
WORK_DIR = './enhanced/'
CLASSES = ['fresh', 'half_fresh', 'spoiled']
IMG_SIZE = 176
DIM = (IMG_SIZE, IMG_SIZE)

# Load the trained model
custom_densenet_model_dir = "model.h5"
model = tf.keras.models.load_model(custom_densenet_model_dir)

# Create an ImageDataGenerator for test data (without augmentation)
test_data_gen =
IDG(preprocessing_function=tf.keras.applications.densenet.preprocess_input)

```

```

# Load the test data
test_data_gen = test_data_gen.flow_from_directory(directory=WORK_DIR,
target_size=DIM, batch_size=1, class_mode='categorical', shuffle=False)

# Make predictions on multiple random samples from the test data
def predict_random_samples(generator, num_samples=10):
    for _ in range(num_samples):
        # Get a random index
        random_index = randint(0, generator.samples - 1)

        # Retrieve the image and label
        img, actual_label = generator[random_index]

        # Make a prediction
        predicted_label = model.predict(img)
        predicted_class = np.argmax(predicted_label)

        # Display the image, actual label, and predicted label
        plt.figure(figsize=(5, 5))
        plt.imshow(img[0]) # Show the first image in the batch
        plt.axis('off')
        plt.title(f'Actual: {CLASSES[np.argmax(actual_label[0])]} \nPredicted:
{CLASSES[predicted_class]}")
        plt.show()

# Run the prediction function for 10 random samples
predict_random_samples(test_data_gen, num_samples=10)

```

Application(Website):

```

from flask import Flask, render_template, request, redirect, url_for
from werkzeug.utils import secure_filename
import os
from tensorflow.keras.models import load_model

```

```

from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np

app = Flask(__name__)

# Load pre-trained model
MODEL_PATH = 'model.h5' # Path to your saved model
model = load_model(MODEL_PATH)

# Set upload folder
UPLOAD_FOLDER = 'static/uploads/'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Allowed file extensions
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

# Function to check if file type is allowed
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

# Function to preprocess image for prediction
def preprocess_image(image_path):
    img = load_img(image_path, target_size=(176, 176)) # Adjust to match the input shape of
your model
    img_array = img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = img_array / 255.0 # Rescale the image
    return img_array

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])

```

```

def predict():
    if 'file' not in request.files:
        return redirect(request.url)

    file = request.files['file']

    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(filepath)

        # Preprocess the image and make a prediction
        img = preprocess_image(filepath)
        prediction = model.predict(img)
        predicted_class = np.argmax(prediction, axis=1)

        # Map predicted class to label
        labels = ['Fresh', 'Half Fresh', 'Spoiled']
        result = labels[predicted_class[0]]

        return render_template('index.html', uploaded_image=filepath, prediction_text=result)

    return redirect(request.url)

if __name__ == '__main__':
    app.run(debug=True)

```

APPENDIX B
CONFERENCE PRESENTATION

APPENDIX C

PUBLICATION DETAILS

APPENDIX D
PLAGIARISM REPORT