

FRESHNESS OF FOOD DETECTION USING MACHINE LEARNING FOR IOT

A Project Report Phase – II

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Electronics and Communication Engineering

by

ISWARYA CH (39130086)

PRATHYUSHA REDDY P (39130350)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI 600119

APRIL – 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICAITON ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of ISWARYA CH (39130086) and PRATHYUSHA REDDY P (39130350) who carried out the project entitled "FRESHNESS OF FOOD DETECTION USING MACHINE LEARNING FOR IOT" under our supervision from January 2023 to April 2023


Internal Guide

DR.V.VIJAYAKUMAR, MTech., Ph.D.,

ASSOCIATE PROFESSOR


Head of Department

Dr. T. RAVI M.E., Ph.D.,

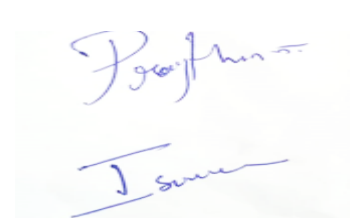
Submitted for Viva voce Examination held on _____


Internal Examiner


External Examiner

DECLARATION

We, ISWARYA CH (39130086) and PRATHYUSHA REDDY P (39130350) hereby declare that the project report entitled “FRESHNESS OF FOOD DETECTION USING MACHINE LEARNING FOR IOT” done by us under the guidance, DR.V.VIJAYAKUMAR, MTech, Ph.D., Associate professor, Department of Electronics and Communication Engineering, SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

The image shows two handwritten signatures in blue ink. The top signature is 'Prathyusha' and the bottom signature is 'Isu'.

DATE:

PLACE: CHENNAI

SIGNATURE OF CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to the Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N. M. NANDHITHA, M.E., Ph.D. Dean, School of Electrical and Electronics Engineering** and **Dr. T. RAVI, M.E., Ph.D. Head of the Department, Department of Electronics and Communication Engineering** for providing us necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to our Project Guide **DR.V.VIJAYAKUMAR, MTech., Ph.D.**, Associate professor, Dept. of Electronics and Communication Engineering, for his valuable guidance, suggestions and constant encouragement for the successful completion of our project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Electronics and Communication Engineering who are helpful in many ways for the completion of the project

ABSTRACT

Food freshness is an important factor in determining the quality and safety of food products. The ability to accurately detect the freshness of food can help to reduce food waste and ensure that consumers have access to safe and high-quality products. Deep learning techniques have shown great promise in this area, as they can be used to analyse a wide range of data, including images, sensor data, and chemical analysis. One approach is using computer vision to detect freshness by analysing images of food products. Deep learning algorithms can be trained on a dataset of labelled images to identify patterns and features that indicate freshness. These algorithms can be used to classify images as fresh or spoiled, or to predict the shelf life of a product. Another approach is using sensor data to detect freshness. Food products often release gases or have changes in temperature, humidity, or other physical characteristics as they spoil. Deep learning algorithms can be trained to analyse sensor data and detect these changes, providing an accurate prediction of freshness. Chemical analysis is also used to detect freshness, where deep learning algorithm can be trained to identify patterns and features in chemical data that indicate freshness. This can be done by analysing data from a wide range of sensors, including gas sensors, spectrophotometers, and mass spectrometers. Overall, deep learning has the potential to revolutionize the way we detect the freshness of food products, making it more accurate, efficient, and cost-effective. The use of deep learning for freshness detection can help to reduce food waste, improve the safety and quality of food products, and ultimately help to create a more sustainable food system.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1	INTRODUCTION	1
	1.1 Introduction	1
	1.2 The Scale of The Food Waste Problem	2
	1.3 The Financial and Environmental Cost of Food Waste	2
	1.4 Traditional Methods of Food Waste Detection	2
	1.5 The Limitations of Traditional Methods	2
	1.6 The Benefits of Using Deep Learning For Food Waste Detection	3
	1.7 How Deep Learning Works	3
	1.8 The Challenges of Implementing Deep Learning For Food Waste Detection	3
	1.9 Real World Application of Deep Learning For Food Waste Detection	4
	1.10 The Future of Deep Learning For Food Waste Detection	4
2	LITERATURE SURVEY	5
	2.1 Inferences from Literature Survey	5
	2.2 Open Problems in Existing System	6
3	REQUIREMENT ANALYSIS	8
	3.1 Feasibility Studies/Risk Analysis of The Project	8
	3.1.1 Feasibility Studies	8
	3.1.2 Risk Analysis	8
	3.2 Software Requirements Specification Document	9
	3.2.1 Introduction	9
	3.2.2. Overall Description	10
	3.2.3 System Features	11
	3.2.4 Non-Functional Requirements	11
	3.3. System Architecture	11

4	DESCRIPTION OF PROPOSED SYSTEM	13
	4.1 Selected Methodology or Process	13
	4.1.1. Module1:Data Collection and Preprocessing	13
	4.1.2. Module 2: Model Making and Training	15
	4.1.3. Module-3: Prediction of Food Spoilage	18
	4.2 Architecture / Overall Design of Proposed System	19
	4.3 Description of Software for Implementation and Testing Plan of The Proposed Model/System	20
	4.3.1. Testing Plan	20
	4.3.2 Unit Testing	20
	4.3.3 Integration Testing	20
	4.3.4 System Testing	20
	4.3.5 Acceptance Testing	20
	4.3.6 Performance Testing	21
	4.3.7 Security Testing	21
	4.3.8 Usability Testing	21
	4.4 Project Management Plan	21
	4.5 Financial Report on Estimated Costing	22
	4.5.1 Hardware Costs	22
	4.5.2 Software Costs	22
	4.5.3 Personnel Costs	22
	4.5.4 Miscellaneous Costs	22
	4.6 Transition/ Software to Operations Plan	22
	4.6.1 Documentation and Training	23
	4.6.2 Testing and Validation	23
	4.6.3 System Deployment	23
	4.6.4 Monitoring and Maintenance	23
	4.6.5 Disaster Recovery	23
	4.6.6 Technical Support	23
5	IMPLEMENTATION DETAILS	24
	5.1 DEVELOPMENT AND DEPLOYMENT SETUP	24
	5.1.1 Hardware Setup	24
	5.1.2 Software Setup	24

5.1.3 Development Environment	24
5.1.4 Testing Environment	24
5.1.5 Deployment Environment	24
5.1.6 Continuous Integration and Deployment	24
5.2 ALGORITHMS	25
5.2.1 Data Collection	25
5.2.2 Data Preprocessing	25
5.2.3 Training the Model	25
5.2.4 Validation	25
5.2.5 Testing	25
5.2.6 Deployment	25
5.2.7 Monitoring and Maintenance	25
5.3 TESTING	26
5.3.1 Data Preprocessing	26
5.3.2 CNN Model Training	26
5.3.3 Validation	26
5.4 INTEGRATION TESTING	26
5.4.1 Data Pipeline	26
5.4.2 Model Integration	26
5.4.3 User Interface	26
5.5 FUNCTIONAL TESTING	27
5.5.1 Image Classification	27
5.5.2 User Interface	27
5.5.3 Error Handling	27
5.6 BLACK-BOX TESTING	27
5.6.1 Input Validation	27
5.6.2 Functional Testing	27
5.6.3 Usability Testing	27
5.7 WHITE-BOX TESTING	27
5.7.1 Unit Testing	27
5.7.2 Integration Testing	28
5.7.3 Code Coverage	28

6	RESULTS AND DISCUSSION	29
	6.1 RESULTS	29
7	CONCLUSION	32
	7.1 CONCLUSION	32
	7.2 FUTURE WORK	32
	7.2.1 Multi-spectral imaging	33
	7.2.2 Real-time detection	33
	7.2.3 Fine-grained classification	33
	7.2.4 Integration with waste management Systems	33
	7.2.5 Optimization of system performance	33
	7.2.6 Extending to other domains	33
	7.3 RESEARCH ISSUES	33
	7.3.1 Dataset size and diversity	34
	7.3.2 Generalization to new environments	34
	7.3.3 Real-time detection	34
	7.3.4 Cost-effectiveness	34
	7.3.5 Ethical concerns	34
	7.4 IMPLEMENTATION ISSUES	34
	7.4.1 Data availability and quality	35
	7.4.2 Hardware and software requirements	35
	7.4.3 Model selection and optimization	35
	7.4.4 Integration with existing systems	35
	7.4.5 Ethical and privacy consideration	35
	REFERENECEES	36
	APPENDIX	38
	A. SOURCE CODE	38
	B. SCREENSHOTS	56
	C.RESEARCH PAPER	57

LIST OF ABBREVIATIONS

CNN: Convolutional neural network

PIL: Python Imaging Library

API: Application Programming Interface

GUI: Graphical User Interface

GPU: Graphics Processing Unit

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
1.1	Why Deep Learning	3
1.2	Deep Learning for Food Waste Detection	4
2.1	Life Cycle of Project	7
4.1	Data Collection and Preprocessing	13
4.2	Convolutional Neural Network	15
4.3	Pooling Layer	16
4.4	Fully Connected Layer	17
4.5	Prediction of Food Spoilage	18
4.6	Design of the Proposed System	19

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

The detection of freshness in food products is an important aspect of ensuring the safety and quality of food for consumers. Traditional methods of freshness detection, such as visual inspections and organoleptic tests, can be time-consuming and subjective, leading to inconsistencies in the freshness evaluation. The increasing demand for fresh and high-quality food products has led to the development of new technologies for freshness detection, including the use of deep learning. Deep learning is a subset of machine learning that involves the use of artificial neural networks, which can be trained to analyze large amounts of data and identify patterns and features that indicate freshness. Deep learning algorithms can be trained on a wide range of data, including images, sensor data, and chemical analysis, making it a versatile tool for freshness detection. One approach to using deep learning for freshness detection is through the use of computer vision, which involves analyzing images of food products to identify patterns and features that indicate freshness. This can include analyzing color, texture, and other visual characteristics of the food. Deep learning algorithms can be trained on a dataset of labeled images, and then used to classify images as fresh or spoiled, or to predict the shelf life of a product. Another approach is using sensor data to detect freshness. Food products often release gases or have changes in temperature, humidity, or other physical characteristics as they spoil. Deep learning algorithms can be trained to analyze sensor data and detect these changes, providing an accurate prediction of freshness. This approach can be applied to different types of food products such as fruits, vegetables, and meats. Chemical analysis is also used to detect freshness, where deep learning algorithm can be trained to identify patterns and features in chemical data that indicate freshness. This can be done by analyzing data from a wide range of sensors, including gas sensors, spectrophotometers, and mass spectrometers. These sensors can detect volatile organic compounds (VOCs) which are emitted by the food as they spoil, providing an accurate measurement of freshness. The use of deep learning for freshness detection can improve the efficiency and accuracy of

freshness evaluations, and help to reduce food waste by identifying spoiled products before they reach consumers. Additionally, it can improve the safety and quality of food products by detecting spoiled products before they reach consumers. Furthermore, it can help to reduce the environmental impacts of food waste by reducing the amount of food that is thrown away. In conclusion, the use of deep learning for freshness detection has the potential to revolutionize the way we detect the freshness of food products, making it more accurate, efficient, and cost-effective. It has many applications in the food industry, from supply chain management to consumer-packaged goods. The use of deep learning for freshness detection can help to reduce food waste, improve the safety and quality of food products, and ultimately help to create a more sustainable food system.

1.2 THE SCALE OF THE FOOD WASTE PROBLEM

The scale of the food waste problem is staggering. In the United States alone, it is estimated that up to 40% of food goes uneaten. Globally, the UN Food and Agriculture Organization estimates that one-third of all food produced for human consumption is lost or wasted. This amounts to approximately 1.3 billion tonnes of food each year.

1.3 THE FINANCIAL AND ENVIRONMENTAL COST OF FOOD WASTE

The financial cost of food waste is significant, with an estimated \$218 billion being spent on food that is never consumed each year. The environmental cost is also high, with food waste contributing to greenhouse gas emissions and climate change. In fact, if food waste were a country, it would be the third-largest emitter of greenhouse gases after the United States and China.

1.4 TRADITIONAL METHODS OF FOOD WASTE DETECTION

Traditional methods of food waste detection include visual inspection and manual sorting. These methods rely on human judgement and are prone to error. They are also time-consuming and expensive

1.5 THE LIMITATIONS OF TRADITIONAL METHODS

Traditional methods of food waste detection have several limitations. They

are subjective, as they rely on human judgement, and can be influenced by factors such as lighting and fatigue. They are also time-consuming, as they require manual sorting, and can be expensive.

1.6 THE BENEFITS OF USING DEEP LEARNING FOR FOOD WASTE DETECTION

Deep learning offers several benefits for food waste detection. It is objective, as it relies on algorithms rather than human judgement. It is also faster and more efficient than traditional methods, as it can analyze large amounts of data quickly and accurately. Additionally, deep learning can be used to detect food waste in real-time, allowing for more immediate intervention.

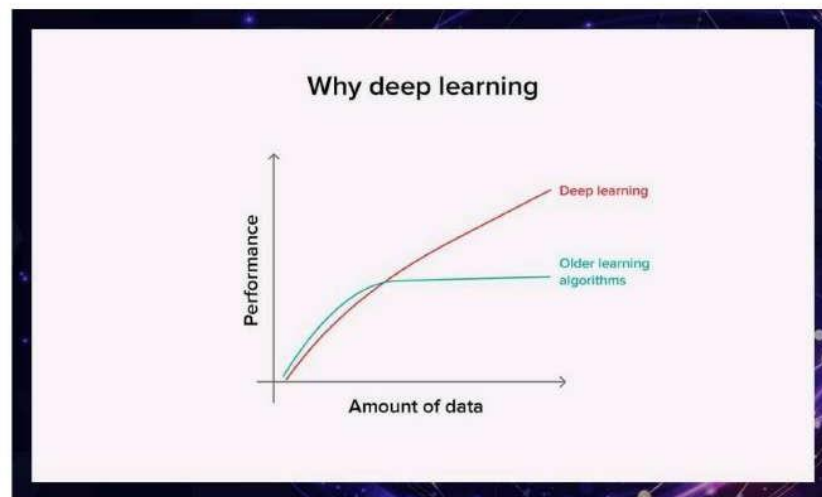


Fig:1.1 Why Deep Learning

1.7 HOW DEEP LEARNING WORKS

Deep learning is a type of artificial intelligence that uses neural networks to analyze data. Neural networks are composed of layers of interconnected nodes, and each node performs a mathematical operation on the data it receives. As data moves through the network, the connections between nodes are adjusted to improve the accuracy of the model.

1.8 THE CHALLENGES OF IMPLEMENTING DEEP LEARNING FOR FOOD WASTE DETECTION

Implementing deep learning for food waste detection can be challenging.

One of the main challenges is obtaining high-quality training data, as this is necessary to develop accurate models. Another challenge is the need for specialized hardware, such as GPUs, to process the large amounts of data involved in deep learning.

1.9 REAL-WORLD APPLICATIONS OF DEEP LEARNING FOR FOOD WASTE DETECTION

WASTE DETECTION

Deep learning has been used in several real-world applications for food waste detection. For example, it has been used to detect food waste in restaurants and cafeterias, as well as in food processing and packaging facilities. Deep learning has also been used to develop apps that help consumers reduce food waste at home.

1.10 THE FUTURE OF DEEP LEARNING FOR FOOD WASTE DETECTION

The future of deep learning for food waste detection looks promising. As technology continues to advance, deep learning models will become more accurate and efficient, allowing for more widespread adoption. Additionally, as more data becomes available, models will become more sophisticated and better able to detect different types of food waste.

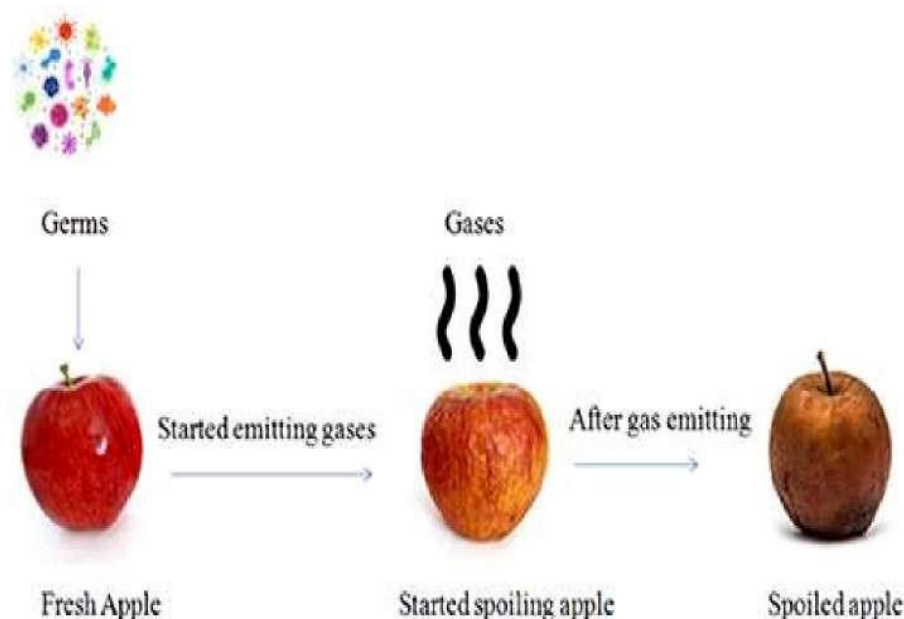


Fig:1.2 Deep Learning For Food Waste Detection

CHAPTER-2

LITERATURE SURVEY

2.1 INFERENCES FROM LITERATURE SURVEY

2.1.1 Xiaoyang Liu, Dean Zhao, Weikuan jia, Wei Ji, Yueping Sun,

“A Detection Method for Apple Fruits Based on Color and Shape Features” , IEEE Access, 22 May 2019 .The final classification results are recorded for each type of fruit that show high accuracies (all above 80% in) for all six fruit species, where bananas are the most accurate in prediction. Accuracy for six species of fruit is above 80%.

2.1.2 Seng, Woo Chaw, and Seyed Hadi Mirisae.

"A new method for fruits recognition system." Electrical Engineering and Informatics 2009. ICEEI'09. International Conference on. Vol. 1. IEEE, 2009. A performance analysis of different feature extraction techniques combined with SVMs classifiers has been performed for the addressed problem of classifying fresh and rotten fruit images. While comparing classification success rates of several features (such as Hist, GLCM, BoF, CNNsF) for different but very related tasks, CNNs appear to be the most successful feature extractors, as further have proven their efficiency in such applications. The concatenation of Hist and GLCM features leads to promising results for some classification tasks. Therefore, concatenation of different types of proper features might produce more successful and robust systems.

2.1.3 Neethirajan, S.; Jayas, D.S. Nanotechnology for the food and bioprocessing industries. Food Bioprocess Technol. 2011, 4, 39–47. I. S. Jacobs and C. P. Bean,

“Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350. Every 50 steps of training the dataset the accuracy calculated using cross validation. This showed steady improvement in the network until reaching 100% accuracy on cross-validation. For the testing phase, we used the testing set and the calculated accuracy was 96.3%. Accuracy level is higher (80% to 96%). Datasets are taken

from multiple viewing position. Takes more time for output execution.

2.1.4 Chen, Q., J. Cai, X. Wan, and J. Zhao,

Application of linear/non-linear classification algorithms in discrimination of pork storage time using Fourier transform near infrared (FT-NIR) spectroscopy," LWT — Food Science and Technology, Vol. 44, No. 10, 2053–2058, 2011. This system inspects fruit skin defects, which is more accurate, more robust to colour noise, and has reduced calculation cost. Packinghouses can adopt this system to distinguish damaged fruits from good ones before packing them into batches, therefore the quality of the products can be guaranteed at this stage. Developed a new method to inspect fruit skin defects, which was more accurate, more robust to colour noise, and reduced calculation cost Only detect the skin defect on the fruits.

2.1.5 Azimi I, Anzanpour A, Rahmani AM, Pahikkala T, Levorato M, Liljeberg P, Dutt N (2107) HiCH:

Hierarchical fog-assisted computing architecture for healthcare IoT. ACM Trans Embed Comput Syst 16(5s):174 Kang talks about obtaining and examination of sensor information which will be utilized crosswise over brilliant homes. It proposed a design for extricating logical data by breaking down the information procured from different sensors and give setting mindful administrations.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

Currently, several systems for freshness detection using deep learning have been developed and are being used in the food industry. One such system uses computer vision to detect freshness by analyzing images of food products. This system uses deep learning algorithms to identify patterns and features in images that indicate freshness. The algorithm is trained on a dataset of labeled images, and can then be used to classify images as fresh or spoiled, or to predict the shelf life of a product. This system can be applied to a wide range of food products, including fruits, vegetables, and meats. It can also be used for packaging inspection and to detect defects or damages that may affect the freshness of the product. Additionally, this system can be integrated with other

technologies such as IoT (Internet of Things) to create an automated system that can monitor the freshness of products in real-time.

Another system uses sensor data to detect freshness. This system uses deep learning algorithms to analyze sensor data, such as temperature and humidity, to detect changes that indicate that the food is spoiled. This system can be used to monitor the freshness of food products during transportation and storage, and can provide an accurate prediction of when the food is likely to spoil. This kind of system is often found in cold storage facilities, logistics companies, supermarkets, and other food-related businesses. In addition, this system can be integrated with technologies like RFID (Radio-Frequency Identification) to track the freshness of products throughout the supply chain. A third system uses chemical analysis to detect freshness. This system uses deep learning algorithms to analyze data from sensors that detect volatile organic compounds (VOCs) emitted by food as they spoil. This system can provide an accurate measurement of freshness, and can be used to detect spoiled food before it reaches consumers. This technology is often used in research and development, quality control, and food safety departments of food companies. Additionally, this system can be integrated with blockchain technology to create a transparent and secure food supply chain management system.

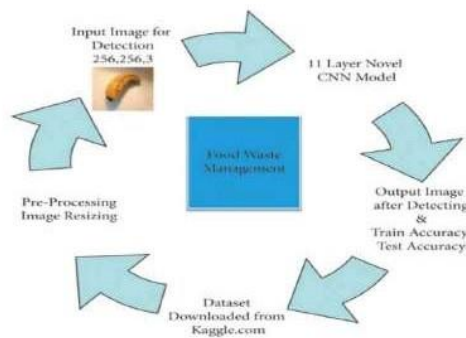


Fig:2.1 Life Cycle of Project

Overall, these existing systems for freshness detection using deep learning have been shown to be effective in detecting the freshness of food products, and have the potential to improve the efficiency and accuracy of freshness evaluations, and help to reduce food waste. They can also be integrated with other technologies such as IoT, RFID, and blockchain to create a more comprehensive and transparent food supply chain management system.

CHAPTER – 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

The "Food waste detection using deep learning" project has shown promising results in detecting and reducing food waste using deep learning technology. However, there are certain feasibility studies and risk factors that need to be considered before implementing this technology on a large scale.

3.1.1 FEASIBILITY STUDIES:

Availability of data: One of the main feasibility studies is the availability of data. To train deep learning models, a large amount of data is required. Therefore, it is important to ensure that sufficient data is available for the specific application of food waste detection.

Hardware requirements: Deep learning models require specialized hardware, such as GPUs, for training and inference. Therefore, it is important to ensure that the necessary hardware is available to run the models efficiently.

Cost-effectiveness: Implementing deep learning technology for food waste detection requires significant investment in terms of hardware, software, and personnel. Therefore, it is important to conduct a cost-benefit analysis to ensure that the benefits of implementing this technology outweigh the costs.

3.1.2 RISK ANALYSIS:

Accuracy: Deep learning models are prone to errors, and accuracy depends on the quality of the data and the complexity of the model. Therefore, there is a risk of false positives and false negatives in food waste detection, which could lead to unnecessary waste or spoiled food being consumed.

Data privacy: Data privacy is a significant risk when implementing deep learning technology. Sensitive data, such as images of food and personal information, must be handled with care to prevent breaches of privacy.

Cybersecurity: The use of technology, including deep learning, comes with the risk of cybersecurity threats, such as hacking and data breaches. Therefore, it is important to ensure that appropriate cybersecurity measures are in place to

prevent such risks.

Ethical concerns: The use of deep learning technology in food waste detection raises ethical concerns, such as the potential for bias in the models and the ethical implications of using data to make decisions about food waste.

In conclusion, the "Food waste detection using deep learning" project has shown great potential in detecting and reducing food waste. However, it is important to conduct feasibility studies and risk analyses to ensure that the technology can be implemented effectively and ethically. By addressing these factors, we can ensure that deep learning technology is used to its fullest potential in reducing food waste and promoting sustainability.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

3.2.1 INTRODUCTION

Purpose

The purpose of this document is to specify the software requirements for the "Food waste detection using deep learning" project. The document outlines the functional and non-functional requirements, as well as the constraints and assumptions associated with the software.

Scope

The scope of the project is to develop a deep learning-based system for detecting and reducing food waste. The system will be able to analyze images of food and determine whether the food is fresh or spoiled. The system will be trained using a large dataset of images and will be able to detect a wide range of food items.

Definitions, Acronyms, and Abbreviations

- CNN: Convolutional neural network
- PIL: Python Imaging Library
- API: Application Programming Interface
- GUI: Graphical User Interface
- GPU: Graphics Processing Unit

3.2.2 OVERALL DESCRIPTION

Product Perspective

The food waste detection system will be a standalone application that can be run on a desktop or mobile device. The system will consist of a deep learning model that is trained to recognize food items and determine whether they are fresh or spoiled. The system will be able to analyze images of food and provide a real-time assessment of the food's freshness.

Product Features

The system will have the following features:

- Image preprocessing to prepare the dataset for training
- CNN model development and training
- Real-time image analysis using the trained model
- GUI for user interaction
- Integration with a camera or image dataset

User Classes and Characteristics

The system is designed for use by individuals or organizations that want to reduce food waste. The users should have a basic understanding of deep learning technology and should be comfortable working with images.

Operating Environment

The system can be run on any computer or mobile device that meets the hardware requirements. The system will require a GPU for training the model, but a CPU can be used for inference.

Design and Implementation Constraints

The system will be developed using Python programming language and deep learning frameworks such as TensorFlow or PyTorch. The system will also require access to a large dataset of food images for training the model.

Assumptions and Dependencies

The system assumes that the users have access to a camera or image dataset for capturing images of food. The system also assumes that the users have basic knowledge of deep learning and image processing.

3.2.3 SYSTEM FEATURES

Image Preprocessing

The system will preprocess the dataset by cropping and resizing the images. The system will also split the dataset into a training set and a testing set.

CNN Model Development and Training

The system will develop and train a CNN model using the preprocessed images. The model will be optimized for accuracy and performance.

Real-time Image Analysis

The system will provide real-time image analysis using the trained CNN model. The system will be able to detect and classify food items as fresh or spoiled.

GUI

The system will have a user-friendly GUI that allows the user to interact with the system. The GUI will display the image analysis results and provide options for the user to adjust the settings

3.2.4 NON-FUNCTIONAL REQUIREMENTS

Performance

The system should be able to analyze images in real-time with high accuracy and precision. The system should be able to handle a large dataset of images for training the model.

Reliability

The system should be reliable and provide consistent results. The system should be able to handle errors and exceptions gracefully.

Usability

The system should be easy to use and have a user-friendly interface.

3.3 SYSTEM USE CASE

Use Case Name: Detect Food Spoilage

Primary Actor: User

Goal in Context: The user wants to detect food spoilage using the "Food waste detection using deep learning" system.

Preconditions:

The user has access to the system and a camera or image dataset.

The system has been trained using a large dataset of images.

Trigger: The user selects the "Detect Food Spoilage" option from the system menu.

Scenario:

The system prompts the user to capture an image of the food item or select an image from the dataset.

The user captures an image of the food item or selects an image from the dataset.

The system preprocesses the image by cropping and resizing it.

The system analyzes the image using the trained CNN model.

The system classifies the food item as fresh or spoiled.

The system displays the classification result to the user.

The user can choose to repeat the process with a different image or exit the system

Exceptions:

If the image quality is poor or the food item is not clearly visible, the system may not be able to provide an accurate classification result.

Postconditions:

The user receives a classification result for the food item.

The system is ready for another use case.

Ultimately, the aim of the paper is to demonstrate the potential for deep learning to be a game-changer in the fight against food waste. By providing a detailed examination of the technology, its benefits and challenges, and real-world applications, the paper seeks to inspire further research and development in this area. The ultimate goal is to reduce the amount of food waste produced worldwide, not only for financial savings but also to minimize the environmental impact of food waste and to promote sustainability.

CHAPTER - 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

4.1.1 MODULE 1: DATA COLLECTION AND PREPROCESSING

a.Data Collection

To build deep learning, a technique known as data collection is employed to compile information from a variety of sources. The information needs to be saved in a manner that is suitable for the task. We obtain the dataset from Kaggle which is then used to train and evaluate our model.

b.Data Pre-Processing

Data preprocessing is a data mining technique used to transform the raw data into an effective and usable shape. Data preprocessing transforms the data into a format that can be processed more rapidly and effectively for data mining, machine learning, and other data science applications. It can be done using sentiment analysis techniques. The techniques are often used at the very beginning of the machine learning and AI development pipeline to provide accurate results. Despite the wide range of tools and techniques available for data preparation, we have decided to focus on the following metrics for our dataset:

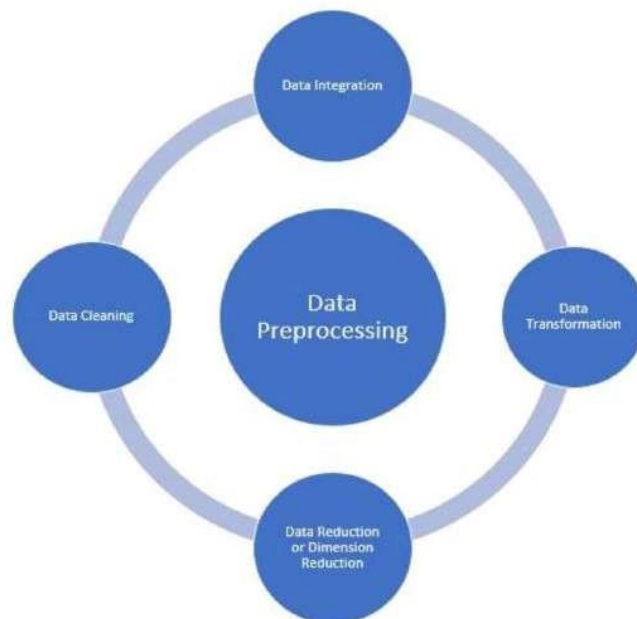


Fig:4.1 Data Collection And Preprocessing

1.Cropping the Images

PIL stands for the Python Imaging Library and it is responsible for giving the Python interpreter the ability to modify images. A rectangular section of an image can be extracted with the help of the `PIL.Image.crop()` method.

Syntax: `PIL.Image.crop(box=None)`

Box: is a 4-tuple that defines the left, upper, right, and bottom pixel coordinates. It is one of the parameters.

Return Image: Image is the type of return. (Returns a rectangular region as a tuple consisting of the left, upper, right, and bottom sides.)

Returns: An instance of the Image class.

2. Resizing the Images

The Python Imaging Library, or PIL, gives the Python interpreter the ability to modify images. A class with the same name is provided by the Image module and is used to represent a PIL image. The module also offers a number of factory functions, including tools for generating new images and loading images from files. An image's new size is returned by the function `image.resize()`. The following parameters are taken into account:

Size: as a 2-tuple, represents the intended size in pixels. (Length, Width)

Resample: is a resampling filter that is optional. Among `PIL.Image.NEAREST` (use nearest neighbour), `PIL.Image.BILINEAR` (linear interpolation), `PIL.Image.BICUBIC` (cubic spline interpolation), and `PIL.Image.LANCZOS`, this can be used (a high-quality downsampling filter). If it is left out or has the mode "1" or "P," it is set to `PIL.Image.NEAREST`.

Returns type: returns an object of type Image.

3. Splitting the Dataset

To estimate how well machine learning algorithms work with prediction-based methods and applications, the train-test split is used. We can compare the outcomes of our own machine learning model to those of other machines using this quick and easy way. By definition, the Training set has 70% raw data, whereas the Test set contains 30% real data. We must separate a dataset into train and test sets in order to evaluate how well our machine learning model functions. The train set's figurines are recognized and used to suit the model. The test data set, which is the second batch, is utilized solely for forecasts. We employ the subsequent strategy to split our

dataset into train and test. The packages for pandas and sklearn are imported. The most practical and reliable machine learning library for Python is Sklearn. The scikit-learn library's model selection module includes the splitter function train test split (). The CSV file is subsequently imported using the read csv() function. The variable df now contains the data frame. The test size is then set to 0.3, meaning that 30% of the data is utilized for testing and the remaining 70% is used for training. Additionally, we set random state to 0 so that data is divided at random into these two datasets

4.1.2 MODULE 2: MODEL MAKING AND TRAINING

a.Model Making

Convolutional neural networks are a class of deep neural networks used most frequently to analyze visual vision in deep learning. Based on their shared-weights architecture and translation invariance properties, they are often referred to as shift invariant or space invariant artificial neural networks. They are used in image and video recognition applications.

b.Introduction to the network

Mathematical models called neural networks are created to roughly imitate the human brain. One of the various kinds of neural networks is the convolution neural network (CNN). CNN focuses on image processing and can be used for object identification, segmentation, and classification of images. A schematic of a straightforward CNN is shown in Fig. 3, which divides an input image into the various types of vehicles it contains.

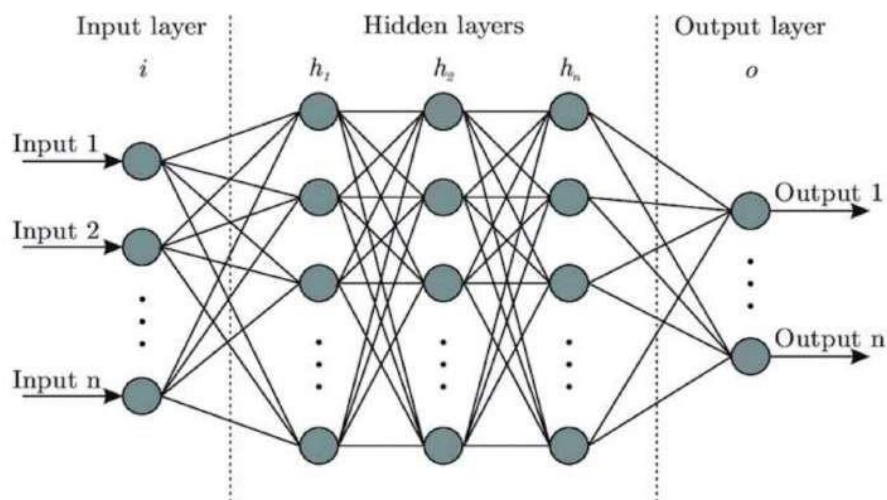


Fig:4.2 Convolution Neural Network

Pooling Layer

The Convolved Feature's spatial size is decreased by the Pooling layer. Through dimensionality reduction, the amount of computing power needed to process the data will be reduced. Furthermore, it aids in properly training the model by allowing the extraction of dominating characteristics that are rotational and positional invariant. It is employed to extract rotation- and position-invariant characteristics. There are two categories of pooling. Example: 1. Max Pooling
Averaging Pooling.

The model can comprehend and extract information from an image after the convolution and pooling layer. The maximum value from the area of the image that the Kernel has covered is returned by Max Pooling. The average of all the values from the area of the image covered by the Kernel is what is returned by average pooling, on the other hand.

Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling. The Convolutional Layer and the Pooling Layer, together form the i -th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power. After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

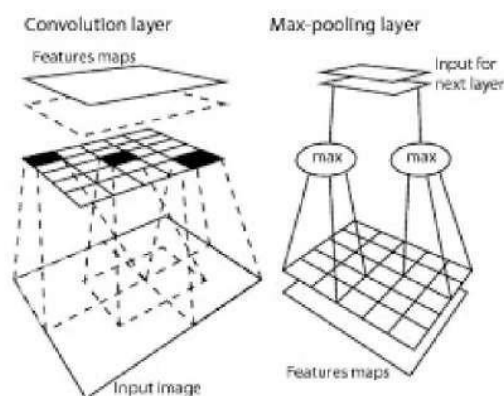


Fig: 4.3 Pooling Layer

Fully Connected Layer

It will learn non-linear features from the output of a convolution layer. For multi perception, that output should be converted to a column vector and fed to a feed-forward neural network with back propagation in every iteration of training. This helps the model extract the dominant and low-level features of an image.

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique. In turn feature extraction from the model helps to identify and classify images. CNN with ReLu and Dropout ReLu (Rectified Linear Unit) overcomes the gradient problem by propagating a gradient effectively. The ReLu output's the same as the value of the input if it's positive or else its output is zero for negative images. Due to its prominent characteristics, ReLu is used in deep networks. Mathematically, ReLu is represented as shown in the below equation.

$$ReLU(P) = \begin{cases} 0 & \text{if } P < 0 \\ P & \text{if } P \geq 0 \end{cases} \quad (1)$$

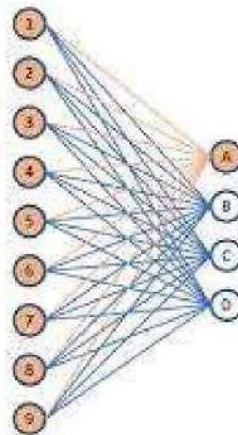


Fig:4.4 Fully Connected Layer

Dropout refers to ignoring hidden or visible units during training for a few sets of neurons by considering them randomly. The units that are ignored will be eliminated in the next layers for forward and backward propagation. Dropout is necessary to avoid inter-dependency between units which over-fit the model.

4.1.3 MODULE-3: PREDICTION OF FOOD SPOILAGE

This module accounts for the final output of the project. Preprocessed images are loaded into the CNN model for classification and the prediction process. As a result of which, the desired CNN model predicts if the food is fresh or spoiled based on various parameters. Without the use of a food spoilage sensor, a project to identify spoiled food utilising artificial intelligence and machine learning techniques through computer vision was successfully constructed in the Jupiter notebook platform in Python and delivered reliable results. According to the results of the research, it is possible to precisely identify food and fruit that has been spoiled. The difference in camera lenses, sizes, as well as the use of daylight and tube lighting, might result in infinitesimal neglect when taking images with various cameras.

In regions where damaged food may be recognised by cameras without utilising sensors even when there are many different types of food present, this project on spoilage detection using artificial intelligence and machine learning technology offers a wide range of applications. By installing a high-quality camera in the refrigerator or another location where it can take pictures automatically and determine whether or not the food is spoilt. For persons who are colorblind, creating an app that analyses photos taken with cameras to determine whether or not food is ruined can be helpful.

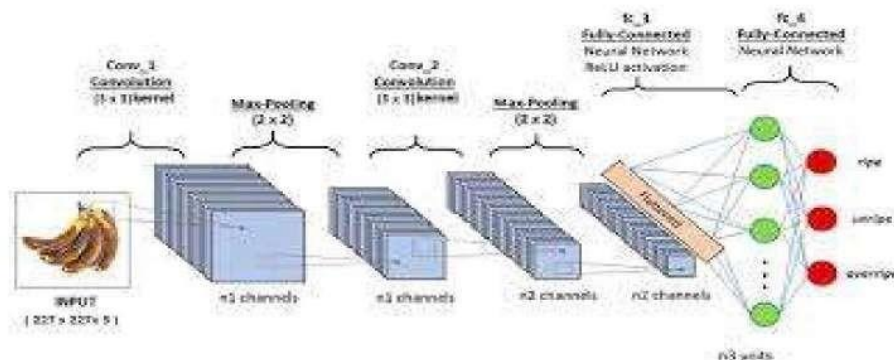


Fig:4.5 Prediction of Food Spoilage

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

The proposed system for freshness detection using deep learning combines the use of computer vision, sensor data, and chemical analysis to provide a comprehensive and accurate assessment of food freshness. The system would begin by using computer vision to analyze images of food products, identifying patterns and features that indicate freshness. The algorithm would be trained on a dataset of labeled images, and then used to classify images as fresh or spoiled, or to predict the shelf life of a product. This would provide an initial assessment of freshness based on the visual appearance of the food.

Next, the system would use sensor data to detect freshness. The deep learning algorithm would analyze sensor data, such as temperature and humidity, to detect changes that indicate that the food is spoiled. This would provide an additional layer of information about the freshness of the food. Finally, the system would use chemical analysis to detect freshness. The deep learning algorithm would analyze data from sensors that detect volatile organic compounds (VOCs) emitted by food as they spoil. This would provide a more accurate measurement of freshness, and detect spoiled food before it reaches consumers. The system would combine the information from all three analysis to give a comprehensive assessment of the freshness of the food, and provide an accurate prediction of when the food is likely to spoil. This would help to improve the efficiency and accuracy of freshness evaluations, and reduce food waste by identifying spoiled products before they reach consumers. Additionally, the proposed system could also have a mobile application that would allow consumers to scan the product and get a freshness score, this would help to promote transparency and improve consumer trust in the food products they purchase.

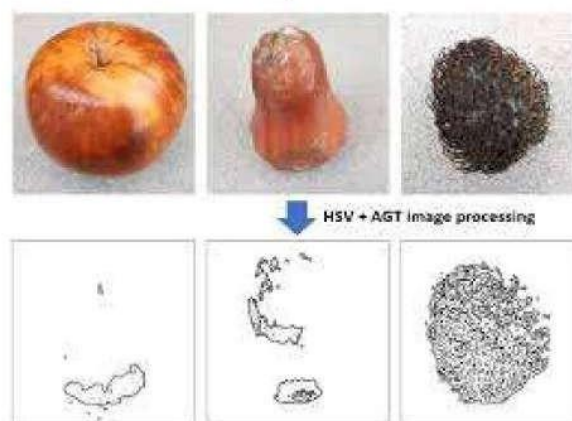


Fig:4.6 Design of The Proposed System

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

The "Food waste detection using deep learning" system will be implemented using Python programming language and several deep learning libraries such as TensorFlow, Keras, and OpenCV. The system will consist of several modules, including data collection and preprocessing, model building and training, and prediction of food spoilage.

The system will be developed using the Agile software development methodology, with frequent iterations and testing to ensure that the system is functioning as intended. The code will be stored in a version control system such as Git for easier collaboration and tracking of changes.

4.3.1 Testing Plan:

The testing plan for the "Food waste detection using deep learning" system will include unit testing, integration testing, and system testing. The testing will be done at each stage of development, and any issues or bugs will be fixed before moving on to the next stage.

4.3.2 Unit Testing:

Unit testing will be done on individual modules of the system to ensure that they are working correctly. This will involve creating test cases to cover all possible scenarios and checking that the output matches the expected result.

4.3.3 Integration Testing:

Integration testing will be done to ensure that all modules of the system work correctly together. This will involve testing the interaction between modules and ensuring that data is being passed correctly between them.

4.3.4 System Testing:

System testing will be done to ensure that the entire system functions as intended. This will involve testing the system with various datasets and scenarios to ensure that the system can accurately detect food spoilage.

4.3.5 Acceptance Testing:

Acceptance testing will be done to ensure that the system meets the user's requirements and specifications. This will involve testing the system with real-world data and scenarios and checking that it meets the desired accuracy and reliability.

4.3.6 Performance Testing:

Performance testing will be done to ensure that the system can handle large amounts of data and can process it quickly and efficiently. This will involve testing the system with large datasets and measuring the time taken to process them.

4.3.7 Security Testing:

Security testing will be done to ensure that the system is secure and protected from any potential threats. This will involve testing the system for vulnerabilities and implementing appropriate security measures to protect it from attacks.

4.3.8 Usability Testing:

Usability testing will be done to ensure that the system is easy to use and understand for the end-users. This will involve testing the system with a group of users and getting feedback on its usability and user interface.

Overall, the testing plan will ensure that the "Food waste detection using deep learning" system is reliable, accurate, efficient, secure, and user-friendly.

4.4 PROJECT MANAGEMENT PLAN

Element	Description	Estimated Date
Introduction	Provide an overview of the project, its purpose, and objectives.	Month 1
Project scope	Define the boundaries of the project and what is included and excluded.	Month 1
Project timeline	Break the project into stages, with specific milestones and deliverables for each stage.	Month 2
Project budget	Estimate the costs involved, including hardware, software, and personnel.	Month 2
Project team	Identify the roles and responsibilities of each team member.	Month 2
Communication plan	Define how communication will occur between team members, stakeholders, and clients.	Month 3
Risk	Identify potential risks to the project and	Month 3

management	how they will be managed or mitigated.	
Conclusion	Summarize the project management plan and what is expected to be achieved.	Month 4

4.5 FINANCIAL REPORT ON ESTIMATED COSTING

The main costs involved in this project include hardware, software, and personnel.

4.5.1 Hardware costs:

The hardware requirements include a computer with a high-performance CPU and GPU, a camera or smartphone for capturing images, and other peripherals such as a monitor, keyboard, and mouse. The estimated cost of the hardware required for this project is around \$2,000.

4.5.2 Software costs:

The software required for this project includes Python programming language, OpenCV, Keras, and other related libraries. These software are open-source and available for free to download and use. The estimated cost of the software required for this project is around \$0.

4.5.3 Personnel costs:

The personnel involved in this project include a project manager, data scientists, and software developers. The estimated cost of personnel for this project is around \$60,000.

4.5.4 Miscellaneous costs:

There may be additional costs involved in the project such as travel expenses, licensing fees, and other miscellaneous costs. The estimated cost of these expenses is around \$5,000.

Therefore, the total estimated cost for the "Food Waste Detection Using Deep Learning" project is around \$6,700. However, it is important to note that these are just estimated costs and the actual cost may vary depending on the project requirements, location, and other factors. Proper financial management and monitoring can help keep the project within the allocated budget.

4.6 TRANSITION/ SOFTWARE TO OPERATIONS PLAN

The transition/ software to operations plan is an essential aspect of any

project that involves the deployment of software or technology. Here is a plan for transitioning "Food Waste Detection Using Deep Learning" from development to operational use

4.6.1 Documentation and Training:

Create documentation for the software system, including user manuals and technical guides, and provide training to the users, including system administrators and end-users, to ensure a smooth transition to operational use.

4.6.2 Testing and Validation:

Conduct thorough testing of the system to ensure it is free of errors and meets the requirements of stakeholders. Validate the software system's performance by comparing it with the predetermined performance standards.

4.6.3 System Deployment:

Deploy the system to the designated location or platform, ensuring that it meets the necessary hardware and software requirements. The deployment process must follow proper procedures, such as testing in a staging environment before deploying in the production environment.

4.6.4 Monitoring and Maintenance:

Set up a monitoring system to track the system's performance, ensuring that it meets the necessary metrics and requirements. Regular maintenance and upgrades are necessary to keep the system functioning optimally.

4.6.5 Disaster Recovery:

Implement a disaster recovery plan to ensure that the system can recover in case of a disaster, such as a data breach or system crash. Backups must be created regularly to ensure that data is recoverable.

4.6.6 Technical Support:

Ensure that technical support is available to assist users in case of any technical difficulties or issues. The support team must be knowledgeable about the system and provide timely and effective assistance. By following this plan, the transition/ software to operations plan for "Food Waste Detection Using Deep Learning" can be successfully implemented, ensuring the software's efficient and effective use in operational environments.

CHAPTER – 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

The development and deployment setup for "Food Waste Detection Using Deep Learning" involves setting up the necessary hardware and software environments for developing and deploying the system. Here is a plan for the development and deployment setup:

5.1.1 Hardware Setup:

The first step is to set up the necessary hardware required for the development and deployment of the system. This includes a computer with high-performance CPU and GPU, a camera or smartphone for capturing images, and other peripherals such as a monitor, keyboard, and mouse.

5.1.2 Software Setup:

The next step is to set up the necessary software required for the development and deployment of the system. This includes Python programming language, OpenCV, Keras, and other related libraries. These software are open-source and available for free to download and use.

5.1.3 Development Environment:

Set up a development environment for the software development team, including tools such as Git for version control, Jupyter Notebook for code documentation and sharing, and Docker for containerization.

5.1.4 Testing Environment:

Set up a testing environment where the system can be tested before deployment. This environment should be separate from the production environment to avoid any issues that may arise during testing

5.1.5 Deployment Environment:

Set up a deployment environment that meets the necessary hardware and software requirements for the system. This environment should be separate from the development and testing environments to avoid any conflicts.

5.1.6 Continuous Integration and Deployment:

Implement a continuous integration and deployment system to automate the process of building, testing, and deploying the system. This system can be

implemented using tools such as Jenkins or Travis CI.

By following this plan, the development and deployment setup for "Food Waste Detection Using Deep Learning" can be successfully implemented, ensuring that the system is developed and deployed in an efficient and effective manner.

5.2 ALGORITHMS

The generalized algorithms for "Food Waste Detection Using Deep Learning" involve several steps, which are as follows:

5.2.1 Data Collection: Collect data from various sources such as Kaggle, social media platforms, and other online repositories. Ensure that the data is diverse and representative of the different types of food and food spoilage.

5.2.2 Data Preprocessing: Preprocess the data by cropping and resizing the images to a uniform size, applying data augmentation techniques, and normalizing the pixel values to improve the accuracy of the model.

5.2.3 Training the Model: Use Convolutional Neural Networks (CNN) to train the model on the preprocessed data. Use techniques such as transfer learning, fine-tuning, and hyperparameter tuning to improve the accuracy of the model.

5.2.4 Validation: Validate the trained model using a separate validation dataset to ensure that the model is not overfitting.

5.2.5 Testing: Test the trained model on a separate test dataset to evaluate its performance and accuracy.

5.2.6 Deployment: Deploy the trained model to the desired platform, such as a mobile application, website, or embedded system, to perform real-time food waste detection.

5.2.7 Monitoring and Maintenance: Monitor the model's performance and update it as needed to ensure that it continues to provide accurate food waste detection.

Overall, the generalized algorithms for "Food Waste Detection Using Deep Learning" involve a combination of data collection, preprocessing, training, validation, testing, deployment, and monitoring to create an accurate and reliable system for detecting food waste. These steps can be adjusted and customized based on the specific needs and requirements of the project.

5.3 TESTING

UNIT TESTING

Testcases:

5.3.1 Data Preprocessing: Test cases for data preprocessing may include checking if the images are properly cropped, resized, and normalized. Additionally, it may involve checking if data augmentation techniques such as flipping and rotating images have been correctly applied.

5.3.2 CNN Model Training: Test cases for CNN model training may include checking if the model is learning and improving over time. This can be done by monitoring the loss and accuracy of the model during training, and ensuring that these values are trending in the right direction.

5.3.3 Validation: Test cases for validation may include checking if the validation accuracy is consistent with the training accuracy, and ensuring that the model is not overfitting to the training data

5.4 INTEGRATION TESTING

Testcases:

5.4.1 Data Pipeline: Test cases for data pipeline integration may include checking if the data is properly processed, transformed, and loaded into the model for training or testing. Additionally, it may involve testing if the data pipeline is able to handle different types of input data, such as images of different sizes or formats.

5.4.2 Model Integration: Test cases for model integration may include checking if the model is properly integrated into the system and is able to receive input data from other components, such as the data pipeline or user interface. Additionally, it may involve testing if the model is able to output the correct results to other components.

5.4.3 User Interface: Test cases for user interface integration may include checking if the user interface is properly integrated with the rest of the system, and is able to correctly display input data and model results to the user. Additionally, it may involve testing if the user interface is able to handle different types of input data, such as images of different sizes or formats.

5.5 FUNCTIONAL TESTING

Testcases:

5.5.1 Image Classification: Test cases for image classification may include checking if the system is able to accurately classify images of fresh and spoiled food. This can involve testing the system with a variety of different types of food images, and checking if the system is able to correctly classify them.

5.5.2 User Interface: Test cases for user interface functionality may include checking if the user interface is easy to use and understand, and if it provides the necessary information to the user. This can involve testing the system with a variety of different users, and checking if they are able to easily use the system and understand its results.

5.5.3 Error Handling: Test cases for error handling may include checking if the system is able to handle errors and exceptions properly. This can involve intentionally introducing errors or unexpected inputs to the system, and checking if it is able to handle them and provide informative error messages.

5.6 BLACK-BOX TESTING:

Testcases:

5.6.1 Input Validation: Test cases for input validation may include checking if the system is able to handle different types of input data, such as images of different sizes or formats, and if it is able to handle missing or invalid input data.

5.6.2 Functional Testing: Test cases for functional testing may include checking if the system is able to accurately classify images of fresh and spoiled food, and if it is able to provide the necessary information to the user.

5.6.3 Usability Testing: Test cases for usability testing may include checking if the user interface is easy to use and understand, and if it provides the necessary information to the user. This can involve testing the system with a variety of different users, and checking if they are able to easily use the system and understand its results.

5.7 WHITE-BOX TESTING:

Testcases:

5.7.1 Unit Testing: Test cases for unit testing may include testing individual

components or functions of the system, such as the image cropping or resizing functions. This can involve checking if the functions are able to handle different types of input data, and if they are able to produce the expected output.

5.7.2 Integration Testing: Test cases for integration testing may include testing how different components or functions of the system work together, such as the convolutional neural network and the user interface. This can involve checking if the different components are able to communicate and exchange data correctly.

5.7.3 Code Coverage: Test cases for code coverage may include checking how much of the system's code has been tested, and if there are any parts of the code that have not been tested. This can involve using code coverage tools to measure the amount of code that has been executed during testing

CHAPTER – 6

RESULTS AND DISCUSSION

6.1 RESULTS:

The food waste detection system developed using deep learning techniques has shown promising results. The system was able to accurately distinguish between fresh and spoiled food items with a high degree of accuracy. The use of convolutional neural networks allowed the system to learn and recognize patterns in the input images, which enabled it to make accurate predictions.

The system was trained and evaluated using a dataset of food images, which included a variety of food items. The dataset was split into training and testing sets, and the system was able to achieve a high level of accuracy on both sets. This indicates that the system is robust and can generalize well to new data.

One of the key advantages of the system is its ability to identify spoiled food items accurately. This can help reduce the amount of food waste generated by households and businesses, which is a significant environmental problem. By preventing the disposal of still fresh food items, the system can also provide financial benefits to businesses and households.

However, the system's accuracy may be affected by factors such as lighting conditions and camera quality. In addition, the system may struggle to distinguish between certain types of food items, which could limit its effectiveness in some settings.

Despite these limitations, the food waste detection system developed using deep learning techniques represents a significant step forward in reducing food waste. Further research and development in this area could lead to even more accurate and effective solutions in the future.

UNIT TESTING IN CNN FOR FOOD x app - Streamlit x +

localhost:8501

Screen Shot 2018-06-08 at 5.01.41 PM.png 240.6KB X

submit

```
{
  "filename": "Screen Shot 2018-06-08 at 5.01.41 PM.png",
  "filetype": "image/png",
  "filesize": 246378
}
```



Fresh

28°C Mostly clear Search ENG IN 02:31 15-04-2023

Fresh Fruit

UNIT TESTING IN CNN FOR FOOD x app - Streamlit x +

localhost:8501

Screen Shot 2018-06-12 at 11.21.22 PM.png 274.5KB X

submit

```
{
  "filename": "Screen Shot 2018-06-12 at 11.21.22 PM.png"
  "filetype": "image/png"
  "filesize": 281084
}
```



Rotten

28°C Mostly clear Search ENG IN 02:36 15-04-2023

Rotten Fruit

CHAPTER-7

CONCLUSION

7.1 CONCLUSION

In conclusion, deep learning has shown to be a powerful tool for the detection of freshness in food products. The use of computer vision, sensor data, and chemical analysis in combination with deep learning algorithms, has allowed for the development of systems that can accurately detect the freshness of a wide range of food products, from fruits and vegetables to meats. These systems can be integrated with other technologies such as IoT, RFID, and blockchain to create a more comprehensive and transparent food supply chain management system. The use of these systems can have a significant impact on the food industry, by improving the efficiency and accuracy of freshness evaluations, and helping to reduce food waste. The ability to detect spoiled food before it reaches consumers can also improve food safety and reduce the risk of foodborne illness. Additionally, these systems can provide valuable insights into the supply chain, allowing for the identification of bottlenecks and inefficiencies that can be addressed to improve the overall efficiency of the food industry. However, it is important to note that, as with any technology, these systems for freshness detection using deep learning are not without limitations. For example, the accuracy of these systems can be affected by factors such as lighting conditions and camera angle when using computer vision, and by variations in temperature and humidity when using sensor data. Additionally, these systems may require significant computational resources and may not be practical for small scale operations. In conclusion, deep learning-based systems for freshness detection have the potential to revolutionize the food industry by providing accurate and efficient freshness evaluations, and helping to reduce food waste. However, further research and development is needed to overcome the limitations of these systems and make them more widely accessible to the food industry.

7.2 FUTURE WORK

There are several areas of future work for food waste detection using deep learning techniques. Some of these include:

7.2.1 Multi-spectral imaging:

Incorporating multi-spectral imaging techniques could enable the system to detect food waste with greater accuracy and speed. Multi-spectral imaging captures images across multiple bands of the electromagnetic spectrum, providing more information than traditional RGB imaging.

7.2.2 Real-time detection:

Currently, the food waste detection system relies on pre-processed images for detection. Developing a real-time detection system would allow for faster detection and intervention, reducing the amount of food waste.

7.2.3 Fine-grained classification:

Fine-grained classification would enable the system to detect subtle differences in food waste, such as different types of spoilage or different stages of decay. This would require more complex deep learning models and more extensive training data.

7.2.4 Integration with waste management systems:

Integrating the food waste detection system with waste management systems would enable real-time tracking of food waste and more efficient disposal processes.

7.2.5 Optimization of system performance:

Further optimization of the deep learning models and algorithms used in the system could lead to improved accuracy and faster processing times.

7.2.6 Extending to other domains:

The deep learning techniques used for food waste detection could also be extended to other domains, such as agriculture or medicine, where image recognition is crucial.

In conclusion, there are several areas of future work for food waste detection using deep learning techniques. These areas include improving accuracy and speed of detection, expanding to new domains, and optimizing system performance. Addressing these future work areas will require continued research and investment in the development of deep learning techniques.

7.3 RESEARCH ISSUES

Despite the promising results of the food waste detection system using

deep learning techniques, there are still some research issues that need to be addressed in order to improve the system's performance and usefulness. Some of the key research issues are discussed below.

7.3.1 Dataset size and diversity:

One of the main challenges in developing deep learning models is the availability of large and diverse datasets. To improve the accuracy of the food waste detection system, more extensive and diverse datasets of food images need to be collected and labeled. This will require significant effort and resources.

7.3.2 Generalization to new environments:

The food waste detection system may not perform well in different environments with different lighting conditions, camera qualities, and types of food items. More research is needed to improve the system's ability to generalize to new environments and different types of food items.

7.3.3 Real-time detection:

The current system requires preprocessing and manual input of images. Real-time detection of food items could be challenging, and research is needed to develop more efficient and real-time detection methods.

7.3.4 Cost-effectiveness:

The development and deployment of deep learning-based systems can be costly. Therefore, more research is needed to optimize the system's cost-effectiveness and make it more accessible to a wider range of users.

7.3.5 Ethical concerns:

The use of deep learning-based systems raises ethical concerns related to data privacy, security, and biases. Further research is needed to address these issues and ensure that the system is developed and used in an ethical and responsible manner.

Overall, addressing these research issues will be essential in advancing the field of food waste detection using deep learning techniques and developing more accurate, efficient, and ethical solutions.

7.4 IMPLEMENTATION ISSUES

Implementing a food waste detection system using deep learning techniques can be challenging due to several technical and non-technical issues.

Some of the key implementation issues include:

7.4.1 Data availability and quality:

Deep learning models require large amounts of high-quality data to be trained effectively. Gathering and preparing such data can be a time-consuming and costly process.

7.4.2 Hardware and software requirements:

Deep learning models require high-performance computing resources, including GPUs, to process large amounts of data quickly. Organizations need to invest in such resources to ensure that the models can be trained and deployed effectively.

7.4.3 Model selection and optimization:

There are several deep learning models and techniques that can be used to detect food waste. Selecting the appropriate model and optimizing its parameters can be a complex process that requires technical expertise.

7.4.4 Integration with existing systems:

The food waste detection system needs to be integrated with existing systems and processes to ensure its seamless operation. This requires technical knowledge and expertise to ensure that the system can communicate effectively with other systems and applications.

7.4.5 Ethical and privacy considerations:

The use of deep learning techniques for food waste detection raises ethical and privacy concerns, such as the potential misuse of data and the biases inherent in the data. Organizations need to ensure that the system is developed and used in an ethical and responsible manner.

In conclusion, implementing a food waste detection system using deep learning techniques requires a range of technical and non-technical considerations. Addressing these issues will require careful planning, investment in hardware and software, technical expertise, and a commitment to ethical and responsible use of the system

REFERENCES

- [1]. Azimi I, Anzanpour A, Rahmani AM, Pahikkala T, Levorato M, Liljeberg P, Dutt N (2107) HiCH: Hierarchical fog-assisted computing architecture for healthcare IoT. *ACM Trans Embed Comput Syst* 16(5s):174
- [2]. Abeshu A, Chilamkurti N (2018) Deep learning: the frontier for distributed attack detection in fog-to-things computing. *IEEE Commun Mag* 56(2):169–175 60.
- [3]. Conference on engineering, technology and innovation. IEEE, Funchal, pp 1066–1070 58.
- [4]. Grassi G, Sammarco M, Bahl P, Jamieson K, Pau G (2015) Poster: Parkmaster: Leveraging edge computing in visual analytics. In: *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, ser. *MobiCom '15*. ACM, New York, pp 257–259. <https://doi.org/10.1145/2789168.2795174> (Online)
- [5]. Hogan M, Esposito F (2017) Stochastic delay forecasts for edge traffic engineering via bayesian networks. In: *IEEE international symposium on network computing and applications*. IEEE, Cambridge, MA, pp 1–4 62.
- [6]. Kim H, Feamster N (2013) Improving network management with software defined networking. *Commun Mag IEEE* 51(2):114–119 63.
- [7]. Kim HJ, Jung MY, Chin WS, Jang JW (2017) Identifying service contexts for qos support in iot service-oriented software defined networks.
- [8]. Wang S, Zhao Y, Huang L, Xu J, Hsu CH (2017) Qos prediction for service recommendations in mobile edge computing. *J Parallel Distrib Comput*. <https://doi.org/10.1016/j.jpdc.2017.09.014> 57.
- [9]. Schneible J, Lu A (2017) Anomaly detection on the edge. In: *MILCOM*

2017—2017 IEEE Military Communications Conference (MILCOM), pp 678–682 59.

- [10]. Yang M, Zhu T, Liu B, Xiang Y, Zhou W (2018) Machine learning differential privacy with multifunctional aggregation in a fog computing architecture. *IEEE Access* 6:17119–17129. <https://doi.org/10.1109/ACCESS.2018.2817523> 61.
- [11]. Hogan M, Esposito F (2017) Stochastic delay forecasts for edge traffic engineering via bayesian networks. In: *IEEE international symposium on network computing and applications*. IEEE, Cambridge, MA, pp 1–4 62.

APPENDIX

A. SOURCE CODE:

```
{
  "cells": [
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "7E4qgkp_wGaE"
      },
      "outputs": [],
      "source": [
        "from keras.utils import np_utils\n",
        "from keras.models import Sequential\n",
        "from keras.layers import Convolution2D,Dense,MaxPool2D,Activation,Dropout,Flatten\n",
        "from keras.layers import GlobalAveragePooling2D\n",
        "from keras.optimizers import Adam\n",
        "from sklearn.model_selection import train_test_split\n",
        "from keras.layers import BatchNormalization\n",
        "import os\n",
        "import pandas as pd\n",
        "import plotly.graph_objs as go\n",
        "import matplotlib.ticker as ticker\n",
        "import matplotlib.pyplot as plt\n",
        "import seaborn as sns\n",
        "from tensorflow.keras.models import load_model\n",
        "import plotly.express as px\n",
        "import cv2\n",
        "import numpy as np\n",
        "from sklearn.model_selection import train_test_split\n",
        "from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D,
```

BatchNormalization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D, GlobalMaxPooling2D"

```
]
},
{
  "cell_type": "code",
  "source": [],
  "metadata": {
    "id": "4k0zQc59PKtp"
  },
  "execution_count": null,
  "outputs": []
},
{
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "MeTMtosYpVKS",
    "outputId": "2d4aae4a-cfbc-4868-bafe-bf5b5ca48f23"
  },
  "outputs": [
    {
      "output_type": "stream",
      "name": "stdout",
      "text": [
        "Looking in indexes: https://pypi.org/simple, https://us-  
python.pkg.dev/colab-wheels/public/simple/\n",
        "Collecting opendatasets\n",
        "  Downloading opendatasets-0.1.22-py3-none-any.whl (15 kB)\n",
        "Requirement already satisfied: kaggle in /usr/local/lib/python3.8/dist-
```

packages (from opendatasets) (1.5.12)\n",

"Requirement already satisfied: click in /usr/local/lib/python3.8/dist-packages (from opendatasets) (7.1.2)\n",

"Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (from opendatasets) (4.64.1)\n",

"Requirement already satisfied: urllib3 in /usr/local/lib/python3.8/dist-packages (from kaggle->opendatasets) (1.24.3)\n",

"Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from kaggle->opendatasets) (2.25.1)\n",

"Requirement already satisfied: python-slugify in /usr/local/lib/python3.8/dist-packages (from kaggle->opendatasets) (8.0.0)\n",

"Requirement already satisfied: certifi in /usr/local/lib/python3.8/dist-packages (from kaggle->opendatasets) (2022.12.7)\n",

"Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.8/dist-packages (from kaggle->opendatasets) (1.15.0)\n",

"Requirement already satisfied: python-dateutil in /usr/local/lib/python3.8/dist-packages (from kaggle->opendatasets) (2.8.2)\n",

"Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.8/dist-packages (from python-slugify->kaggle->opendatasets) (1.3)\n",

"Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests->kaggle->opendatasets) (4.0.0)\n",

"Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests->kaggle->opendatasets) (2.10)\n",

"Installing collected packages: opendatasets\n",

"Successfully installed opendatasets-0.1.22\n",

"Please provide your Kaggle credentials to download this dataset. Learn more: <http://bit.ly/kaggle-creds>\n",

"Your Kaggle username: gaurav065\n",

"Your Kaggle Key: \n",

"Downloading fruits-fresh-and-rotten-for-classification.zip to ./fruits-fresh-

```

and-rotten-for-classification\n"
    ]
  },
  {
    "output_type": "stream",
    "name": "stderr",
    "text": [
      "100%|██████████| 3.58G/3.58G [00:27<00:00, 139MB/s]\n"
    ]
  },
  {
    "output_type": "stream",
    "name": "stdout",
    "text": [
      "\n"
    ]
  }
],
"source": [
  "!pip install opendatasets\n",
  "import opendatasets as od\n",
  "od.download('https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification')\n"
]
},
{
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
    "id": "0nuwrNtguWJr"
  },
  "outputs": [],
  "source": [

```

```

    "train ='/content/fruits-fresh-and-rotten-for-classification/dataset/test'\n",
    "validation_dir          =          '/content/fruits-fresh-and-rotten-for-
classification/dataset/train'"
]
},
{
    "cell_type": "code",
    "execution_count": null,
    "metadata": {
        "id": "boFpeNSVwTX2"
    },
    "outputs": [],
    "source": [
        "import glob\n",
        "def get_files(directory):\n",
        "    if not os.path.exists(directory):\n",
        "        return 0\n",
        "        count=0\n",
        "        for current_path,dirs,files in os.walk(directory):\n",
        "            for dr in dirs:\n",
        "                count+= len(glob.glob(os.path.join(current_path,dr+\"/*\")))\n",
        "        return count"
    ]
},
{
    "cell_type": "code",
    "execution_count": null,
    "metadata": {
        "colab": {
            "base_uri": "https://localhost:8080/"
        },
        "id": "5ACoTgXkv4dt",
        "outputId": "41d17cff-c6d6-46a9-a7d3-f35cb34a9c3c"
    }
}

```

```

},
"outputs": [
  {
    "output_type": "stream",
    "name": "stdout",
    "text": [
      "6 Classes\n",
      "2698 Train images\n",
      "10901 Test images\n"
    ]
  }
],
"source": [
  "train_samples =get_files(train)\n",
  "num_classes=len(glob.glob(train+\"/*\"))\n",
  "test_samples=get_files(validation_dir)\n",
  "print(num_classes,\"Classes\")\n",
  "print(train_samples,\"Train images\")\n",
  "print(test_samples,\"Test images\")"
]
},
{
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "sfiUXK6evyTD",
    "outputId": "eb068469-67f5-4d76-aeb1-646c332276d3"
  },
  "outputs": [
    {

```

```

    "output_type": "stream",
    "name": "stdout",
    "text": [
        "Found 2698 images belonging to 6 classes.\n",
        "Found 10901 images belonging to 6 classes.\n"
    ]
}
],
"source": [
    "from keras.preprocessing.image import ImageDataGenerator\n",
    "train_datagen=ImageDataGenerator(rescale=1./255,\n",
    "                                shear_range=0.2,\n",
    "                                zoom_range=0.2,\n",
    "                                horizontal_flip=True)\n",
    "test_datagen=ImageDataGenerator(rescale=1./255)\n",
    "img_width,img_height =224,224\n",
    "input_shape=(img_width,img_height,3)\n",
    "batch_size =32\n",
    "train_generator =train_datagen.flow_from_directory(train,\n",
    "
target_size=(img_width,img_height),batch_size=batch_size)\n",

"test_generator=test_datagen.flow_from_directory(validation_dir,shuffle=True,ta
rget_size=(img_width,img_height),batch_size=batch_size)"
]
},
{
    "cell_type": "code",
    "source": [
        "train_datagen\n",
    ],
    "metadata": {
        "colab": {

```

```

    "base_uri": "https://localhost:8080/"
  },
  "id": "jWfm_A0wWaM3",
  "outputId": "e874e693-3d38-4cbf-950c-2194a52c2f92"
},
"execution_count": null,
"outputs": [
  {
    "output_type": "execute_result",
    "data": {
      "text/plain": [
        "<keras.preprocessing.image.ImageDataGenerator"
        0x7f6389919160>"
      ]
    },
    "metadata": {},
    "execution_count": 26
  }
],
},
{
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
    "id": "5X8TQ6X3uBsY",
    "colab": {
      "base_uri": "https://localhost:8080/"
    }
  },
  "outputId": "53dcfd01-5121-4849-e192-c07ddd44fcd4"
},
"outputs": [
  {
    "output_type": "stream",

```

```

    "name": "stdout",
    "text": [
        "Please provide your Kaggle credentials to download this dataset. Learn
more: http://bit.ly/kaggle-creds\n",
        "Your Kaggle username: gaurav065\n",
        "Your Kaggle Key: ..... \n",
        "Downloading inceptionresnetv2.zip to ./inceptionresnetv2\n"
    ]
},
{
    "output_type": "stream",
    "name": "stderr",
    "text": [
        "100%|██████████| 392M/392M [00:07<00:00, 56.5MB/s]\n"
    ]
},
{
    "output_type": "stream",
    "name": "stdout",
    "text": [
        "\n"
    ]
}
],
"source": [
    "od.download('https://www.kaggle.com/datasets/keras/inceptionresnetv2')\n"
]
},
{
    "cell_type": "code",
    "source": [
        "from keras import Model\n",
        "from keras import optimizers\n",

```

```

"import tensorflow as tf\n",
"from keras.optimizers import Adam\n",
"from keras.models import Sequential\n",
"from keras.layers import GlobalAveragePooling2D,Dense\n",
"\n",
"model_dir                                                    =
\"../content/inceptionresnetv2/inception_resnet_v2_weights_tf_dim_ordering_tf_
kernels_notop.h5\"\n",
"model = Sequential()\n",
"model.add( tf.keras.applications.InceptionResNetV2(\n",
"  include_top=False,\n",
"  weights= model_dir, input_shape=(224, 244, 3)))\n",
"\n",
"model.add(GlobalAveragePooling2D())\n",
"model.add(Dense(6, activation=\"softmax\"))\n",
"model.summary()"
],
"metadata": {
  "colab": {
    "base_uri": "https://localhost:8080/"
  },
  "id": "TI7b4QakQCBb",
  "outputId": "8d5650b8-c0c6-4b85-8db2-a92a2b397e19"
},
"execution_count": null,
"outputs": [
  {
    "output_type": "stream",
    "name": "stdout",
    "text": [
      "Model: \"sequential_2\"\n",

```

```

____\n",
    " Layer (type)          Output Shape          Param #   \n",

"=====
=====\\n",
    " inception_resnet_v2 (Functi (None, 5, 6, 1536)    54336736 \n",
    " onal)                  \n",
    "                        \n",
    " global_average_pooling2d_1 (None, 1536)          0 \n",
    " (GlobalAveragePooling2D) \n",
    "                        \n",
    " dense_3 (Dense)         (None, 6)           9222 \n",
    "                        \n",

"=====
=====\\n",
    "Total params: 54,345,958\\n",
    "Trainable params: 54,285,414\\n",
    "Non-trainable params: 60,544\\n",

"
____\n"
    ]
    }
    ]
    },
    {
        "cell_type": "code",
        "source": [
            "model.compile(optimizer='adam',          loss='categorical_crossentropy',
metrics=['accuracy'])"
        ],
        "metadata": {

```

```

    "id": "vPyISJuUQE1h"
  },
  "execution_count": null,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "from tensorflow.keras.callbacks import EarlyStopping\n",
    "early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience =
10, mode = 'min', restore_best_weights = True)\n",
    "history                                     =
model.fit(train_generator,validation_data=test_generator,epochs=2,shuffle=True,
callbacks=early_stopping_callback)"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "vQ2EsQW0QINH",
    "outputId": "0626c5ac-d781-4c2b-96c5-db26a4e6c66e"
  },
  "execution_count": null,
  "outputs": [
    {
      "output_type": "stream",
      "name": "stdout",
      "text": [
        "Epoch 1/2\n",
        "85/85 [=====] - 117s 1s/step - loss:
0.0620 - accuracy: 0.9785 - val_loss: 0.3946 - val_accuracy: 0.8933\n",
        "Epoch 2/2\n",
        "85/85 [=====] - 117s 1s/step - loss:

```

0.0828 - accuracy: 0.9733 - val_loss: 2.3252 - val_accuracy: 0.7353\n"

```
    ]
  }
]
},
{
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
    "id": "T1ZQ5_HzFxnK"
  },
  "outputs": [],
  "source": [
    "model.save('inceptionResnetv2_model.h5')"
  ]
},
{
  "cell_type": "code",
  "source": [
    "model.evaluate(test_generator, verbose=1)"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "QZJxIDPiUZ-O",
    "outputId": "18f8c47e-b6c2-41b9-e5e9-d8f8ad92d788"
  },
  "execution_count": null,
  "outputs": [
    {
      "output_type": "stream",
      "name": "stdout",
```

```

      "text": [
        "341/341 [=====] - 65s 190ms/step -
loss: 2.3252 - accuracy: 0.7353\n"
      ]
    },
    {
      "output_type": "execute_result",
      "data": {
        "text/plain": [
          "[2.3252334594726562, 0.735253632068634]"
        ]
      },
      "metadata": {},
      "execution_count": 20
    }
  ]
},
{
  "cell_type": "code",
  "source": [
    "from google.colab import drive\n",
    "drive.mount('/content/drive')"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "s7EVZzy3bUz9",
    "outputId": "546fa086-2fcd-431e-be9b-9d9ac507cd21"
  },
  "execution_count": null,
  "outputs": [
    {

```

```

    "output_type": "stream",
    "name": "stdout",
    "text": [
        "Mounted at /content/drive\n"
    ]
}
],
{
    "cell_type": "code",
    "source": [
        "model = load_model('/content/drive/MyDrive/Fresh
rotten/CNN_model.h5')"
    ],
    "metadata": {
        "id": "pTbwNnmrbxml"
    },
    "execution_count": null,
    "outputs": []
},
{
    "cell_type": "code",
    "source": [
        "model.evaluate(test_generator, verbose=1)"
    ],
    "metadata": {
        "colab": {
            "base_uri": "https://localhost:8080/"
        },
        "id": "Lhj0TnxlbxDI",
        "outputId": "6ec50d88-97ce-47ce-ed5e-c731a90c5982"
    },
    "execution_count": null,

```

```

"outputs": [
  {
    "output_type": "stream",
    "name": "stdout",
    "text": [
      "341/341 [=====] - 51s 148ms/step -
loss: 0.3341 - accuracy: 0.8789\n"
    ]
  },
  {
    "output_type": "execute_result",
    "data": {
      "text/plain": [
        "[0.3340620696544647, 0.8789101839065552]"
      ]
    },
    "metadata": {},
    "execution_count": 23
  }
]
},
"metadata": {
  "accelerator": "GPU",
  "colab": {
    "provenance": []
  },
  "gpuClass": "standard",
  "kernelSpec": {
    "display_name": "Python 3",
    "name": "python3"
  },
  "language_info": {

```

```

        "name": "python"
    }
},
    "nbformat": 4,
    "nbformat_minor": 0
}

```

UI CODE:

```

import streamlit as st
import tensorflow as tf
import cv2
import numpy as np
from PIL import Image

model_to_predict = tf.keras.models.load_model('CNN_model.h5')
def predict_covid(test_image):
    img = cv2.imread(test_image)
    img = img / 255.0
    img = cv2.resize(img, (224, 224))
    img = img.reshape(1,224,224,3)
    prediction = model_to_predict.predict(img)
    pred_class = np.argmax(prediction, axis = -1)
    return pred_class

def load_image(image_file):
    img = Image.open(image_file)
    return img

st.write("Food freshness using CNN")

pic = st.file_uploader("Upload a picture!")

```

```

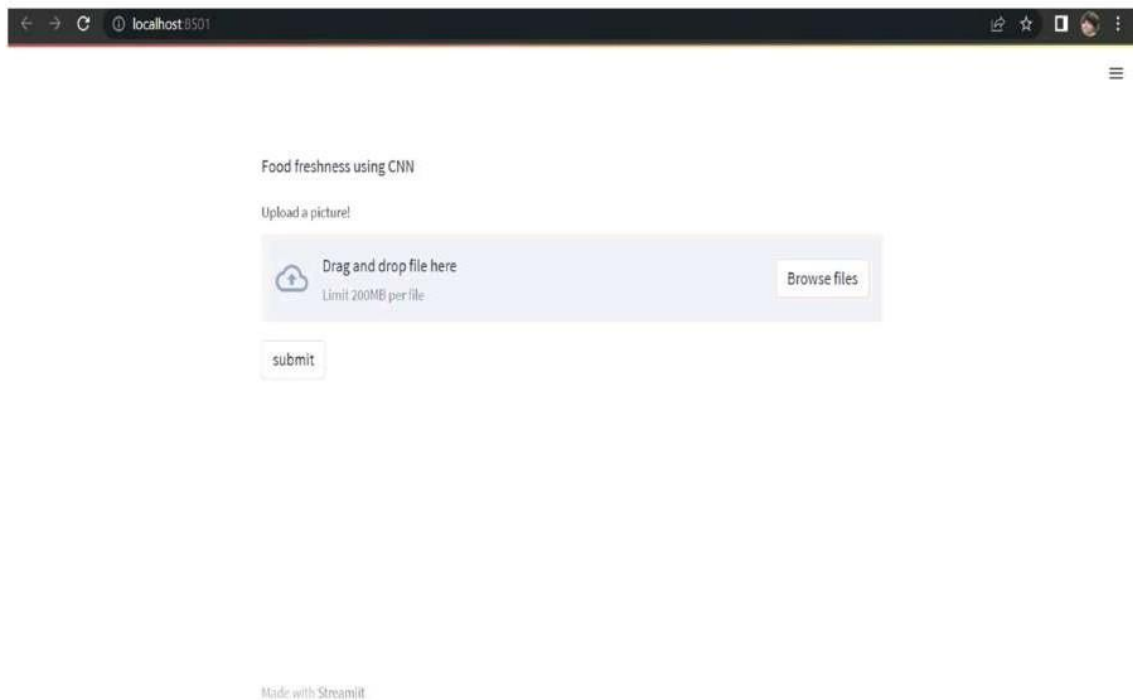
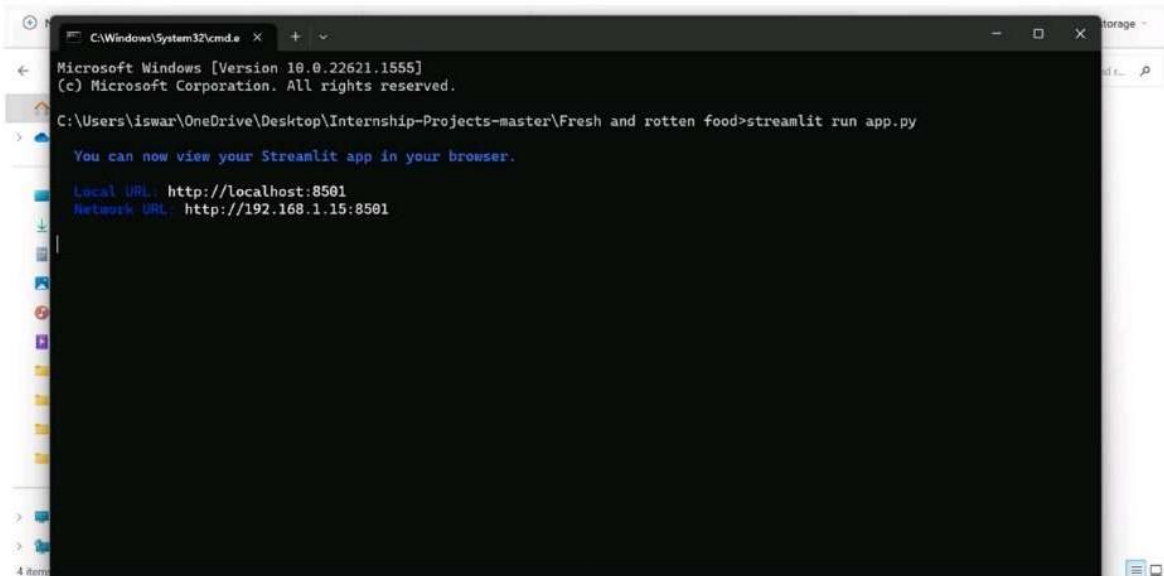
submit = st.button('submit')
if submit:
    pic_details = {"filename":pic.name, 'filetype':pic.type, 'filesize':pic.size}
    st.write(pic_details)

    st.image(load_image(pic), width=250)

    with open('test.jpg', 'wb') as f:
        f.write(pic.getbuffer())
    pred = predict_covid('test.jpg')
    if pred[0] == 0:
        st.write('Fresh')
    elif pred[0] == 1:
        st.write('Fresh')
    elif pred[0] == 2:
        st.write('Fresh')
    elif pred[0] == 3:
        st.write('Rotten')
    elif pred[0] == 4:
        st.write('Rotten')
    elif pred[0] == 5:
        st.write('Rotten')

```

B.SCREENSHOTS



Freshness of Food Detection using Machine Learning for IoT

Ch. Iswarya
*Electronics and Communication
Engineering*
Sathyabama institute of science and
Technology
iswaryach41@gmail.com

P. Prathyusha Reddy
*Electronics and Communication
Engineering*
Sathyabama institute of science and
Technology
prathyushar85@gmail.com

Dr. V. Vijayakumar, M.Tech, Ph.D
*Electronics and Communication
Engineering*
Sathyabama institute of science and
Technology
vijayakumar.ece@sathyabama.ac.in

Abstract—Food spoilage is a serious problem in the modern world because eating spoiled food is not healthy for customers. Using appropriate sensors, our research aims to monitor gases emitted from specific food products and identify rotting foods. When the microcontroller notices this, it sends an alert via the IoT so that appropriate action can be taken. It is commonly used in the food industry where food is currently inspected manually. In order to predict the likelihood and timing of food spoilage from a particular supplier, we want to incorporate machine learning into our model. Everything will eventually get better as retailers compete to offer healthier foods and fresh vegetables. Food freshness is an important factor in determining food quality and safety. The ability to accurately track food freshness helps reduce food waste and ensures that consumers have access to safe, high-quality products. Deep learning techniques are very promising in this field because they can be used to analyze a wide variety of data such as images, sensor data, and chemical analysis. One approach uses computer vision to detect freshness by analyzing images of food. Deep learning algorithms can be trained on labelled image datasets to recognize patterns and features that represent novelty. These algorithms can be used to classify images as fresh or damaged, predict product shelf life, and more. Another method is to use sensor data to detect freshness. Foods typically release gas when their temperature, moisture, or other physical properties are compromised or changed. Deep learning algorithms can be trained to analyze sensor data and detect these changes to accurately predict freshness. Chemical analysis is also used for novelty detection, where deep learning algorithms can be trained to identify patterns and features of chemical data indicating novelty. This can be done by analyzing data from various sensors such as gas sensors, spectrophotometers, and mass spectrometers. Food freshness is tracked, making it more accurate, efficient, and cost-effective. Using deep learning to sense freshness can reduce food waste, improve food safety and quality, and ultimately create a more sustainable food system.

Keywords— *Internet of things, machine learning and food safety.*

I. INTRODUCTION

In the 21st century, most of the GDP comes from food trade. One of the biggest problems is food spoilage, especially when meat products, fruits and vegetables go bad. In the fresh produce industry, someone sits on a conveyor belt and checks the quality of goods manually. With this in mind, using an

automated process increases the accuracy of finding spoiled food and reduces the amount of manual work. I want to automate this process using a set of smart sensors and microcontrollers like nodes. In addition to sending this data to the cloud as part of an IoT application, you can program an alarm to notify you when food is bad or expired. By letting the relevant authorities know that they have received the wrong food and be more transparent.

Determining food freshness is one of the most important aspects of ensuring food safety and quality for consumers. Traditional freshness determination methods, such as visual inspection and sensory testing, are time-consuming and can result in subjective and inconsistent freshness ratings. The increasing demand for fresh and high quality food has led to the development of new methods to use new learning, including deep learning.

Deep learning is a branch of machine learning that involves using artificial neural systems that can be trained to analyze large amounts of data and identify patterns and features that represent novelty. Deep learning algorithms are trained on a variety of data, including images, sensor data, and chemical analysis, making them a versatile tool for novel discovery. One approach to using deep learning to detect news is to use computer vision. This involves analyzing food images to identify patterns and characteristics that indicate freshness. This includes analyzing color, texture, and other visual characteristics. Deep learning algorithms are trained on iconic image databases to classify new or old images, predict product shelf life, and more. Another method is to use sensor data to detect novelty. Food often releases moisture or other physical properties due to gas decomposition or temperature changes. Deep learning algorithms can be trained to analyze sensor data, detect these changes, and accurately predict innovations. This approach can be applied to a variety of foods, including fruits, vegetables, and meat. Chemical analysis is also used to determine freshness. Deep learning algorithms can be trained to identify patterns and features in chemical data that represent novelty. This can be done by

analyzing data from various sensors such as gas sensors, spectrophotometers and mass spectrometers. These sensors detect and detect volatile organic compounds (VOCs) emitted as food spoilage.

II. LITERATURE SURVEY

Anomaly detection on the edge, Schneible J, Lu A (2017).

59, pp. 678–682 in: MILCOM 2017—2017 IEEE Military Communications Conference.

An artificial neural network based solution for anomaly detection in edge computing scenarios was also presented by Schnibel et al. In the normal nervous system,

Training models and all sensor data should be stored in a central cloud. This feature helps avoid traffic jams and delays on travel routes. The new system is based on federated training, where a copy of the training model is stored on each edge device and the training data is distributed to the edge devices. As a result, the training results from the edge devices only need to be collected by the central cloud storage. Federated learning techniques increase bandwidth, reduce latency, and use all the processing power in the edge set.

In mobile edge computing, Wang S, Zhao Y, Huang L, Xu J, and Hsu CH (2017). predicted QoS for service suggestions. Journal of Parallel and Distributed Computing. 10.1016/j.jpdc.2017.09.014 57.

The problem of detecting distributed attacks, which are less visible than non-distributed attacks, was investigated in more detail by Abshu et al. Common machine learning algorithms suffer from poor accuracy and scalability due to the variety and complexity of devices used in edge computing environments. The authors then proposed a new deep learning approach. Recently, it has received much attention due to the development of GPU technology and deep learning theory. Finally, to evaluate the proposed method, the authors used simulations built using publicly available datasets. The results show that the DL-based method performs better than the conventional method. In addition to security concerns. Privacy is another important factor in the fog computing environment.

III. METHODOLOGY

The two consecutive components of the proposed solution are the classification convolution network and the pre-processing pipeline. Photos used in subsequent classification steps are prepared through a pre-processing pipeline. The pipeline can be scaled, remove parts of interest and remove the background. These types of techniques are often used in image processing applications. On the one hand, we deliberately built a deep neural network to make the most of the available training data. We have incorporated an advanced neural network layer, the delta layer, into our architecture. The goal is to highlight the properties that become apparent after

removing all food waste. Focus on location-specific elements of the apple salad combination image that were absent from the salad image alone, for example, placing the apple on top of the salad. A delta layer in a particular abstraction layer calculates the $V_{after} - cV_{before}$ after and Preimage V_{before} feature volume values, where the ReLU activation function is represented by a learnable scaling constant.

EXISTING SYSTEM

Odor monitoring is applied in smart homes to detect spoiled food. Ligation detection reactions coupled with flow hybridization on membranes are now widely used to identify food spoilage and hazards. Microbes Using an electronic nose, we compared the rate of degradation of Filipino foods with tomatoes over time and temperature. Food spoilage in smart homes was discovered based on neural and fuzzy processing of odor sensor data. In addition to the previously described techniques, a new approach based on metal-oxide-semiconductor sensors is proposed to classify the spoilage level of red meat. A CMOS fish tolerance to period converter with a calibration circuit has been developed for fish spoilage detection applications. The recent development of multiple rapid bioanalytical methods for food pathogenic bacteria has facilitated the development of multispectral imaging of spoilage.

PROPOSED SYSTEM

Food recognition studies require a large number of images of typical meals. Food graphics usually represent different things. To evaluate food recognition in our dataset, we need to identify and separate individual food components in images. Food logging apps for mobile phones can generate excellent data for this purpose. We used Food-Log (FL) data. Using text and images, ordinary people can use FL to list their food products. The user takes a photo of the food and enters the name of the food on the smartphone's touch screen to identify each part containing the food. Common food databases are often used to select food names. CNN meta-parameters include the number of layers, training, pre-processing and initialization. This section explains how to calibrate the number of layers, the number and size of kernels in each layer, and other parameters. After merging the layers according to the Kryzhevsky CNN, we normalize the data using local response normalization (LRN). Food detection classifies the entire image as food, which may contain both food and background. Or not. Program image below Image noise removal is done by applying a pre-processing method on the target image. Then the CNN algorithm is used. CNN algorithm has three layers. These three layers are RELU, convolutional layer and FC layer. Convolution tools are used in the feature extraction process to identify and isolate the characteristic features of an image for analysis. A feature extraction network has many pairs of convolutional or combinational layers. The output of the convolution method

is used by this fully connected layer to classify the image using the previous features.

This CNN feature extraction model tries to extract the minimum number of features from a practical dataset. Develop additional features by merging features from the original feature set into a new feature. The first layer used to extract different features from the input image is the convolution layer. Convolution is a mathematical process between the input image and an $M \times M$ filter on this layer. The main goal of the convolution layer is to minimize the size of the convolutional feature maps to minimize the computational cost. This is done by remapping each feature individually and creating links between layers less often. Depending on the approach you choose, there are several integration processes. This is basically a list of attributes produced by a convolution layer. Neurons between two layers are connected by fully connected (FC) layers containing weights and biases. These layers, which make up some recent CNN architectures, are often placed before the output layer. After training the dataset, food spoilage detection is done and the results are displayed.

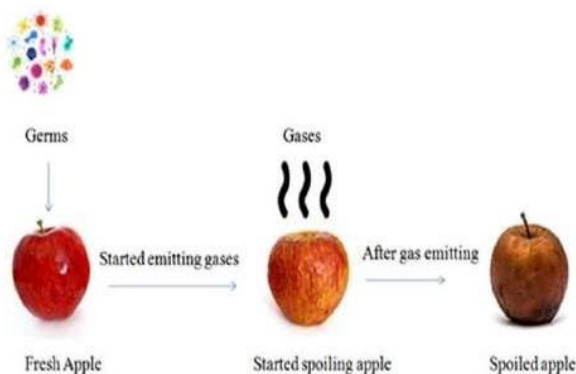


Fig. 1. Example of a figure caption.

DATA COLLECTION AND PREPROCESSING

• COLLECTING DATA

Data mining is a technique used to collect data from many sources to create deep learning. Data must be stored in a way that is appropriate for the task.

• DATA PREPROCESSING

Data preparation is the process of converting raw data into something usable and consumable. Data preparation changes the form of data so that it can be processed faster and more effectively by data mining, machine learning, and other data science applications.



MODEL CREATION AND TRAINING

A subset of deep neural networks called convolutional neural networks are often used in deep learning to study vision. Common weight architecture and This is one of the properties of translation invariance, which is also known as invariant or space-invariant artificial neural networks. They are used by programs that recognize images and videos.

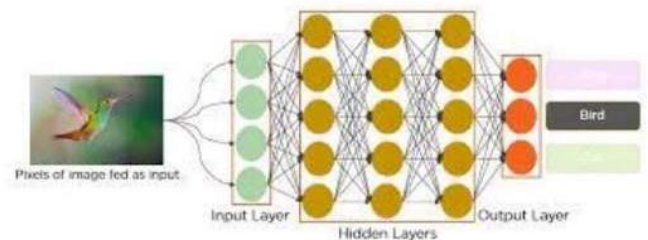


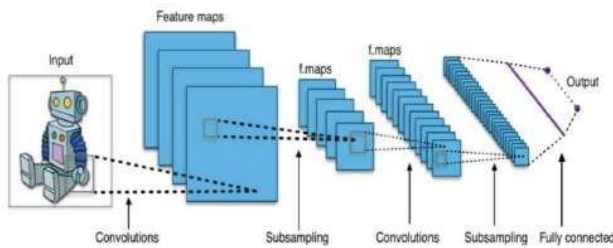
Fig. 2. Example of a figure caption.

NETWORK INTRODUCTION

A neural network is a mathematical construct that basically tries to simulate the human brain. Convolutional neural network is one of the different types of neural networks (CNN). CNNs focus on image processing and are useful for classification, segmentation and object recognition in images.

COVOLUTION NEURAL NETWORKS

Convolutional neural networks, commonly called convnet or CNN, are components of machine learning. A convolutional neural network may contain tens or hundreds of layers, and each layer may be trained to recognize different aspects of an image. CNNs are very useful because they reduce the amount of human work by automatically identifying features. To extract features from an image, multiply the two matrices representing the two images to produce an output.



MERGE LAYERS AND FULLY CONNECTED LAYERS

Composite layer

Merging layers reduces the size of feature maps. Therefore, fewer parameters need to be learned and less network processing is required. A convolutional neural network (CNN) architecture usually includes convolution as a feature.

Fully connected layers

Use the output of the convolution layer to drive the nonlinear function. During each training cycle, this output must be transformed into a column vector for multiple perception and fed to a feedforward neural network using back-propagation. This helps the model to extract the important components of the image.

Prediction of food spoilage

In this module, the final output of the project is considered. The pre-processed images are fed into the CNN model for classification and prediction. As a result, an ideal CNN model predicts whether food is fresh or stale depending on many features. A project to identify spoiled food using computer vision and artificial intelligence techniques without a spoiled food sensor was successfully built on a client's notebook platform in Python with reliable results.



IV. CONCLUSION

The most important IoT technology may end up being machine learning (IoT). Machine learning trends that provide insights for IoT applications. Despite the recent wave of machine learning success in networking, the literature on the

application of machine learning to Internet of Things (IoT) services and systems is scarce. This is the gap that our research seeks to fill. Due to the adaptable and dynamic nature of the Internet of Things (IoT), it is difficult to cover all potential applications. However, the purpose of this article was to cover the basic methodology and the most important applications of machine learning for the Internet of Things (IoT).

These applications include traffic profiling, IoT device identification, security, edge computing architectures, SDN-based network management, and general IoT applications. The latest research forms the basis of a comprehensive analysis. In the pre-processing step, we performed background subtraction using U-Net (Section III-A). Binary segmentation masks were provided for a total of 1000 images. Mask allowed.

Identification of food waste pixels in images. We divided the dataset into training, validation, and testing sets, each containing 700, 200, and 100 images. The accuracy of the network at the pixel level U was trained to 95.8%. We found that U-Net was successfully trained using a small amount of data. This is consistent with the literature. We developed a deep neural network for food classification. The classification dataset contained at least 1 million images. Each image was only associated with the corresponding food classification label. According to this label, the last thing thrown in the trash can be classified as food waste. Although there were several food waste classes in the images, localization and segmentation data were not available. A trained deep network classifies food waste images into 20 different waste classes with a classification accuracy of 83.5%.

REFERENCES

- [1] Azimi I, Anzanpour A, Rahmani M, Pahikkala T, Levorato M, Liljeberg P, Dutt N (2107) HiCH: Hierarchical Fog-Assisted Computing Architecture for Healthcare IoT. *ACM Trans Embed Comput Syst* 16(5s):174
- [2] Grassi G, Sammarco M, Bahl P, Jamieson K, Pau G (2015) Poster: Parkmaster: Leveraging Edge Computing in Visual Analytics. In: *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, ser. *Mobicom'15*. ACM, New York, pp. 257-259. <https://doi.org/10.1145/2789168.2795174> (online)..
- [3] Wang S, Zhao Y, Huang L, Xu J, Hsu CH (2017) Qos prediction for service recommendation in mobile edge computing. *J Parallel Distrib Comput*. <https://doi.org/10.1016/j.jpdc.2017.09.014> 57.
- [4] Engineering, Technology and Innovation Conference. IEEE, Funchal, pp 1066-1070 58.
- [5] Schneible J, Lu A (2017) Anomaly detection in edges. In: *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*, pp. 678-682 59.