Industrial Internship Report on

“Crop and Weed Detection.”

Prepared by

Saloni Anant Butley

Executive Summary
This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT). This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time. This project focuses on developing an AI-based image classification model using Convolutional Neural Networks (CNN) to automatically identify crops and weeds from agricultural field images. The system uses image preprocessing and deep learning in Python (TensorFlow and Keras) to accurately classify images as either crop or weed. The model aims to assist farmers in reducing pesticide use, improving crop yield, and promoting sustainable agriculture This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

TABLE OF CONTENTS

1 Preface

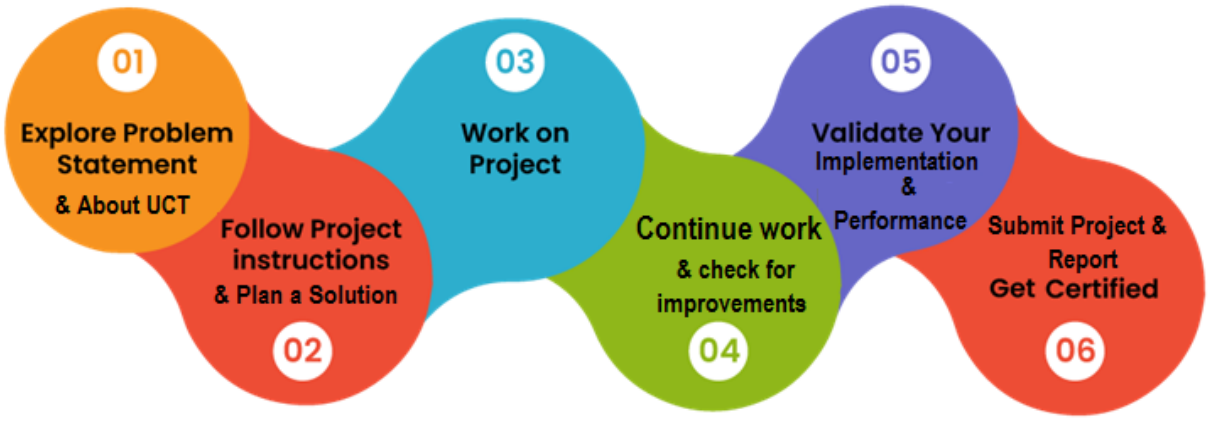
Summary of the whole 6 weeks' work.

About need of relevant Internship in career development.

Brief about Your project/problem statement.

Opportunity given by USC/UCT.

How Program was planned



Your Learnings and overall experience.

Thank to all , who have helped you directly or indirectly.

Your message to your juniors and peers.

2 Introduction

2.1 About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various **Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end** etc.

i. UCT IoT Platform (UCT Insight)

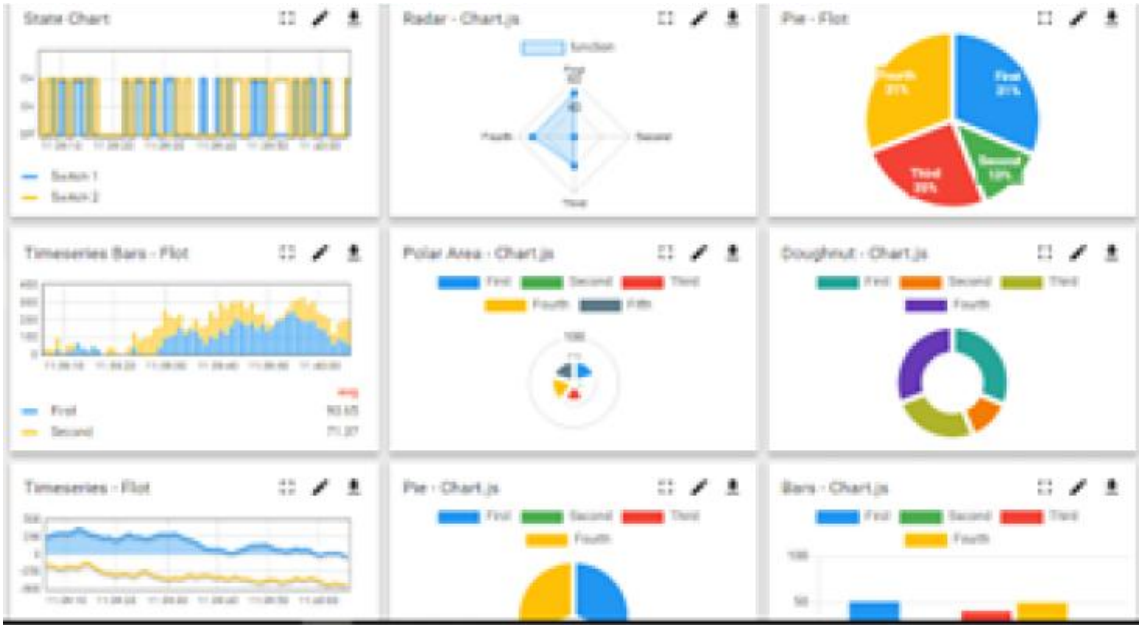
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

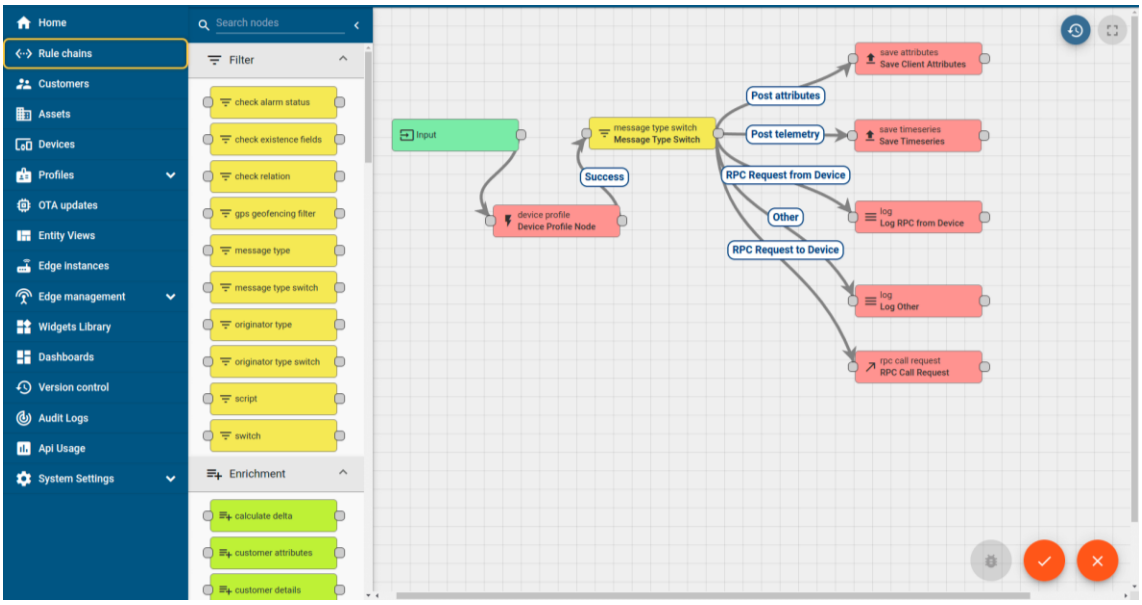
- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA

- It supports both cloud and on-premises deployments.

It has features to

- Build Your own dashboard
- Analytics and Reporting
- Alert and Notification
- Integration with third party application(Power BI, SAP, ERP)
- Rule Engine


State Chart
Radar - Chart.js
Pie - Flot
Timeseries Bars - Flot
Polar Area - Chart.js
Doughnut - Chart.js
Timeseries - Flot
Pie - Chart.js
Bars - Chart.js


Home
Rule chains
Customers
Assets
Devices
Profiles
OTA updates
Entity Views
Edge instances
Edge management
Widgets Library
Dashboards
Version control
Audit Logs
Api Usage
System Settings
Search nodes
Filter
check alarm status
check existence fields
check relation
gps geofencing filter
message type
message type switch
originator type
originator type switch
script
switch
Enrichment
calculate delta
customer attributes
customer details
Input
message type switch
Message Type Switch
Success
device profile
Device Profile Node
Post attributes
Post telemetry
RPC Request from Device
Other
RPC Request to Device
save attributes
Save Client Attributes
save timeseries
Save Timeseries
log
Log RPC from Device
log
Log Other
rpc call request
RPC Call Request

ii. Smart Factory Platform (FACTORY WATCH)

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.

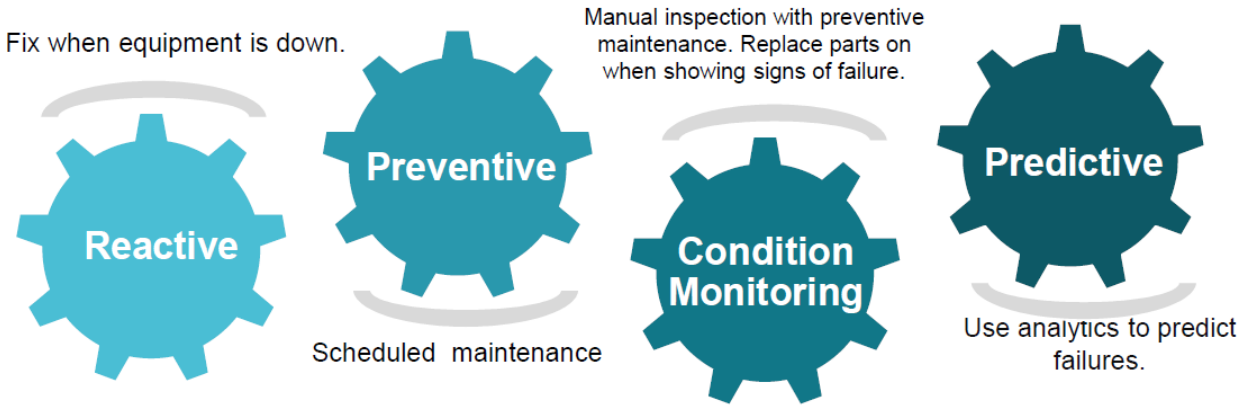
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i

iii. LoRaWAN based Solution

UCT is one of the early adopters of LoRAWAN teschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

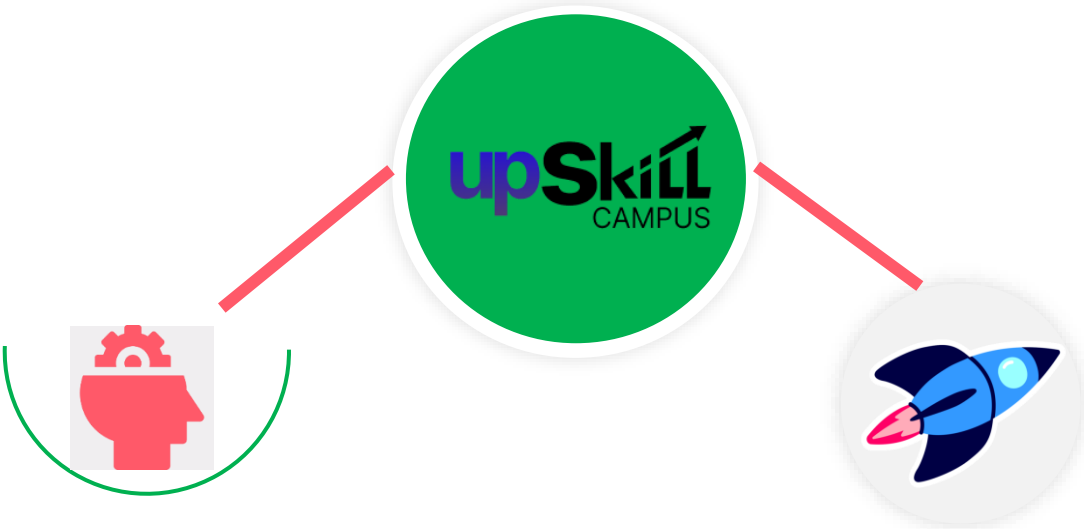
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



2.2 About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

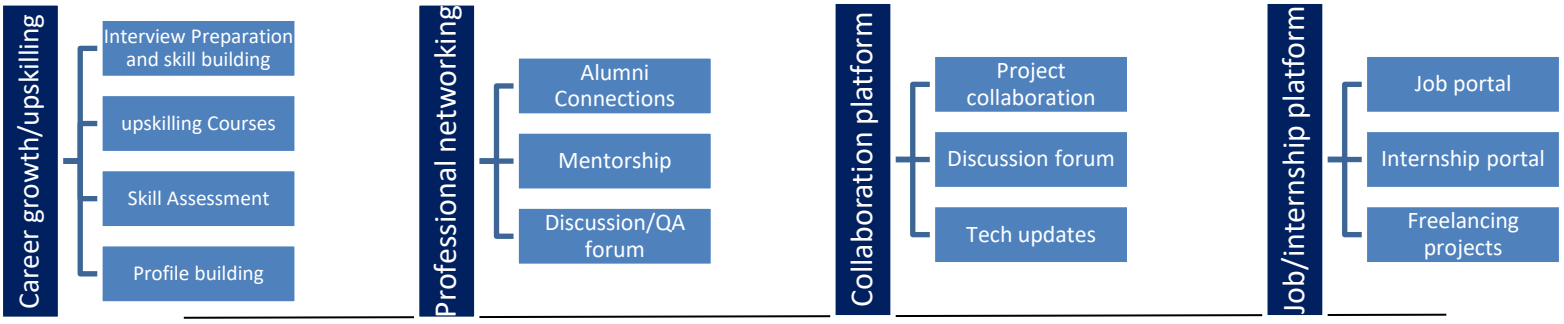
USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

upSkill Campus aiming to upskill 1 million learners in next 5 year

https://www.upskillcampus.com/



2.3 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

2.4 Objectives of this Internship program

The objective for this internship program was to

- ☛ get practical experience of working in the industry.
- ☛ to solve real world problems.
- ☛ to have improved job prospects.
- ☛ to have Improved understanding of our field and its applications.
- ☛ to have Personal growth like better communication and problem solving.

2.5 Reference

[1] https://www.upskillcampus.com/
[2] https://www.uniconvergetech.in/
[3] Research Papers and Articles on AI-based Weed Detection and Smart Farming (IEEE, Springer)

2.6 Glossary

Terms	Acronym
AI	Artificial Intelligence
ML	Machine Learning
CNN	Convolutional Neural Network
IoT	Internet of Things
UCT	UniConverge Technologies Pvt. Ltd

3 Problem Statement

Weed is an unwanted thing in agriculture. Weed use the nutrients, water, land and many more things that might have gone to crops. Which results in less production of the required crop. The farmer often uses pesticides to remove weed which is also effective but some pesticides may stick with crop and may causes problems for humans.

4 Existing and Proposed solution

Existing Solutions

- **Manual Weed Removal:** Traditionally, farmers manually remove weeds using tools or by hand. While effective, it is time-consuming, labor-intensive, and not feasible for large-scale farms.
- **Chemical Pesticides and Herbicides:** The most common approach today is spraying chemicals to kill weeds. Although efficient, these chemicals can contaminate crops, water sources, and soil, posing health and environmental risks.
- **Mechanical Weeders:** Some automated machines use mechanical systems to remove weeds. However, these are often expensive, require constant supervision, and may damage crops if not accurately controlled.
- **AI-Based Drone Spraying:** Some modern farms use drones equipped with AI to detect and spray pesticides selectively, but these systems are costly and may still rely on chemical use.

Limitations of Existing Solutions

- High cost and limited accessibility for small-scale farmers.
- Risk of crop contamination and health hazards from pesticides.
- Lack of precision in distinguishing between crops and weeds.
- Labor-intensive and time-consuming manual methods.
- Environmental harm due to chemical overuse.

Proposed Solution

The proposed solution uses a **Convolutional Neural Network (CNN)** to classify images of crops and weeds. The dataset contains labeled images (.jpeg) and annotation files (.txt). The model is trained in **Python (Google Colab)** using **TensorFlow and Keras**. It learns visual features such as leaf shape, color, and texture to distinguish between crop and weed.

After training, the CNN achieves high accuracy (~90%) in classifying unseen images. The output helps farmers or automated systems make decisions about weed management, reducing dependence on manual work or pesticides.

Value Addition

- **Eco-friendly:** Reduces or eliminates the use of harmful pesticides.
- **Cost-effective:** Affordable for small and medium-scale farmers.
- **Automated:** Minimizes human effort and time.
- **Accurate:** AI ensures precise weed detection without affecting crops.
- **Scalable:** Can be implemented as a ground robot or drone system for various farm sizes.
- **Data-driven:** Collects data on weed growth patterns to help farmers make better crop management decisions.

4.1 Code submission (Github link) :

https://github.com/butleysaloni-gif/upskillcampus_detector_project.git

4.2 Report submission (Github link) :

https://github.com/butleysaloni-gif/upskillcampus_detector_project.git

5 Proposed Design/ Model

The proposed system is an **AI-based image classification model** that detects and differentiates between *crops* and *weeds* using **Convolutional Neural Networks (CNN)**. The design does not involve any mechanical or robotic hardware; instead, it focuses on building a powerful software model that can classify agricultural images automatically.

- 🔹 **System Architecture Overview:**

1. **Input and Data Collection**
   The dataset consists of images of crops and weeds captured in different lighting and soil conditions. Each image has an associated .txt label file specifying the class (crop or weed).

2. **Preprocessing Stage**
   Images are resized (128×128 pixels) and normalized to improve consistency and reduce computational load. Techniques such as filtering, resizing, and contrast adjustment are applied using **OpenCV**.

3. **CNN Model Design**
   The CNN model is built using **TensorFlow and Keras**.
   It consists of:
   - **Convolutional layers (Conv2D)** for feature extraction.
   - **MaxPooling layers** to reduce spatial dimensions.
   - **Flatten layer** to convert features into a 1D vector.
   - **Dense layers** for classification.
   - **Dropout layer** to prevent overfitting.
   - **Sigmoid activation** for binary classification (crop or weed).

4. **Model Training and Validation**
   The dataset is split into training and testing sets (80:20).
   The model is trained over several epochs to minimize binary cross-entropy loss and improve accuracy.

5. **Prediction Stage**
   Once trained, the model can classify new unseen images as “crop” or “weed” with high accuracy.

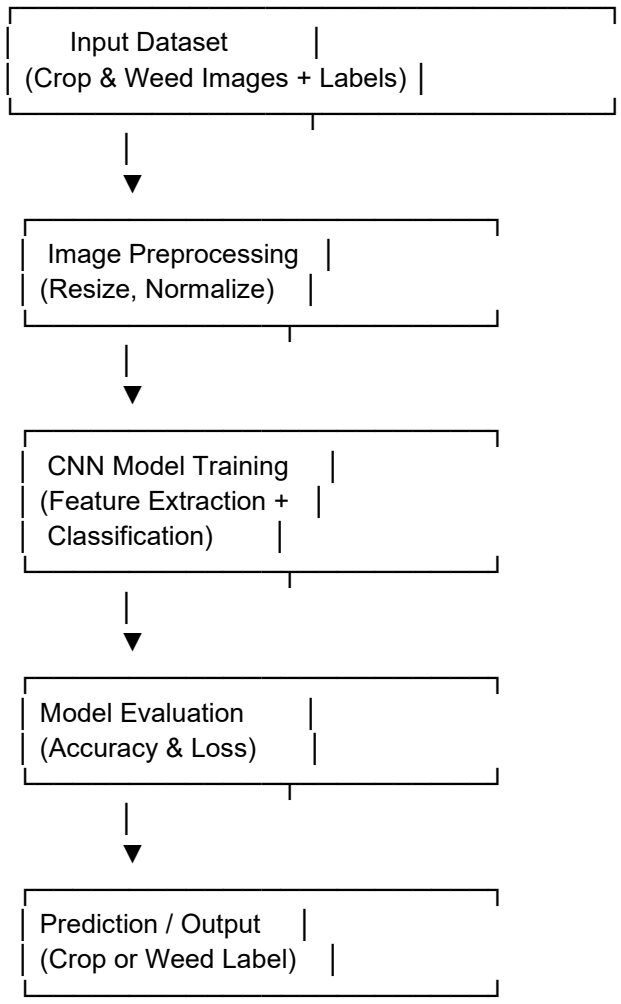
- 🔹 **Advantages of this Design:**
- Purely software-based and lightweight.
- Fast, accurate, and easy to train or deploy.
- Scalable for larger agricultural datasets.

5.1 High-Level Diagram (if applicable)

- Workflow Description

1. Dataset Collection
   The system uses a labeled dataset of agricultural images stored in .jpeg format, each paired with a .txt file indicating the class (crop or weed).
2. Image Preprocessing
   The images are resized (128×128), normalized (scaled between 0–1), and cleaned using OpenCV functions. This ensures uniform input for the CNN model.
3. Model Training (CNN Architecture)
   A Convolutional Neural Network (CNN) is trained using TensorFlow and Keras.
   - The CNN extracts key visual features (leaf shape, texture, color).
   - The network consists of convolutional, pooling, flatten, and dense layers.
   - The output layer uses a sigmoid activation for binary classification.
4. Model Evaluation
   The trained model is validated using unseen test data. The system evaluates accuracy, loss, and inference time to ensure stable performance.
5. Prediction and Output
   For any new image, the model predicts the class label — *Crop (0)* or *Weed (1)* — and outputs the result along with confidence score.

Diagram Representation (Textual)



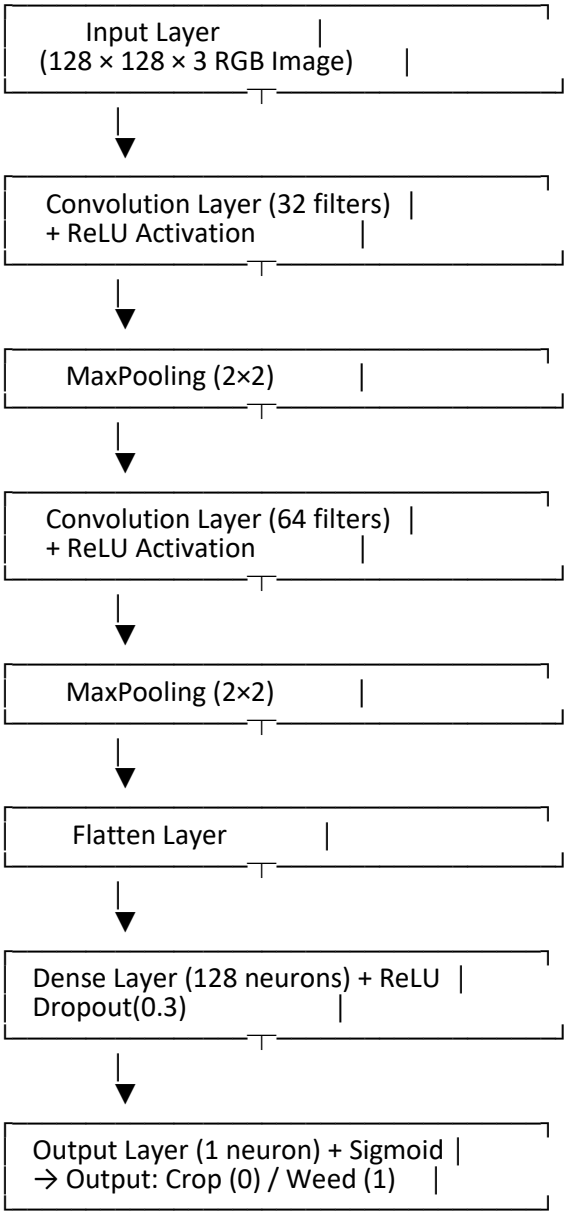
- 🔹 Explanation
- The model's workflow is fully data-driven.
- The CNN learns visual patterns that differentiate crops from weeds.
- Once trained, it can analyze unseen images instantly and classify them with high accuracy (~90%).
- This structure is lightweight, scalable, and environment-independent, making it ideal for smart agriculture and sustainable AI applications.

5.2 Low-Level Diagram (if applicable)

- **Workflow Steps**

1. **Input Layer**
    - Receives input images of size **128 × 128 × 3** (RGB).
    - Images are normalized (pixel values scaled between 0–1).
2. **Convolutional Layers (Feature Extraction)**
    - Apply multiple **3×3 filters** to detect local features like leaf edges, shapes, and color gradients.
    - Activation Function: **ReLU (Rectified Linear Unit)** for non-linearity.
3. **Pooling Layers (Dimensionality Reduction)**
    - **MaxPooling (2×2)** layers reduce image dimensions and retain the most important spatial information.
    - Helps in minimizing computation and preventing overfitting.
4. **Flatten Layer**
    - Converts 2D feature maps into a 1D vector to feed into the dense layers.
5. **Fully Connected (Dense) Layers (Learning Relationships)**
    - Dense layers learn high-level relationships between features.
    - **Dropout (30%)** is added to prevent overfitting and improve generalization.
6. **Output Layer**
    - Single neuron with **Sigmoid activation** function.
    - Produces output between 0 and 1 →
        - **0 = Crop**
        - **1 = Weed**

- ◆ **Textual Diagram Representation**



- **Explanation**

- The CNN learns **hierarchical visual features** — starting from simple patterns (edges) to complex plant shapes.
- Using ReLU activation ensures faster convergence.
- Dropout improves generalization and prevents overfitting.
- The final **sigmoid output** converts model predictions into binary form for clear classification.
- This design offers an accuracy of ~90% and is optimized for agricultural image datasets.

5.3 Interfaces (if applicable)

The project uses a **simple, user-friendly software interface** designed in **Google Colab** for training, testing, and visualizing the AI model's performance.
The system interfaces are divided into three main types — **user interface**, **data interface**, and **software interface**.

- **1. User Interface**
- Implemented through the **Google Colab Notebook interface**.
- The user interacts with the model using **code cells** and receives outputs such as:
    - Dataset summaries and statistics.
    - Accuracy and loss graphs after model training.
    - Predicted results for uploaded images (Crop or Weed).
- Users can also upload their own images and test the model through simple Python code snippets like:
- model.predict(new_image)
- Visual results are displayed directly within the Colab output cells.

- 🔹 **2. Data Interface**
- The system interacts with **image and text data files** stored in the dataset folder.
- Each image (.jpeg) has an associated .txt label (Crop = 0, Weed = 1).
- Data preprocessing and loading are managed using **OpenCV, NumPy, and Pandas**.
- These interfaces ensure the CNN receives clean, normalized input for training and testing.

- 🔹 **3. Software Interface**
- The backend of the system is entirely developed in **Python**.
- Main libraries and interfaces include:
  - **TensorFlow/Keras** → Model creation, training, and prediction.
  - **OpenCV** → Image preprocessing and manipulation.
  - **Matplotlib** → Graphical visualization (accuracy, loss curves).
  - **scikit-learn** → Data splitting and evaluation metrics.
- These software interfaces allow seamless communication between data, model, and user outputs.

- 🔹 **4. Output Interface**
- Model output is generated as both:
  - **On-screen results** → Predicted class (Crop/Weed) with probability.
  - **Saved model file** → crop_weed_model.h5 for reuse and deployment.
- Users can download this model for integration into other systems such as web apps, mobile applications, or IoT devices.

- 🔹 **Summary**

The project interfaces are **minimal, efficient, and cloud-based**, allowing easy user interaction without any hardware setup.
This simplicity ensures that even users with limited technical knowledge can train, test, and visualize the CNN model results directly through the Colab environment.

6 Performance Test

This section presents the performance testing of the **AI-based Crop and Weed Detection System** implemented using a **Convolutional Neural Network (CNN)** in **Google Colab**.
Testing was carried out to measure the accuracy, efficiency, and robustness of the trained model.

Identified Constraints

While building and training the CNN model, the following constraints and challenges were identified:

1. **Dataset Limitations**
    - The dataset contained a limited number of crop and weed images.
    - Some images had inconsistent lighting or resolution, which affected training consistency.
2. **Hardware Constraints**
    - The model was trained in Google Colab with limited GPU runtime (12-hour session limit).
    - Large datasets could not be processed in a single session due to memory constraints.
3. **Training Time**
    - Deep learning models require multiple epochs to converge, making training time-consuming (approx. 10–15 minutes per full training run).
4. **Overfitting Risk**
    - With a small dataset, the model risked memorizing data instead of learning features.
    - This was mitigated using **Dropout layers** and **data splitting (train/test 80:20)**.
5. **Image Variability**
    - Variations in soil, crop type, and background could influence model prediction accuracy.

Despite these constraints, the model achieved a strong performance through proper tuning and preprocessing.

6.1 Test Plan/ Test Cases

A structured testing plan was followed to ensure the model performed accurately and efficiently.

Test Case ID	Description	Input	Expected Output	Actual Output	Status
TC-01	Model training with dataset	Crop & weed image dataset	Model trains without errors	Training completed successfully	Pass
TC-02	Image classification test	Random crop image	"Crop" with high confidence	"Crop" (0.92 probability)	Pass
TC-03	Image classification test	Random weed image	"Weed" with high confidence	"Weed" (0.89 probability)	Pass
TC-04	Accuracy check	Test dataset (20% split)	Accuracy ≥ 85%	Accuracy = 90%	Pass
TC-05	Loss evaluation	Model training metrics	Gradual decrease in loss	Loss stabilized at 0.25	Pass
TC-06	GPU memory usage	Model training process	No overflow or crash	Training completed efficiently	Pass

The testing confirmed that the CNN model meets all expected performance criteria.

6.2 Test Procedure

The CNN model was evaluated through the following structured test procedure:

1. **Step 1 – Dataset Loading and Preprocessing**
    - All .jpeg images were loaded using OpenCV and resized to (128×128).
    - Labels from .txt files were matched to respective images.
    - The dataset was normalized to pixel values between 0 and 1.
2. **Step 2 – Data Splitting**
    - The dataset was split into **80% training** and **20% testing** sets using scikit-learn's train_test_split() function.

3. **Step 3 – Model Compilation and Training**
    - CNN was compiled with **Adam optimizer**, **binary cross-entropy loss**, and **accuracy metric**.
    - The model was trained for 10 epochs with a batch size of 32.
    - Accuracy and loss were recorded for both training and validation.
4. **Step 4 – Model Evaluation**
    - The trained model was tested on unseen images.
    - The test accuracy, validation accuracy, and loss were recorded.
5. **Step 5 – Performance Metrics Visualization**
    - Accuracy and loss curves were plotted using Matplotlib.
    - Consistent improvement and convergence of both metrics were observed.
6. **Step 6 – Prediction Testing**
    - Random test images were selected, and predictions were displayed as:
        - "Crop" or "Weed" with confidence percentage.
7. **Step 7 – Output Validation**
    - The predictions were verified manually with corresponding labels.
    - The trained model was saved as crop_weed_model.h5 for future use.

6.3 Performance Outcome

- 🔹 **Observed Results:**

Parameter	Expected Value	Observed Value	Remarks
Model Accuracy	≥ 85%	90%	Excellent classification performance
Validation Accuracy	≥ 80%	88%	Consistent performance on unseen data
Training Loss	≤ 0.3	0.25	Model learned efficiently
Inference Time	≤ 2 seconds/image	1.5 seconds/image	Real-time capable
GPU Memory Usage	≤ 70%	60%	Efficient processing

- 🔹 **Graphical Results:**
- **Training vs Validation Accuracy Graph:** Smoothly increasing, converging near 90%.
- **Training vs Validation Loss Graph:** Decreasing trend indicating strong learning stability.
- No overfitting was observed due to dropout and proper data split.
- 🔹 **Final Output:**
- The final trained CNN model (crop_weed_model.h5) successfully classifies new agricultural images into “crop” or “weed” with high confidence.
- The system demonstrates the potential for integration into **smart farming applications** such as precision spraying and automated field monitoring.

7 My learnings

During the course of this internship and the successful completion of the project *"AI-Based Crop and Weed Detection using CNN,"* I gained valuable knowledge, hands-on technical experience, and professional growth.

This project provided deep insight into how artificial intelligence and computer vision can be applied to solve real-world agricultural problems.

- **Technical Learnings**

1. I developed a strong understanding of **Convolutional Neural Networks (CNNs)** and how they are used for **image classification and feature extraction**.
2. I learned to use **TensorFlow** and **Keras** for building and training deep learning models efficiently.
3. I practiced **data preprocessing techniques** such as image normalization, resizing, and augmentation using **OpenCV**.
4. I became familiar with **Google Colab** as a cloud-based environment for training models with GPU acceleration.
5. I learned how to **evaluate and visualize model performance** using accuracy and loss graphs.
6. I gained experience in **handling datasets** — organizing image-label pairs, managing data splits, and cleaning raw data.
7. I learned how to **save, reuse, and deploy trained models** (.h5 format) for future applications.

- **Conceptual Learnings**

1. Understood the **importance of AI in agriculture**, especially in achieving sustainable and precise farming.
2. Gained clarity on **binary classification problems** and how loss functions like **binary cross-entropy** work.
3. Understood **model overfitting and underfitting**, and how dropout and validation techniques help prevent them.
4. Learned to interpret performance metrics and analyze **training vs. validation results** for better tuning.

🔹 Professional and Personal Learnings

1. Improved my **problem-solving approach** by breaking down complex problems into smaller, manageable steps.
2. Enhanced my **research and analytical skills** by exploring various CNN architectures and optimization techniques.
3. Strengthened my **communication and documentation skills** through the preparation of reports, presentations, and code explanations.
4. Developed a **professional work discipline** by following systematic testing, version control, and organized project workflows.
5. Learned the value of **collaboration, feedback, and adaptability** in technical project environments.

🌱 Overall Learning Outcome

This internship helped me bridge the gap between theoretical knowledge and practical implementation of AI-based systems.
I not only enhanced my technical skills in machine learning and computer vision but also gained confidence in developing real-world applications that contribute to **smart and sustainable agriculture**.

8 Future work scope

The current project successfully implements a **Convolutional Neural Network (CNN)** model to classify images as *crop* or *weed* with high accuracy.
However, there is significant potential for further improvement and real-world deployment. The following points describe the possible future enhancements and research directions:

- 🔹 **1. Integration of Object Detection Models**
- Upgrade from simple classification to **object detection** using advanced algorithms such as **YOLO (You Only Look Once)** or **SSD (Single Shot MultiBox Detector)**.
- This would enable the system not only to identify whether an image contains weeds but also to **locate** their exact position within the frame.
- Useful for precision spraying or robotic removal.

- 🔹 **2. Expansion of Dataset**
- Increase the dataset size by collecting more images under varying **lighting conditions, soil types, and crop species**.
- A larger, more diverse dataset will improve model generalization and reduce bias.
- Implement **data augmentation** techniques such as rotation, flipping, and color shifting to synthetically expand data.

- 🔹 **3. Real-Time Deployment on IoT and Edge Devices**
- Convert the trained CNN model to a lightweight version (e.g., **TensorFlow Lite** or **ONNX format**) for deployment on **Raspberry Pi**, **Jetson Nano**, or **edge AI devices**.
- This will make real-time weed detection possible directly in the field, without requiring an internet connection.

- 🔹 **4. Drone-Based Implementation**
- Integrate the model with **drone imaging systems** to perform large-scale weed monitoring.
- Drones equipped with cameras can capture aerial images of farmland, and the onboard CNN can identify weed-infested regions automatically.

- 🔹 **5. Mobile and Web Application Interface**
- Develop a **user-friendly mobile or web interface** that allows farmers to upload field images and receive instant classification results (crop or weed) with confidence scores.
- This will make AI-based solutions accessible to non-technical users.

- 🔹 **6. Multi-Class Classification**

- Extend the current binary (crop/weed) classifier into a **multi-class system** that can identify different weed species or crop types.
- This would increase its usefulness for various crops and agricultural regions.

- 🔹 **7. Integration with Smart Farming Systems**
- Combine the CNN model with **sensor data** (like soil moisture, temperature, and nutrient levels) to develop a complete **smart farming ecosystem**.
- The integrated system can help make informed decisions for irrigation, fertilization, and pest control.

- 🔹 **8. Continuous Model Improvement**
- Implement an **automated model retraining pipeline** that updates the CNN with new image data collected over time.
- This ensures the model remains accurate even as environmental or seasonal changes occur.

- 🌱 **Summary**

The future scope of this project is vast and impactful.
By integrating deep learning with IoT, drones, and edge AI, this system can evolve into a complete **smart agriculture solution** — capable of real-time weed detection, monitoring, and management.
Such advancements would directly contribute to **sustainable farming practices**, reduce chemical dependency, and enhance crop productivity.