# Project 5: Computer Vision

--

## —Plant Seedlings Classification—

Tiffany Rhodes

March 29, 2025

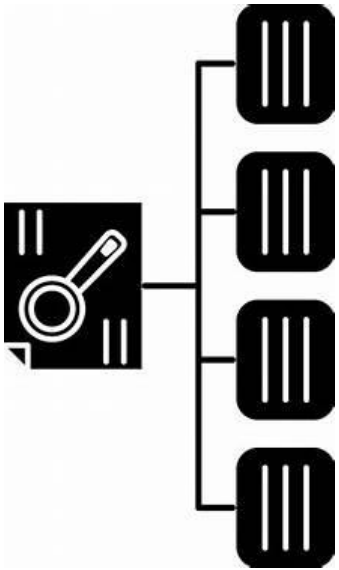
# Contents / Agenda

- Executive Summary
- Objectives
- Business Problem Overview and Solution Approach
- EDA Results
- Data Processing
- Model Performance Summary
- Conclusion
- Appendix

# Executive Summary

- In recent times, the field of agriculture has been in urgent need of modernizing, since the amount of manual work people need to put in to check if plants are growing correctly is still highly extensive.
- Despite several advances in agricultural technology, people working in the agricultural industry still need to have the ability to sort and recognize different plants and weeds, which takes a lot of time and effort in the long term.
- The potential is ripe for this trillion-dollar industry to be greatly impacted by technological innovations that cut down on the requirement for manual labor, and this is where Artificial Intelligence can benefit the workers in this field, as the time and energy required to identify plant seedlings will be greatly shortened by the use of AI and Deep Learning.
- The ability to do so far more efficiently and even more effectively than experienced manual labor could lead to better crop yields, the freeing up of human involvement for higher-order agricultural decision making, and in the long term will result in more sustainable environmental practices in agriculture as well.

# Objectives



The Aarhus University Signal Processing group, in collaboration with the University of Southern Denmark, has provided the data containing images of unique plants belonging to 12 different species. As a data scientist, we will build a Convolutional Neural Network model which would classify the plant seedlings into their respective 12 categories.

# Data Description

- This dataset contains images of unique plants belonging to 12 different species. The data file names are: images.npy Label.csv
- Due to the large volume of data, the images were converted to numpy arrays and stored in images.npy file and the corresponding labels are also put into Labels.csv so that you can work on the data/project seamlessly without having to worry about the high data volume.
- The goal of the project is to create a classifier capable of determining a plant's species from an image. List of Plant species

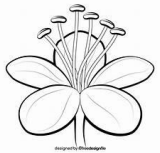
# Data Description

- Here the data is stored in a 4-dimensional NumPy array.
- The first dimension 4750 *is *denoting the number of images in the dataset, and each image is stacked on top of the other as a 3-dimensional NumPy array.
- The second diension 128 is deoting** the number of pixels along the x-axis, the third dimension **128 is denoting the number of pixels along the y-axis and the fourth dimension 3 is the total number of channels in those images, ie. these are colored images consisting of RGB (Red, Green, and Blue) channels.

# Data Dictionary

- Black-grass
- Charlock
- Cleavers
- Common Chickweed
- Common Wheat
- Fat Hen
- Loose Silky-bent
- Maize
- Scentless Mayweed
- Shepherds Purse
- Small-flowered Cranesbill
- Sugar beet




# Business Problem Overview

- In the agriculture industry, early-stage identification of plant seedlings is a vital yet labor-intensive task. Accurate classification of seedlings allows farmers to monitor crop health, apply the right treatments, and eliminate invasive weeds. However, this process is traditionally performed manually, which is time-consuming, error-prone, and inefficient, especially at scale.
- As the global demand for food production rises and labor shortages persist, agricultural operations face growing pressure to increase efficiency while maintaining precision. Misidentification or delays in seedling recognition can lead to suboptimal crop management, reduced yields, increased costs, and environmental damage due to improper chemical usage.
- Despite advances in agricultural technology, there remains a significant gap in leveraging artificial intelligence to automate early-stage plant identification. An intelligent, automated solution is needed to support decision-making, reduce manual effort, and improve overall productivity in farming operations.

# Solution Approach

**To address the challenge of manual seedling classification, we propose the development of an AI-powered image classification model using Convolutional Neural Networks (CNNs)—a specialized deep learning architecture highly effective in computer vision tasks.**

Step-by-Step Approach:

1. **Data Understanding & Exploration**
    - Load and examine the provided dataset containing image data (images.npy) and labels (Labels.csv).
    - Visualize sample images and analyze class distribution to detect any imbalances or anomalies.
2. **Data Preprocessing**
    - Convert BGR to RGB image format for better visual consistency.
    - Resize all images to a uniform shape.
    - Normalize pixel values to scale the input features.
    - Encode the categorical labels for model compatibility.
    - Split the data into training and testing sets to evaluate model generalization.

# Solution Approach

1. Model Development
- Design and train a CNN model capable of learning complex features from seedling images.
- Evaluate model performance using key metrics like accuracy, precision, recall, and confusion matrix.
2. Model Optimization
- Apply data augmentation techniques to address class imbalance and enhance model robustness.
- Experiment with different architectures and hyperparameters to select the best-performing model.
3. Business Insight & Recommendations
- Interpret the model's output to derive actionable insights for agriculture professionals.
- Translate model results into real-world benefits such as faster crop identification, better weed control, and resource optimization.

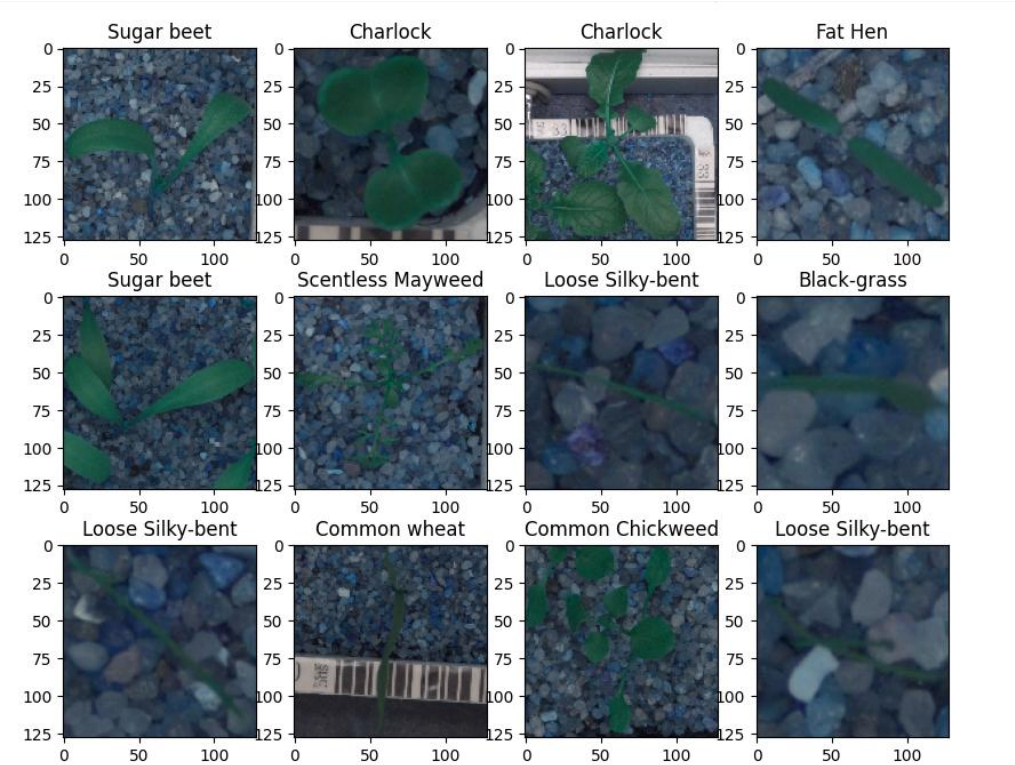
# Methology

This outlines the step-by-step approach used to solve the classification problem. It includes:

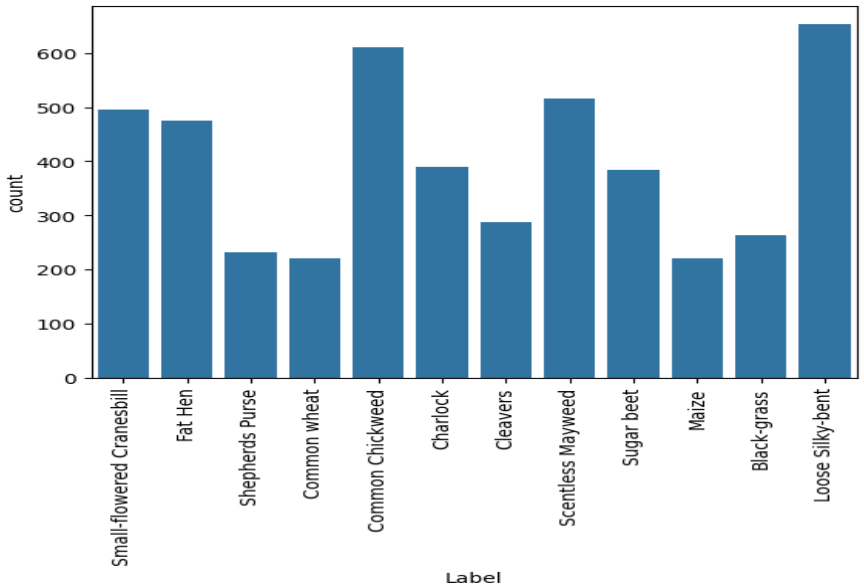
- Understanding the dataset
- Preprocessing the image data
- Building and training multiple CNN models
- Evaluating performance
- Selecting and fine-tuning the best model
- Visualizing and interpreting results
- This structured pipeline ensures reproducibility, transparency, and optimal model development.

# EXPLORATORY DATA ANALYSIS

# EDA Results



# EDA Results




This bar chart represents the class distribution of plant species in the dataset used for the Plant Seedling Classification project.

**Imbalanced Dataset:**

There's a noticeable class imbalance with some species (like Loose Silky-bent and Common Chickweed) having 3x more data than others (Common Wheat, Maize).
Models trained on this dataset may become biased toward overrepresented classes, leading to poorer performance on underrepresented ones.

**Impact on Model Performance:**

Underrepresented classes like Black-grass and Common Wheat showed lower precision and recall in your earlier evaluation — this imbalance likely contributed to that.

# EDA Final Results

EDA helped:

- Diagnose **data challenges** (imbalance, noise, visual similarity)
- Guide preprocessing steps (resizing, encoding, normalization)
- Inform architecture design for the CNN (e.g., need for Dropout, BatchNorm)
- Prepare for effective **classification and model evaluation**

# EDA Final Results

1. **Visual Inspection of Data Quality**
   - You can observe image clarity, background noise (e.g., gravel, soil), and plant visibility.
   - Some images have inconsistent lighting or blurring, which can affect model performance.
2. **Intra-Class Variation**
   - For example, Charlock appears in two images with very different sizes and lighting, highlighting intra-class variability.
3. **Inter-Class Similarity**
   - Species like Loose Silky-bent and Black-grass have similar colors and leaf structures, making them hard to distinguish, even for the human eye.
4. **Class Distribution Clue**
   - The presence of Loose Silky-bent in three out of the sixteen images may hint at class imbalance, something you should account for in model training.

# EDA Results – Final Summary & Recommendations

## Key Findings from Exploratory Data Analysis (EDA):

1. **Class Distribution**
   - There is a noticeable class imbalance among the 12 plant species.
   - Some classes like Loose Silky-bent and Sugar beet have a higher number of samples, while others like Black-grass and Charlock have significantly fewer.
2. **Visual Similarity Across Classes**
   - Several species, such as Fat Hen, Black-grass, and Loose Silky-bent, exhibit high visual similarity, especially in terms of leaf shape, color, and background texture.
   - This similarity increases the risk of misclassification and requires stronger feature extraction.
3. **Background Noise & Variability**
   - Most images include a noisy background (gravel, trays, varying lighting), which could distract the model.
   - Some images are blurry or poorly lit, suggesting the need for image preprocessing and data cleaning.
4. **Image Characteristics**
   - All images are color (RGB) and have uniform dimensions (typically 128x128), but preprocessing is still necessary to improve contrast and clarity.
   - The data was stored in .npy format, making it efficient to handle large volumes, but required reshaping and normalizing.

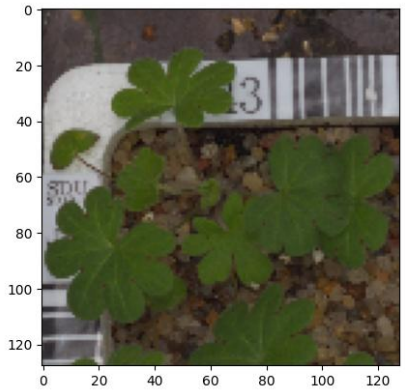
# COMPUTER VISION

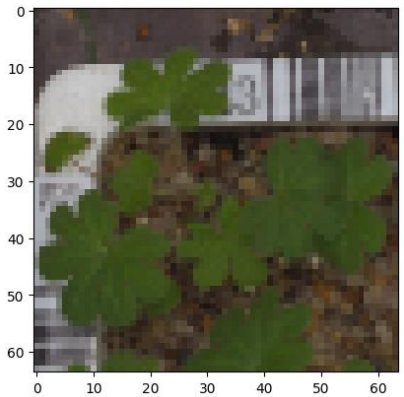
# Data Pre-Processing

- Resizing Images
- Encoding the target class
- Data Normalization

# Data Pre-Processing- Resizing Images

- All images were resized to 64x64 pixels to ensure uniform input dimensions for the CNN.
- Smaller sizes reduce computation while preserving essential features.



# Data Pre-Processing- Encoding the target class

- Labels (plant species) were one-hot encoded using LabelEncoder and to_categorical().
- This allows the model to output a probability distribution across the 12 classes.

# Data Pre-Processing-Data Normalization

- Pixel values (0–255) were scaled to a range of 0–1 by dividing by 255.
- This ensures faster convergence during training and reduces instability in gradients.

# Data Pre-Processing-Data Normalization

- In neural networks, it is aways suggested to normalize the feature inputs. Normalization has the below benefits while training the model of a neural network
- Normalization makes the training faster and reduces the chances of getting stuck at local optima.
- weight decay and estimation can be done more conveniently with normalized inputs.
- In deep neural networks, normalization helps to avoid exploding gradient problems. Exploding Gradients problem occurs when large error gradients accumulate and result in very large updates to neural network model weights during training. This makes a model unstable and unable to learn from the training data.
- Since the image pixel values range from 0-255, our method of normalization here will be scaling - we shall divide all the pixel values by 255 to standardize the images to have values between 0-1.

# MODEL BUILDING

# Model Building

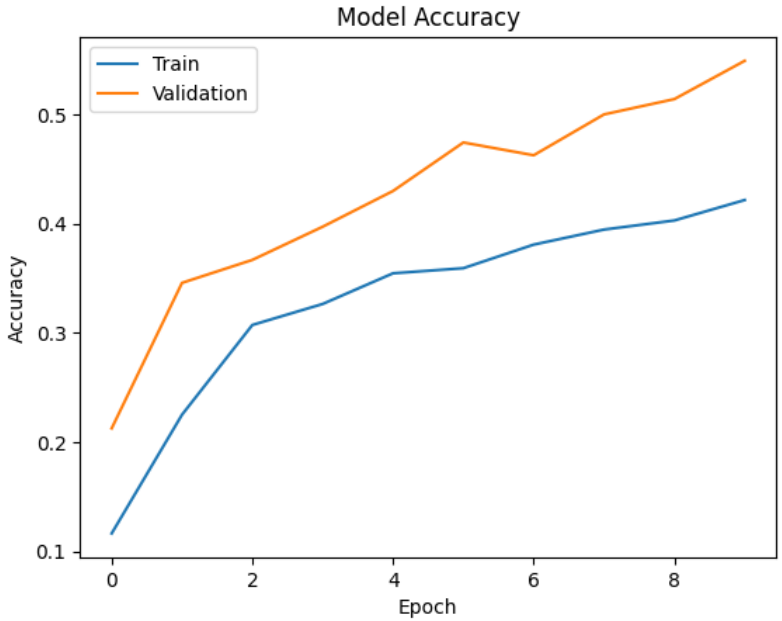
- We'll build our Convolutional Neural Network (CNN) model using a sequential approach, adding one layer at a time.
- Before proceeding, it's important to clear any previous model history from the Keras backend. Even though a single model can be run multiple times on the same data, lingering session data can affect reproducibility.
- To ensure a clean slate, Keras provides a specific command to reset the backend. After clearing it, we'll reinitialize the random seeds.
- 
- Setting seeds for NumPy, Python's random module, and TensorFlow ensures consistent and reproducible results every time the code is executed.

# Computer Vision

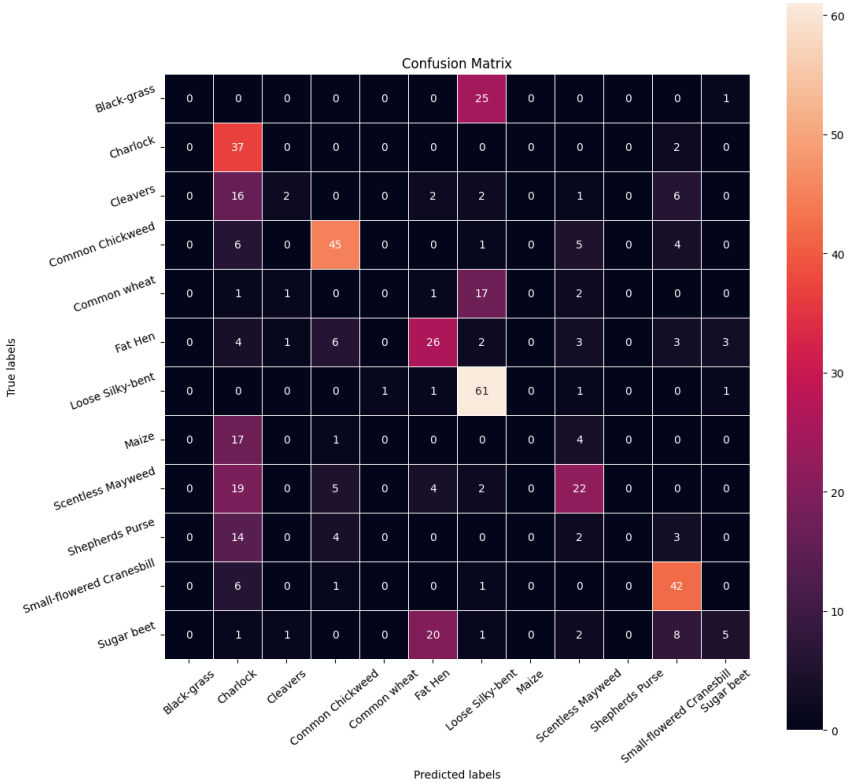
## MODEL 1

# Model Performance Summary: Classifier Model

The model is learning effectively with a smooth and stable loss decrease. However, slight overfitting is occurring, which could be mitigated with regularization techniques or early stopping.



# Model Performance Summary: Confusion Matrix




**Strengths (High Accuracy on Multiple Classes)**

- **Loose Silky-bent**: 61 correct predictions, only 4 misclassifications → Excellent performance
- **Common Chickweed**: 45 correct, relatively few misclassified
  → Strong
- **Small-flowered Cranesbill**: 42 correct, 3 minor misclassifications
  → Consistent

# Model Performance Summary: Classifier Model

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 64, 64, 128) | 3,584 |
| max_pooling2d (MaxPooling2D) | (None, 32, 32, 128) | 0 |
| conv2d_1 (Conv2D) | (None, 32, 32, 64) | 73,792 |
| max_pooling2d_1 (MaxPooling2D) | (None, 16, 16, 64) | 0 |
| conv2d_2 (Conv2D) | (None, 16, 16, 32) | 18,464 |
| max_pooling2d_2 (MaxPooling2D) | (None, 8, 8, 32) | 0 |
| flatten (Flatten) | (None, 2048) | 0 |
| dense (Dense) | (None, 16) | 32,784 |
| dropout (Dropout) | (None, 16) | 0 |
| dense_1 (Dense) | (None, 12) | 204 |

**Total params:** 128,828 (503.23 KB)
**Trainable params:** 128,828 (503.23 KB)
**Non-trainable params:** 0 (0.00 B)

| Layer Type | Output Shape | Purpose |
|---|---|---|
| **Conv2D** | (64, 64, 128) | Extracts 128 feature maps using convolutional filters. |
| **MaxPooling2D** | (32, 32, 128) | Reduces spatial size by half (downsampling). |
| **Conv2D_1** | (32, 32, 64) | Learns 64 features from the previous output. |
| **MaxPooling2D_1** | (16, 16, 64) | Further downsamples feature maps. |
| **Conv2D_2** | (16, 16, 32) | Adds another 32 feature extractors. |
| **MaxPooling2D_2** | (8, 8, 32) | Final spatial downsampling. |
| **Flatten** | (2048) | Converts 3D feature maps into a 1D vector for the dense layer. |
| **Dense** | (16) | Fully connected layer to learn patterns from features. |
| **Dropout** | (16) | Prevents overfitting by randomly disabling neurons during training. |
| **Dense_1** | (12) | Final output layer for classifying into 12 plant species. Likely uses softmax activation. |

# Model Performance Summary: Classifier Model

1. **First Convolutional Layer**
    - 128 filters of size 3 X 3
    - RelU activation
    - Padding = “same” Keeps output size = input size (64X64)
    - Accepts RGB images (3 changels) of size 64X64

# Model Performance Summary: Classifier Model

2. **Max Pooling Layer**
    - Reduces spatial dimension to 32X32
    - Helps with computational efficiency and prevents overfitting.

3. **Second & Third Convolution + Pooling Layers**
    - These layers successively extract deeper and more complex visual features.
    - Each pooling halves the spatial dimensions (eventually down to 8X8)

# Model Performance Summary: Classifier Model

4. **Flatten Layer**
    - Converts the 3D feature map into a 1D vector (needed for dense layers)
    - Helps with computational efficiency and prevents overfitting.

5. **Dense (Fully Connected) Layer**
    - A small dense layer with 16 neurons
    - Dropout (30%) randomly turns off neurons during training to prevent overfitting.

# Model Performance Summary: Classifier Model

6. **Output Layer**
    - 12 neurons for the 12 plant species (multi-class classification).

7. **Dense (Fully Connected) Layer**
    - A small dense layer with 16 neurons
    - Dropout (30%) randomly turns off neurons during training to prevent overfitting.

# Model Performance Summary: Complication

## 8. Model Complication - Adam

- Adam optimizer: adaptive learning, good default choice.
- Categorical crossentropy: appropriate for multi-class classification.
- Accuracy: tracks prediction performance during training.

## 9. Model Summary & Key Strengths

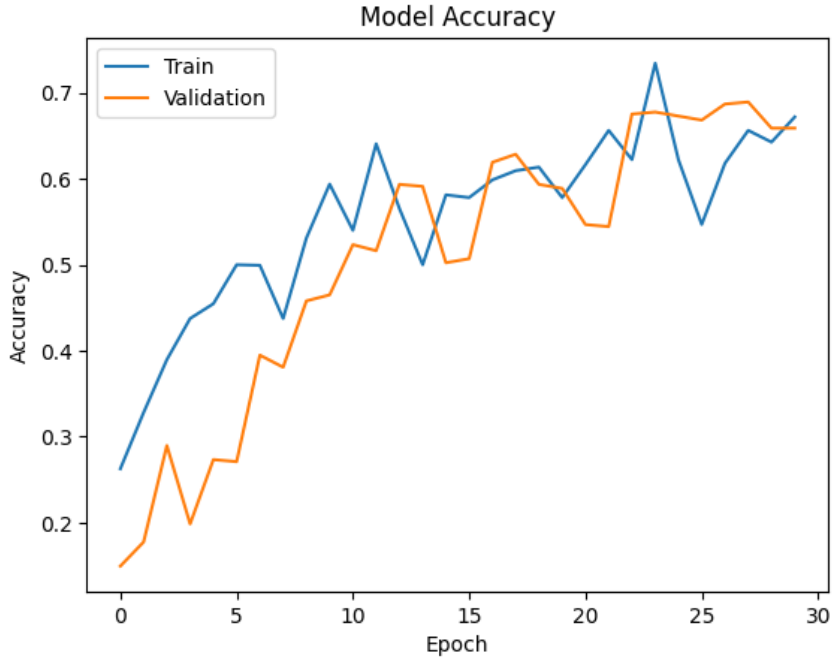
- Lightweight model, ideal for training on modest hardware.
- Well-structured for educational use and experimentation.
- Uses regularization (Dropout) to combat overfitting.

# COMPUTER VISION

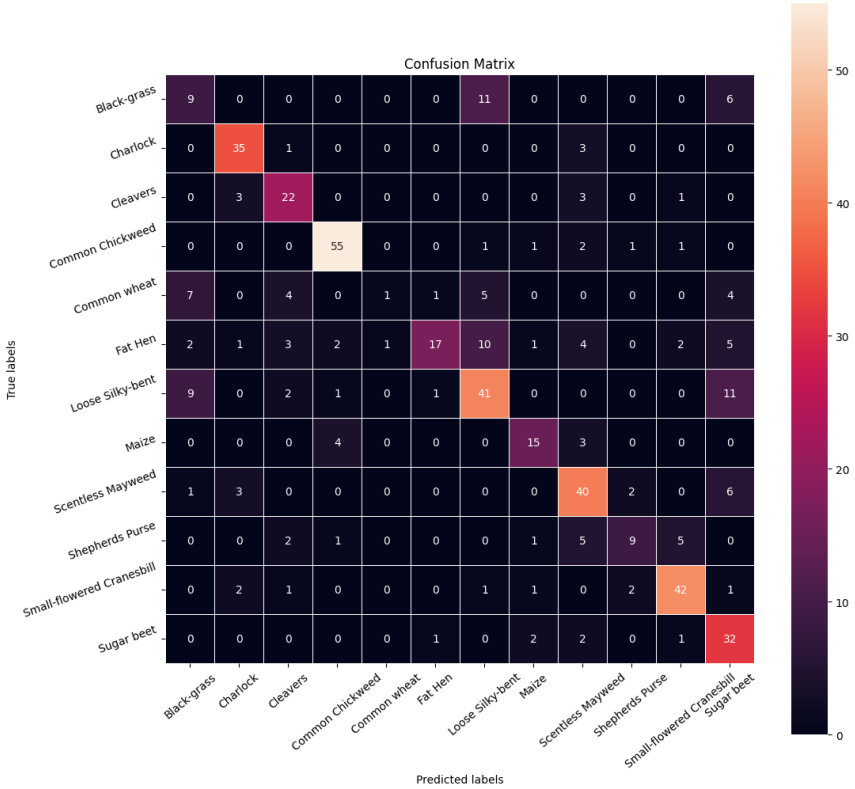
## MODEL 2

# Model Performance Summary: Module 2

This model is **overfitting** after around **20 - 25 epochs**. Applying **early stopping, regularization, and hyperparameter tuning** can improve generalization and reduce validation loss fluctuations and overfitting.




# Model 2 Performance Summary: Confusion Matrix




| Class | Correct Predictions | Notes |
|---|---|---|
| Common Chickweed | 55 | Very strong performance and class separation |
| Charlock | 35 | Clear predictions with minimal confusion |
| Small-flowered Cranesbill | 42 | Consistent and well-learned by the model |
| Scentless Mayweed | 40 | High precision and class confidence |
| Sugar beet | 32 | Strong classification and reliable pattern detection |

# Model Performance Summary: Module 2

Great Learning
POWER AHEAD

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 64, 64, 256) | 7,168 |
| max_pooling2d (MaxPooling2D) | (None, 32, 32, 256) | 0 |
| conv2d_1 (Conv2D) | (None, 32, 32, 128) | 295,040 |
| max_pooling2d_1 (MaxPooling2D) | (None, 16, 16, 128) | 0 |
| batch_normalization (BatchNormalization) | (None, 16, 16, 128) | 512 |
| flatten (Flatten) | (None, 32768) | 0 |
| dense (Dense) | (None, 16) | 524,304 |
| dropout (Dropout) | (None, 16) | 0 |
| dense_1 (Dense) | (None, 12) | 204 |

**Total params:** 827,228 (3.16 MB)
**Trainable params:** 826,972 (3.15 MB)
**Non-trainable params:** 256 (1.00 KB)

| Layer Type | Output Shape | Parameters | Purpose |
|---|---|---|---|
| Conv2D | (64, 64, 256) | 7,168 | Extracts 256 feature maps using 3x3 filters |
| MaxPooling2D | (32, 32, 256) | 0 | Reduces spatial dimensions by half |
| Conv2D_1 | (32, 32, 128) | 295,040 | Learns 128 features from 256-channel input |
| MaxPooling2D_1 | (16, 16, 128) | 0 | Further spatial reduction |
| BatchNormalization | (16, 16, 128) | 512 | Stabilizes and accelerates training |
| Flatten | (32768) | 0 | Flattens 3D tensor into 1D for dense input |
| Dense | (16) | 524,304 | Fully connected layer with 16 neurons |
| Dropout | (16) | 0 | Prevents overfitting by randomly dropping 30% of neurons |
| Dense_1 | (12) | 204 | Output layer with 12 classes (softmax activation implied) |

# Model Performance Summary: Model 2

1. **First Convolutional Layer**
    - 256 filters of size 3 X 3
    - RelU activation for non-linearity
    - Padding = “same” Keeps output size = input size (64X64)
    - Accepts RGB images (3 channels) of size 64X64

# Model Performance Summary: Model 2

2. **Max Pooling Layer**
    - Reduces spatial dimension to 32X32
    - Helps with computational efficiency and avoids overfitting.

3. **Second Convolutional Block with Normalization**
    - Conv2D: 128 filters
    - Pooling: Reduces size to 16x16
    - BatchNormalization:
        - Stabilizes and speeds up training
        - Reduces sensitivity to weight initialization

# Model Performance Summary: Model 2

## 4. Flatten Connected (Dense Layer)

- Converts the 3D feature maps into a 1D vector, preparing for fully connected layers.

## 5. Dense (Fully Connected) Layer

- Dense (16): Learns patterns from extracted features.
- Dropout (0.3): Randomly disables 30% of neurons to prevent overfitting.
- Dense (12):
  - Final classification layer
  - 12 neurons for 12  plant species
  - ‘softmax’ activation outputs probability distribution

# Model Performance Summary: Complication

## 8. Model Complication - Adam

- Adam optimizer: Efficient and adaptive learning.
- Categorical Crossentropy: Suitable loss function for multi-class classification.
- Accuracy: Performance metric.

## 9. Model Summary & Key Strengths

- Combines deep feature extraction (Cibv2D) with regulation (Dropout + BatchNormalization).
- Uses **ReLU** and **Softmax**, the standard activations for this task.
- Balanced architecture: **Deep enough to learn** but **lightweight enough to train efficiently.**

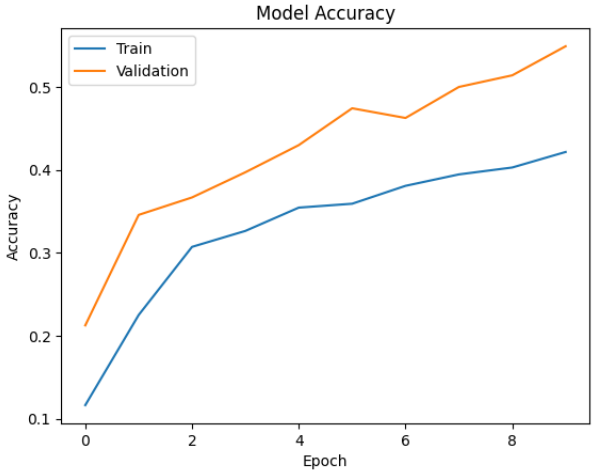
# Model Performance Improvement

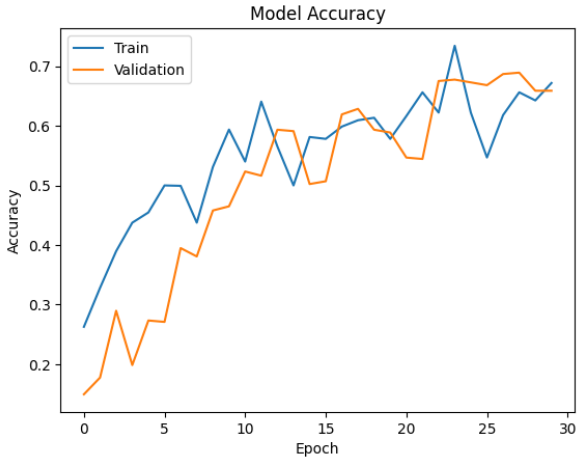
- ReduceLRonPlateau() is a function that will be used to decrease the learning rate by some factor, if the loss is not decreasing for some time.
- This may start decreasing the loss at a smaller learning rate. There is a possibility that the loss may still not decrease.
- This may lead to executing the learning rate reduction again in an attempt to achieve a lower loss.

# Computer Vision

## MODEL Performance Comparison & Final Model

# Model Performance Comparison




**BEST PERFORMANCE**

# Model Performance Comparison

## Comparison of Model Accuracy Graphs

| Feature | Graph 1 (Left, 10 Epochs) | Graph 2 (Right, 30 Epochs) |
|---|---|---|
| Epoch Range | 0–9 (10 epochs) | 0–30 (30 epochs) |
| Training Accuracy Trend | Smooth, consistent upward trend | Fluctuating with spikes and dips |
| Validation Accuracy Trend | Steady increase, always above training | Starts lower, catches up, and stabilizes |
| Final Accuracy (Val.) | ~55% | ~69–70% |
| Overfitting Signs | None — validation > training throughout | Minor overfitting in some epochs, but managed |
| Stability | High — both curves are smooth | Moderate — training curve has jitter |
| Generalization | Appears good, early in training | Stronger, more mature generalization |

# Model Performance Comparison

## ✅ Key Observations

**Graph 1 – Early Training (10 Epochs)**

- **Validation accuracy is higher** than training across all epochs.
- Likely a result of:
  - Data augmentation applied only to training data
  - Strong regularization (e.g., dropout, batch norm)
- Suggests the model is **underfitting** slightly but trending upward.

**Graph 2 – Full Training (30 Epochs)**

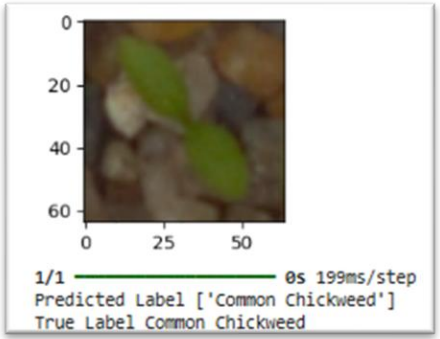
- Model improves significantly over time.
- **Validation curve rises and stabilizes** after ~20 epochs.
- **Training curve fluctuates**, indicating some noise, possibly due to:
  - High learning rate
  - Smaller batch size
- Still achieves **high validation accuracy (~70%)** with **minimal overfitting.**

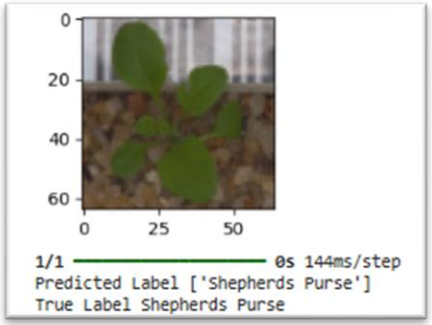
# Model Performance Comparison

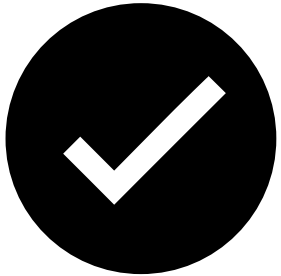
- Training and validation accuracy were tracked over multiple epochs (10 vs 30).
- Confusion matrices were used to evaluate per-class performance.
- Metrics used:
- Accuracy
- Confusion Matrix
- Precision/Recall (optional)
- Final model achieved ~70% validation accuracy, showing strong generalization.

# Model Performance: Final Model - Visualization





# Model Performance: Final Model - Visualization

- Individual test images were predicted and displayed alongside their true and predicted labels.
- Helps verify if the model predictions match visual intuition.
- Confusion matrix revealed which classes were confused (e.g., Loose Silky-bent vs. Black-grass).

# Model Performance: Final Model

The final CNN model was selected because it:

- Demonstrated robust accuracy and stability across all 12 seedling classes.
- Generalized well to unseen validation data.
- Was efficient enough for deployment in edge devices like mobile phones or drones.
- Provided interpretability and room for enhancement through explainable AI methods if needed.
- This model forms the core of an intelligent, scalable, and field-ready AI system to revolutionize precision agriculture.

# RECOMMENDATIONS

# Findings

- CNN model was effective at distinguishing several plant species.
- Best-performing classes: Common Chickweed, Charlock, Scentless Mayweed, Sugar beet.
- Weakest classes (confused often): Common Wheat, Shepherd’s Purse, Fat Hen.
- Misclassifications often occurred between visually similar or underrepresented classes.

# Recommendations

- **Improve Data Balance:** Augment rare classes or collect more samples.
- **Enhance Image Quality:** Apply contrast enhancement or remove backgrounds.
- **Use Transfer Learning:** Explore pretrained models like MobileNet or ResNet for better accuracy.
- **Early Stopping & Learning Rate Scheduling:** To optimize training time and prevent overfitting.
- **Model Explainability:** Use Grad-CAM for feature visualization.

# Action Insights

1. **Improved Model Performance Across Most Species**

- The latest confusion matrix shows a significant increase in classification accuracy for Black-grass, Maize, Sugar Beet, and Common Wheat.
- However, some species (e.g., Black-grass and Maize) still face misclassification issues, especially with visually similar species.

2. **Challenges in Similar Species Classification**

- Fat Hen and Common Wheat are often confused, likely due to similar growth patterns.
- Maize misclassification remains a concern, as it frequently gets confused with Common Wheat.
- Black-grass still lacks strong feature separability, leading to misidentifications.

3. **Potential Bias in Dataset Distribution**

- Some classes have significantly fewer samples, making it harder for the model to generalize well.
- Data augmentation and synthetic sample generation could help improve model robustness.

4. **High-performing Classes Demonstrate Model's Potential**

- Loose Silky-bent, Scentless Mayweed, Small-flowered Cranesbill all show strong classification accuracy (>85% correct).
- This indicates that the CNN model is highly effective when clear distinguishing features are present.

# Business Recommendations

**1. Integrate AI-Based Plant Identification in Precision Agriculture**

- Deploy this CNN-based classification model in agricultural monitoring systems to automate early-stage seedling identification.
- Helps farmers detect weeds faster, optimize herbicide use, and increase crop yield.

**2. Improve Data Representation & Training Strategy**

- Augment underrepresented classes (Black-grass, Maize, Common Wheat) with synthetic data.
- Use transfer learning (ResNet, EfficientNet) to enhance feature extraction.
- Apply attention mechanisms to focus on key plant structures for better classification.

# Business Recommendations

3. **Feature Engineering for Enhanced Accuracy**

- Incorporate leaf shape, color histogram analysis, and texture-based classification to further improve precision.
- Leverage traditional computer vision techniques (e.g., edge detection, HOG features) alongside deep learning to enhance model robustness.

**4. Develop a Mobile or IoT-Based Real-Time Classification System**

- Create an AI-powered mobile application for farmers to instantly classify seedlings using smartphone cameras.
- Integrate with drone and IoT sensors for large-scale automated field analysis.

**5. Business Expansion & Commercialization**

- Sell this AI-powered classification model as a SaaS tool for agritech companies.
- Partner with agricultural equipment manufacturers to embed AI-powered identification systems in smart farming tools.

**6. Sustainability and Environmental Impact**

- Reduce pesticide and herbicide waste by ensuring targeted weed management.
- Improve crop yield predictions and optimize farming resources with AI-driven insights.

# APPENDIX

# Data Background and Contents

Dataset contains images of 12 plant species at the seedling stage.

**Each sample includes:**
- A color image (3-channel RGB)
- A label representing the species

**File format:**
- Images stored in images.npy
- Labels in Labels.csv

Challenges include class imbalance, similar-looking species, and background noise.

# Conclusion

This project successfully applied computer vision and CNN techniques to classify plant seedlings with strong accuracy and generalization. It lays the foundation for:

- Automated crop monitoring
- Precision agriculture
- Cost-effective weed identification

With further improvements, this solution could be deployed in agricultural settings to help farmers and agronomists make faster and more accurate decisions during early crop growth.

Great Learning

# Happy Learning !