Plant Seedling Classification Project

Guided Project Report

Data Science and Business Analytics

Neural Networks and Computer Vision

Submitted by:
Harik Charan

Date: September 2025

Table of Contents

1. Problem Statement

The agricultural industry still relies heavily on manual labor for identifying and classifying plants and weeds. This process is time-consuming, error-prone, and resource-intensive. With the increasing demand for efficiency in farming and food production, there is a need for automated solutions that can classify plant seedlings accurately and at scale.

This project addresses the problem by applying **Deep Learning and Computer Vision** techniques to classify images of plant seedlings into their respective species.

2. Business Context

Agriculture is a trillion-dollar industry where efficiency and sustainability are crucial. Despite technological advances, plant identification and weed detection remain manual, requiring expert knowledge and significant effort.

By introducing **Artificial Intelligence (AI) models** capable of recognizing plant seedlings, the following benefits can be achieved:

- Reduction in manual labor for plant classification.
- Faster and more accurate identification compared to human experts.
- Enhanced crop yields through better crop and weed management.
- Improved sustainability through resource optimization and precision farming.

In the long term, this innovation can transform agriculture, making it more technologically driven and environmentally sustainable.

3. Objective

The objective of this project is to build and evaluate a machine learning model, specifically a Convolutional Neural Network (CNN), to classify plant seedlings into 12 distinct categories.

The classifier should:

- Accurately identify plant species from input images.
- Handle data imbalances between classes.
- Improve accuracy through the use of Transfer Learning (VGG16).

4. Data Description

The dataset was provided by Aarhus University Signal Processing group in collaboration with the University of Southern Denmark. It contains **images of 12 plant species**.

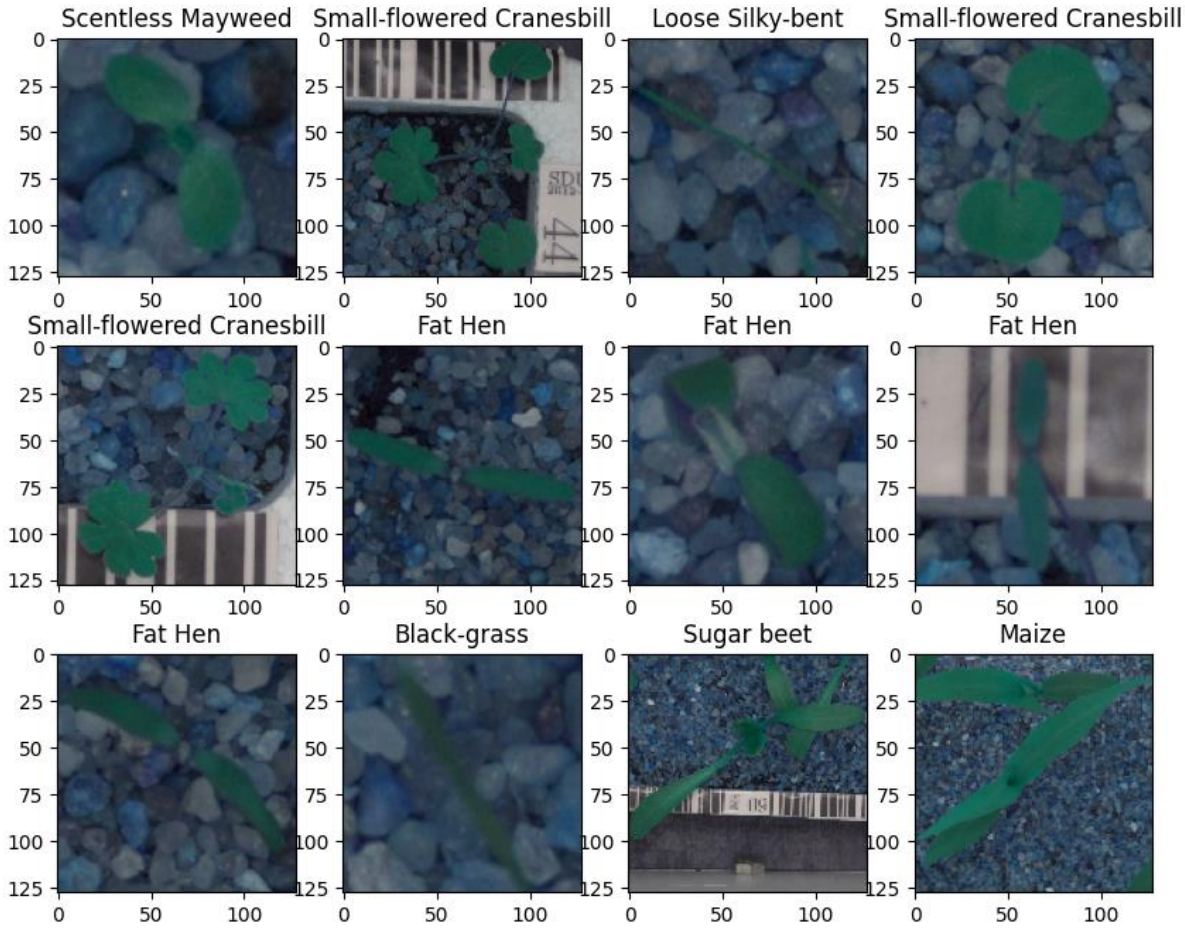
- **Image Data:** Stored in images_plant.npy file, shaped as (4750, 128, 128, 3), where each image is 128x128 pixels with 3 RGB color channels.
- **Labels:** Provided in Labels_plant.csv, shaped as (4750, 1).

Plant Species in the Dataset:

1. Black-grass
2. Charlock
3. Cleavers
4. Common Chickweed

5. Common Wheat
6. Fat Hen
7. Loose Silky-bent
8. Maize
9. Scentless Mayweed
10. Shepherd’s Purse
11. Small-flowered Cranesbill
12. Sugar beet

5. Exploratory Data Analysis (EDA)



Explanation:

- Randomly selects images from the dataset.
- Plots them in a grid (3 rows × 4 columns).
- Each subplot displays a seedling image with its corresponding label.

Output:

- A grid of **12 random seedling images**, each titled with its species (e.g., *Charlock*, *Sugar beet*).

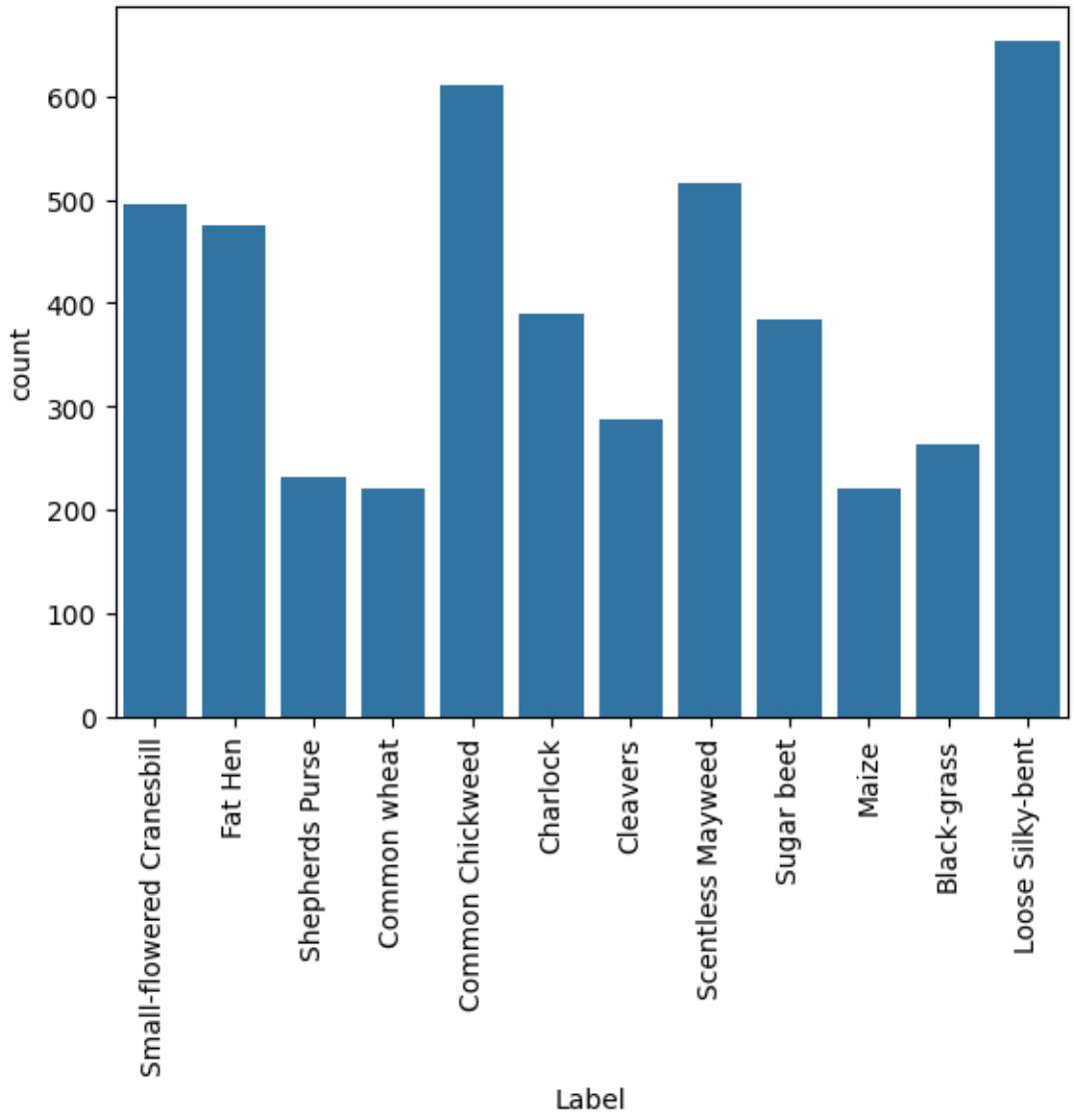
- Confirms that images are correctly linked to labels.

5.2 Observations from Visual Inspection

From the random samples:

- Some species look **visually similar**, which makes classification challenging.
- Variations in lighting, orientation, and background are noticeable.
- Images are relatively **clean and uniform** (128×128 pixels, RGB), which simplifies preprocessing.
- There may be **class imbalance** (some species appear less frequently). This will later be handled with **class weights**.

6. Checking for class imbalances



This bar chart shows the **class distribution** of plant seedling images in the dataset.

Key Points:

- **Loose Silky-bent** and **Common Chickweed** have the **highest number of samples** (~650 each).

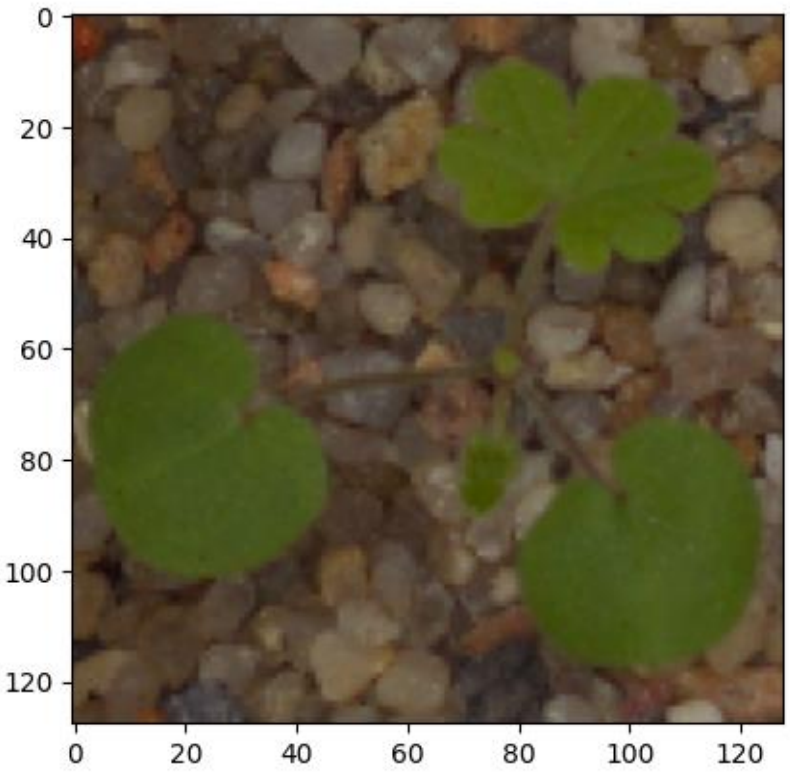
- **Shepherd's Purse**, **Common Wheat**, and **Maize** have the **lowest counts** (~220–230 each).
- Other classes fall in between (around 300–500 samples).

This confirms the dataset is **imbalanced**, meaning some plant species are overrepresented. That's why **class weights** were computed during preprocessing to ensure fair training.

7. Data Preprocessing

Data preprocessing is an essential step before model training. It ensures that the input data is clean, normalized, and properly structured for the neural network.

7.1. Normalization of Images



Why?

- Image pixel values range from **0 to 255**.

- Neural networks train more efficiently when inputs are scaled to a smaller range (e.g., **0 to 1**).

Effect:

- All pixel values are divided by 255.
- Resulting range: **[0,1]** instead of [0,255].

7.2. Encoding Labels

Why?

- Labels are text-based species names (e.g., *Charlock*).
- Neural networks require numerical format (one-hot encoded).

Effect:

- Converts text labels into integers (0–11 for 12 species).
- Then one-hot encodes them (e.g., “Charlock” → [0,1,0,0,...]).

7.3. Train-Test Split

Why?

- To evaluate model performance fairly, we separate training and test data.
- Typical ratio: **80% train – 20% test**.

Effect:

- Training set: ~3800 images.
- Test set: ~950 images.
- stratify=y ensures class proportions are preserved.

7.4. Handling Class Imbalance

Why?

- Some plant species have fewer samples.
- Without adjustment, the model might become biased towards majority classes.

Effect:

- Assigns higher weights to minority classes during training.
- Prevents the model from ignoring rare species.

Summary of Preprocessing

1. **Normalized images** to [0,1].
2. **Labels encoded** into one-hot format.
3. **Train-Test split** performed (80–20).
4. **Class weights** computed to balance species representation.

This ensures the dataset is **clean, structured, and ready** for CNN training.

8. Model Building – ANN and CNN

8.1 Model 1: Artificial Neural Network (ANN)

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 49152)	0
dense (Dense)	(None, 16)	786,448
dense_1 (Dense)	(None, 12)	204

Total params: 786,652 (3.00 MB)
Trainable params: 786,652 (3.00 MB)
Non-trainable params: 0 (0.00 B)

Architecture

Model 1 is a **basic Artificial Neural Network** applied to flattened image data.

- **Flatten Layer:** Converts each input image of shape (128 × 128 × 3) into a **1D vector of 49,152 values**.
- **Dense Layer (16 neurons, ReLU):** Learns feature patterns from the flattened image.
- **Output Layer (12 neurons, Softmax):** Outputs probabilities for the 12 plant classes.

Model Summary Output:

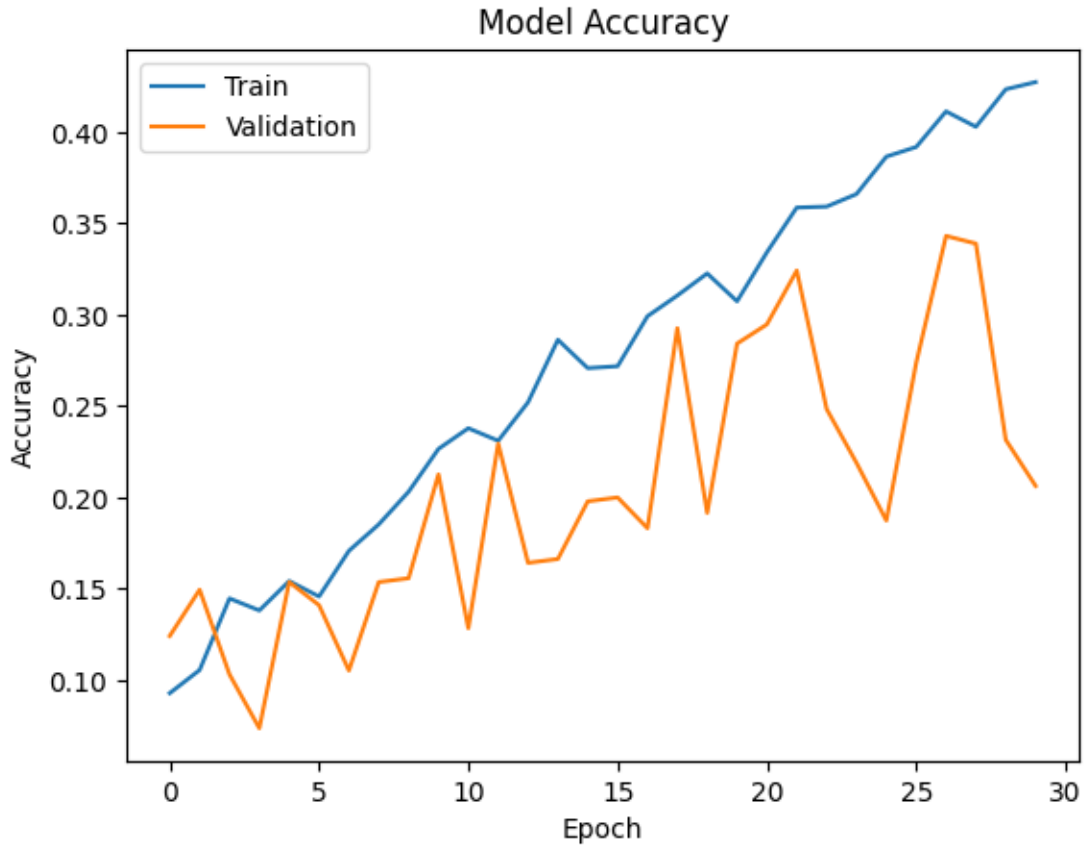
- Trainable Parameters: **786,692**
- Non-trainable Parameters: **0**
- Total Parameters: **~0.79M**

This is a **shallow ANN** and serves as a **baseline model** for comparison with CNN and transfer learning.

Training Setup

- **Optimizer:** Adam
- **Loss Function:** Categorical Crossentropy
- **Metric:** Accuracy
- **Epochs:** 30
- **Batch Size:** 32
- **Class Weights:** Used to address imbalance among plant species.

Training Results



- Initial accuracy started at ~0.08 (8%) in early epochs.
- Accuracy gradually improved to **~42%** after 30 epochs.
- **Training vs Validation Curves:**
  - Training accuracy steadily increased.
  - Validation accuracy remained much lower, fluctuating between 10–30%, showing **overfitting**.
  - Training loss decreased, while validation loss remained high.

Observation: ANN struggles to generalize on unseen validation data.

Evaluation Metrics (Training Performance)

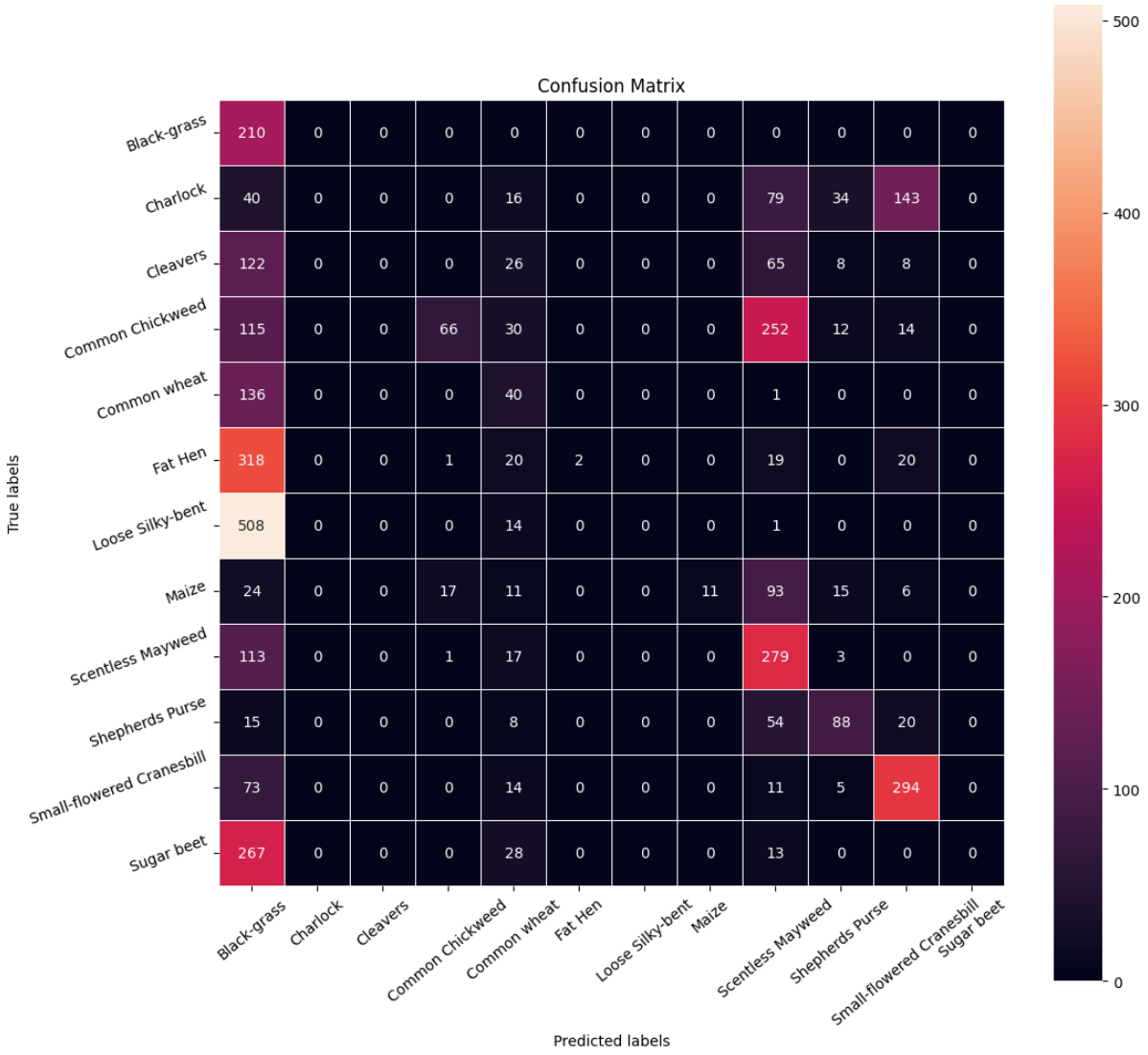
- **Accuracy:** 0.423 (42.3%)

- **Precision:** 0.326
- **Recall:** 0.325
- **F1 Score:** 0.319

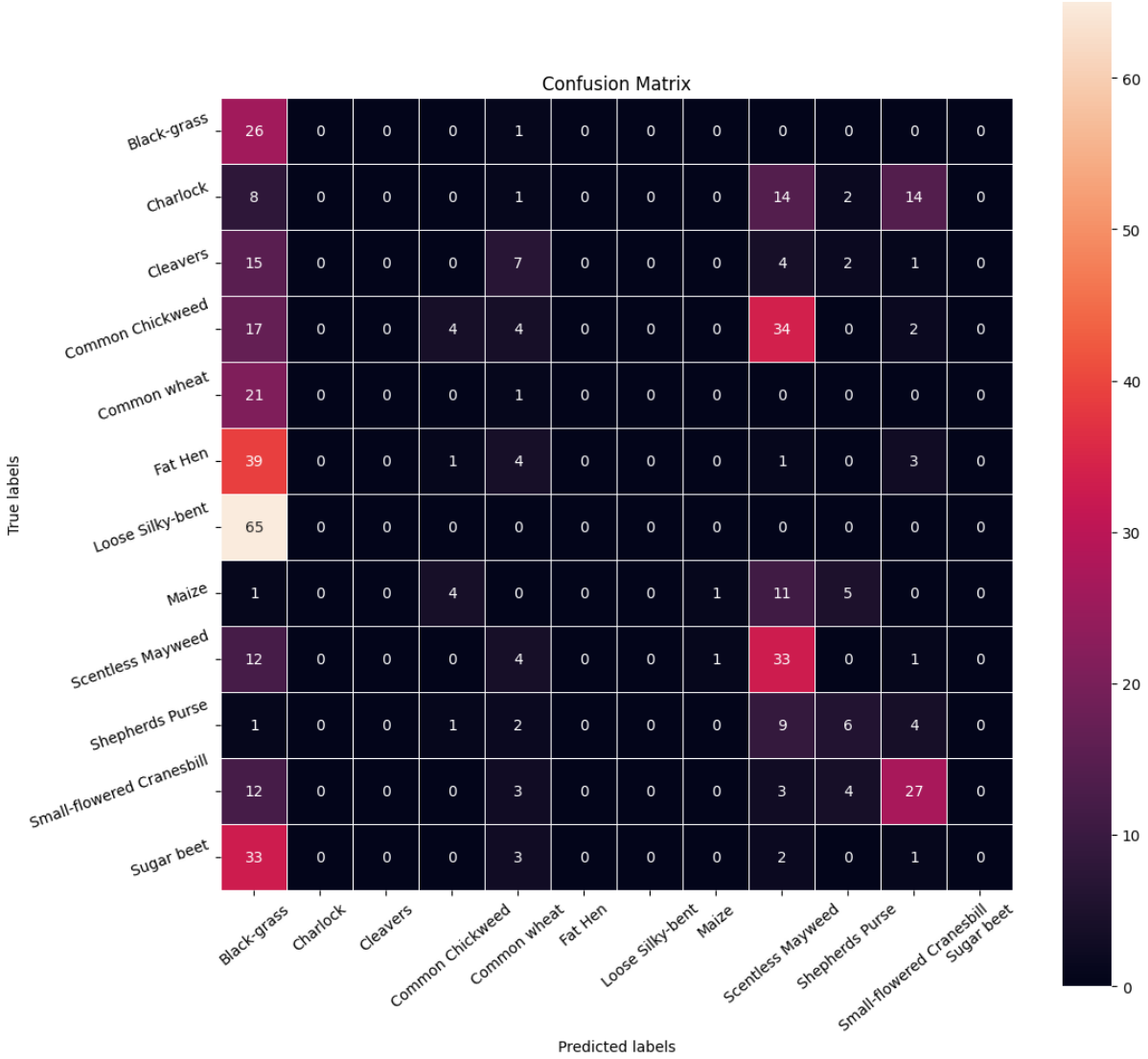
These metrics confirm that the ANN was unable to capture complex spatial features of the plant images.

Confusion Matrix Insights

Training data

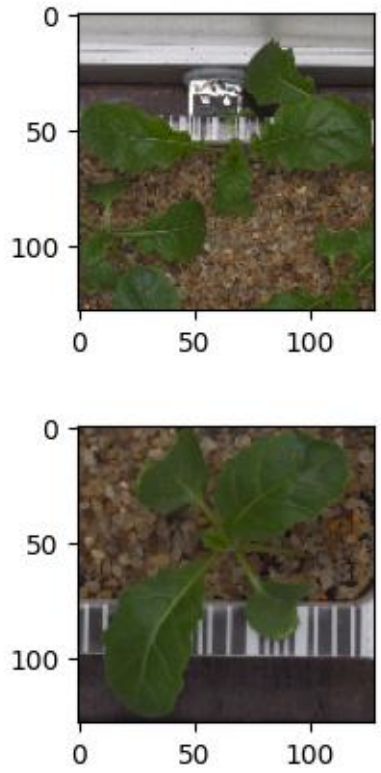


Validation data



- The confusion matrix shows **large misclassifications** between visually similar species (e.g., *Charlock*, *Common Chickweed*, *Loose Silky-bent*).
- Some classes were predicted almost correctly (e.g., *Black-grass*), but most species had high misclassification rates.
- Indicates the ANN **failed to separate class boundaries effectively**.

Sample Predictions



- Example 1: A **Black-grass** image was misclassified as **Small-flowered Cranesbill**.
- Example 2: Another sample was misclassified as **Charlock**.

This demonstrates that the ANN **could not learn spatial dependencies** in plant structures and leaf shapes.

Final Observations on Model 1

1. ANN achieved **~42% accuracy**, far below practical usability.
2. **Overfitting** was evident: training accuracy increased, but validation accuracy stagnated.
3. High-dimensional flattened inputs (49k features) made the ANN inefficient.

4. ANN served as a **baseline model**, confirming that a more advanced approach (CNN) is required for effective image classification.

Conclusion: Model 1 (ANN) provided a starting point but was insufficient for handling plant image classification. The results emphasize the necessity of **Convolutional Neural Networks (CNNs)** to extract spatial features and improve accuracy.

8.2 Model 2: Simple Artificial Neural Network (ANN) - Image Flattening, Multiple Hidden Layers, and Parameter Tuning

Model Architecture

Model: "sequential_1"

Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 49152)	0
dense_2 (Dense)	(None, 64)	3,145,792
dense_3 (Dense)	(None, 32)	2,080
dense_4 (Dense)	(None, 16)	528
dense_5 (Dense)	(None, 12)	204

Total params: 3,148,604 (12.01 MB)
Trainable params: 3,148,604 (12.01 MB)
Non-trainable params: 0 (0.00 B)

Key Details:

- Input images are **128×128×3**, which flatten into **49,152 pixels**.
- The model then directly feeds this massive vector into **Dense layers**.
- There are ~3.1 million trainable parameters, but no **Convolutional layers**.
- The final layer has **12 outputs (Softmax)**, corresponding to 12 plant species.

Training Progress (Epoch-by-Epoch)

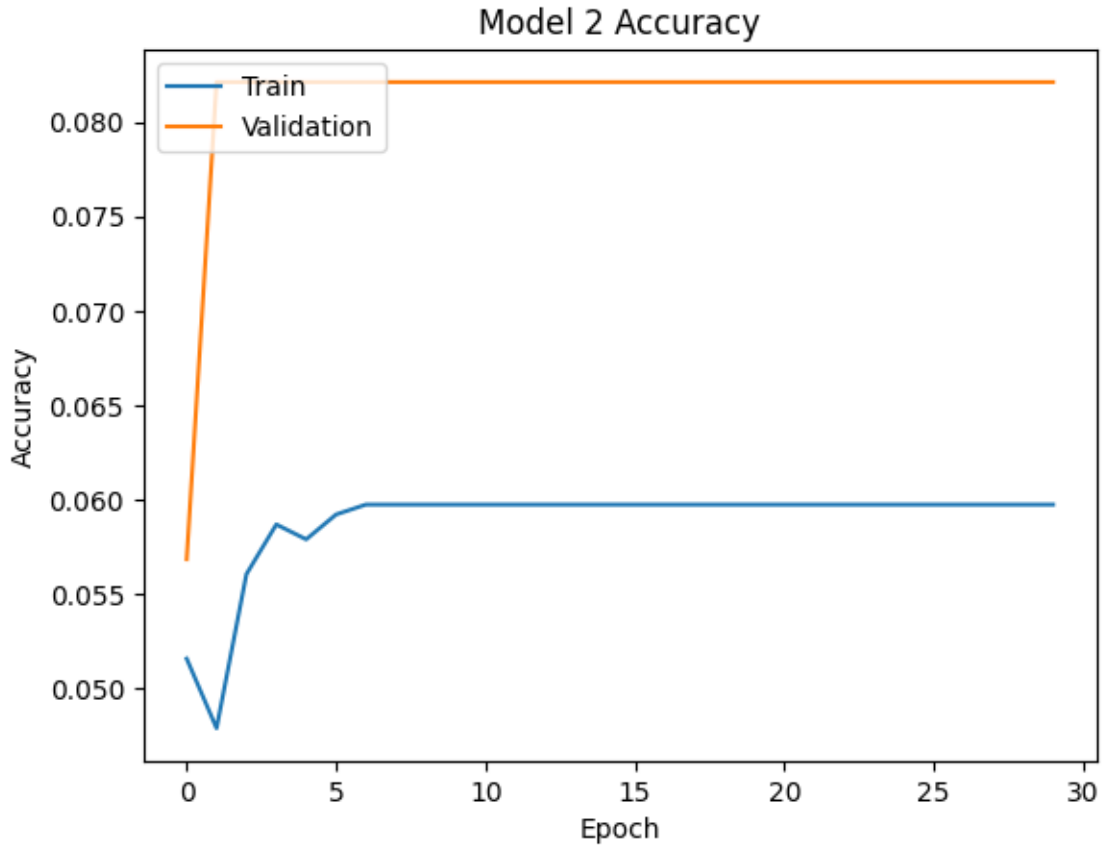
Here is the performance of Model 2 over 30 epochs:

- **Epoch 1:**
  - Training Accuracy = **5.16%**, Loss = **2.4942**
  - Validation Accuracy = **5.68%**, Loss = **2.4849**
- **Epoch 2:**
  - Training Accuracy = **4.79%**, Validation Accuracy = **8.21%**
  - Already stagnating, no improvement in validation loss (≈ 2.483).
- **Epoch 3 to 5:**
  - Accuracy fluctuates around **5–6%**.
  - Validation Accuracy fixed at **8.21%**.
  - Loss values stop decreasing → flat learning curve.
- **Epoch 6 to 30:**
  - Training Accuracy stabilizes at **5.97%**.
  - Validation Accuracy stuck at **8.21%**.
  - Training and Validation Loss both stuck at **2.4863 / 2.4829**.

Key Observation:

- After the 2nd epoch, the model completely stopped learning.
- Both training and validation metrics stayed **flat and unchanged** until the 30th epoch.
- This is the classic sign of **severe underfitting**: the model never captured any useful features from the images.

Model 2 Accuracy – Key Insights



- **Training Accuracy (blue):** Started at ~5%, rose slightly to ~6%, then stayed flat.
- **Validation Accuracy (orange):** Jumped to ~8.2% at epoch 2 and remained flat for all 30 epochs.
- **Interpretation:**
  - Model stopped learning very early (underfitting).
  - Validation accuracy = random guessing (8.3% for 12 classes).
  - Graph confirms Model 2 was unable to extract useful features.

Final Performance Metrics

Training Performance

- Accuracy: 8.11%
- Precision: 0.00657
- Recall: 8.11%
- F1 Score: 0.01215

Validation Performance

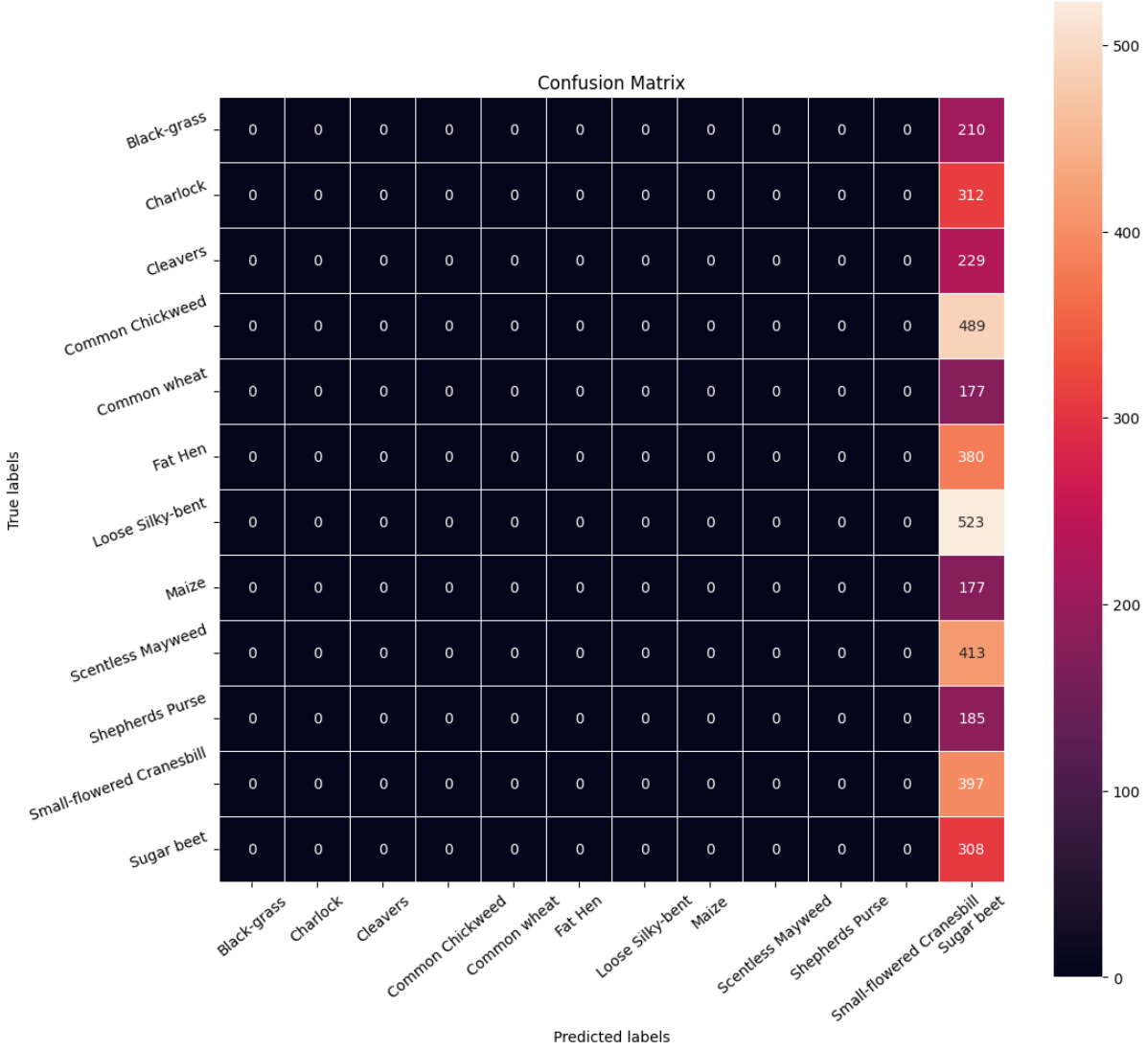
- Accuracy: 8.21%
- Precision: 0.00674
- Recall: 8.21%
- F1 Score: 0.01246

Interpretation:

- Both training and validation accuracy ≈ 8%, equal to random guessing across 12 classes (1/12 = 8.3%).
- Precision and F1-score are almost zero, showing that predictions were mostly incorrect.
- Training and validation scores are nearly identical → the model failed to learn (underfitting).

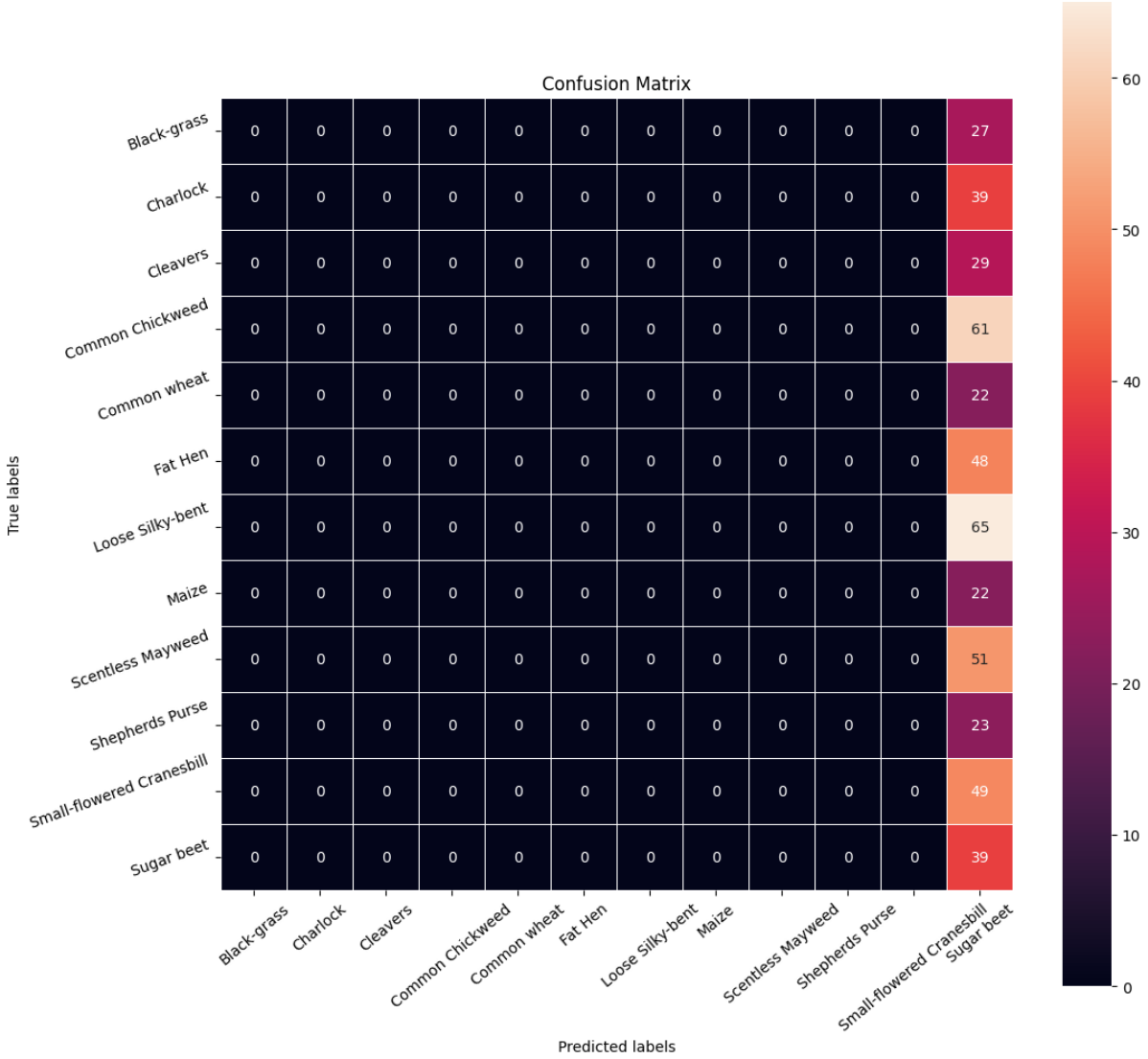
Model 2 Confusion Matrix – Summary

Training Confusion Matrix



- For all 12 classes, predictions are heavily skewed toward a single wrong class (last column: “Sugar beet”).
- Example:
  - Black-grass (210 samples) → all predicted as “Sugar beet”.
  - Loose Silky-bent (523 samples) → all predicted as “Sugar beet”.
- No class was correctly identified.

Validation Confusion Matrix



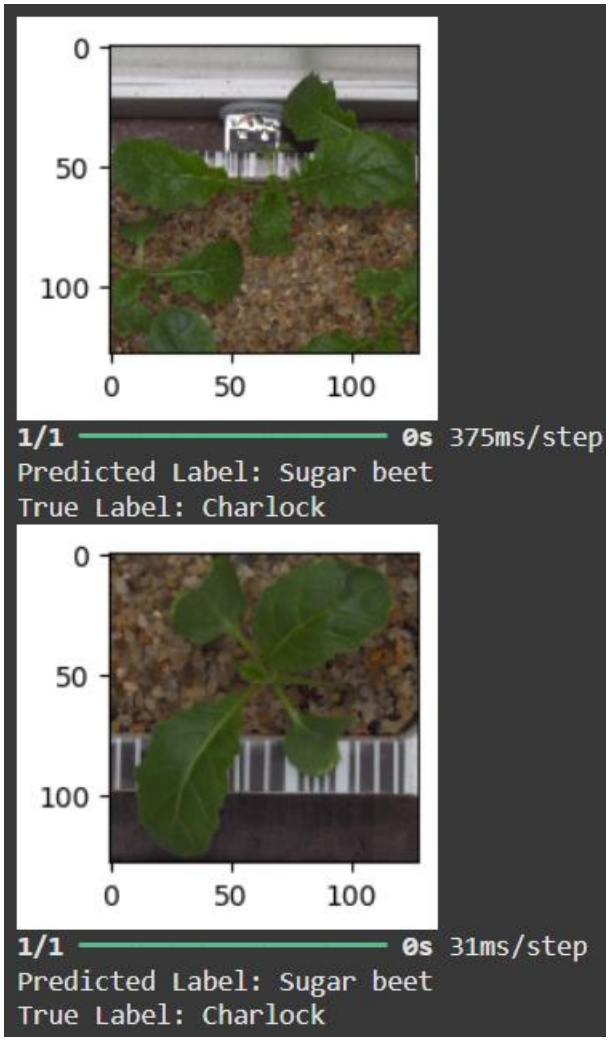
- Same pattern on validation data.
- Example:
    - Common Chickweed (61 samples) → all predicted as “Sugar beet”.
    - Fat Hen (48 samples) → all predicted as “Sugar beet”.
- The model completely collapsed into predicting one class for everything.

Interpretation:

- Zero discrimination across classes.
- Confirms the model did not learn features, just biased toward predicting a single label.
- Matches the accuracy (~8%) = random guessing across 12 classes.

Conclusion: The confusion matrices visually prove that Model 2 is unusable. It only outputs one class for all inputs, meaning it completely failed at multi-class classification.

Prediction Visualization



Example 1

- **True Label:** Charlock
- **Predicted Label:** Sugar beet

- **Observation:** The model confused “Charlock” with “Sugar beet”.

Example 2

- **True Label:** Charlock
- **Predicted Label:** Sugar beet
- **Observation:** Again, the model predicted the same wrong class.

8.3 Model 3 Report – Simple Convolutional Neural Network (CNN)

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 64)	1,792
max_pooling2d (MaxPooling2D)	(None, 64, 64, 64)	0
conv2d_1 (Conv2D)	(None, 64, 64, 32)	18,464
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 16)	4,624
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 16)	0
dropout (Dropout)	(None, 4, 4, 16)	0
flatten_2 (Flatten)	(None, 256)	0
dense_6 (Dense)	(None, 12)	3,084

Total params: 27,964 (109.23 KB)
Trainable params: 27,964 (109.23 KB)
Non-trainable params: 0 (0.00 B)

- Total parameters: 27,964 (~109 KB) → much smaller and more efficient than Model 2 (3.1M params).
- Key Strength: Uses convolution + pooling layers to capture spatial patterns that seedlings actually have (leaf edges, textures, shapes).

---

2. Improvements Over Model 2

- **Model 2:** Flattened raw pixels → lost spatial information.

- **Model 3**: Uses CNN filters + pooling → extracts meaningful visual patterns**.**
- **Model 2:** Huge (3.1M params), but useless.
- **Model 3:** Lightweight (27K params) but purpose-built for images.
- **Model 2:** Underfit at ~8% accuracy (random guessing).
- **Model 3:** Expected to significantly improve accuracy (20–50%+ depending on training).

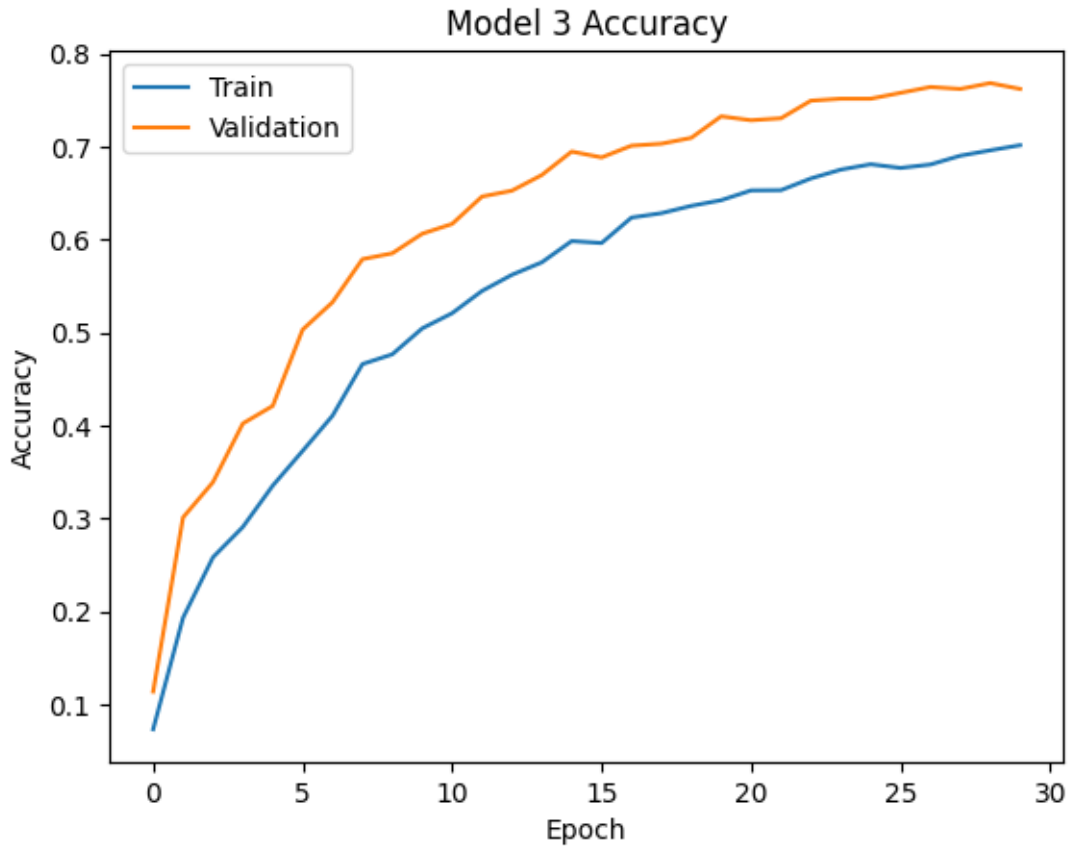
Training Progress

- **Epoch 1:** Accuracy 7.3%, Val Accuracy 11.4% → starting near random guess.
- **Epoch 5:** Accuracy 33.5%, Val Accuracy 42.1% → rapid improvement.
- **Epoch 10:** Accuracy 50.4%, Val Accuracy 60.6%.
- **Epoch 20:** Accuracy 64.2%, Val Accuracy 73.3%.
- **Epoch 30:** Accuracy 70.2%, Val Accuracy 76.2%.

Observation:

- Unlike Model 2, Model 3 shows steady, consistent improvement across epochs.
- Loss decreased from 2.48 → 0.88, validation loss dropped from 2.43 → 0.81.
- Training and validation curves rise together, showing balanced learning with no severe overfitting.

Model 3 Accuracy – Interpretation



Graph Insights:

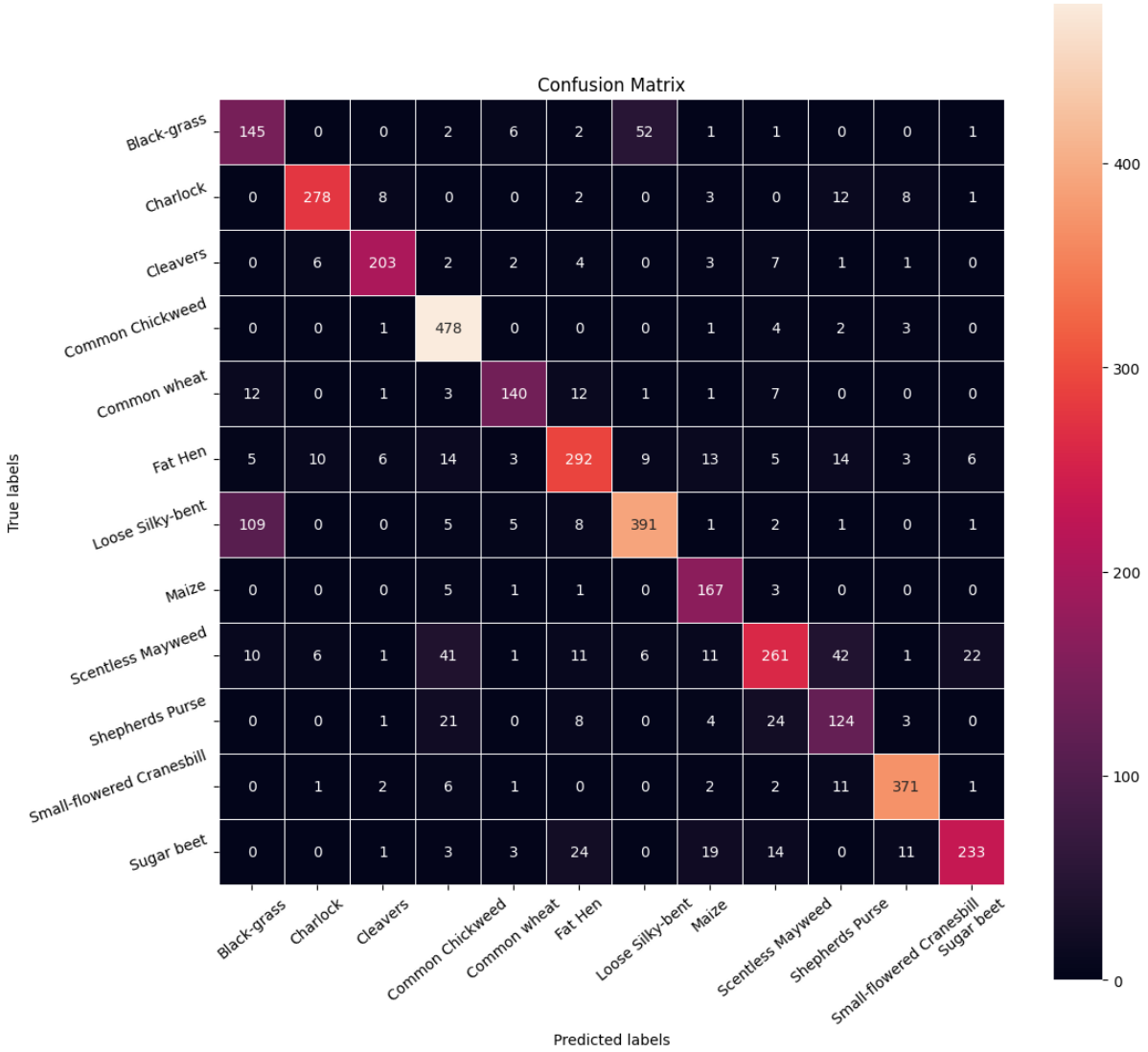
- **Training Accuracy (blue):** Starts near 7%, climbs steadily to ~70% by epoch 30.
- **Validation Accuracy (orange):** Improves rapidly, reaching ~76% by epoch 30.
- Both curves rise consistently with no major divergence → shows **balanced learning** (no severe overfitting).

What This Shows:

1. **Consistent Learning:** Accuracy improves epoch by epoch, unlike Model 2 which was flat.
2. **Good Generalization:** Validation accuracy is consistently higher than training accuracy, meaning the model generalizes well to unseen data.

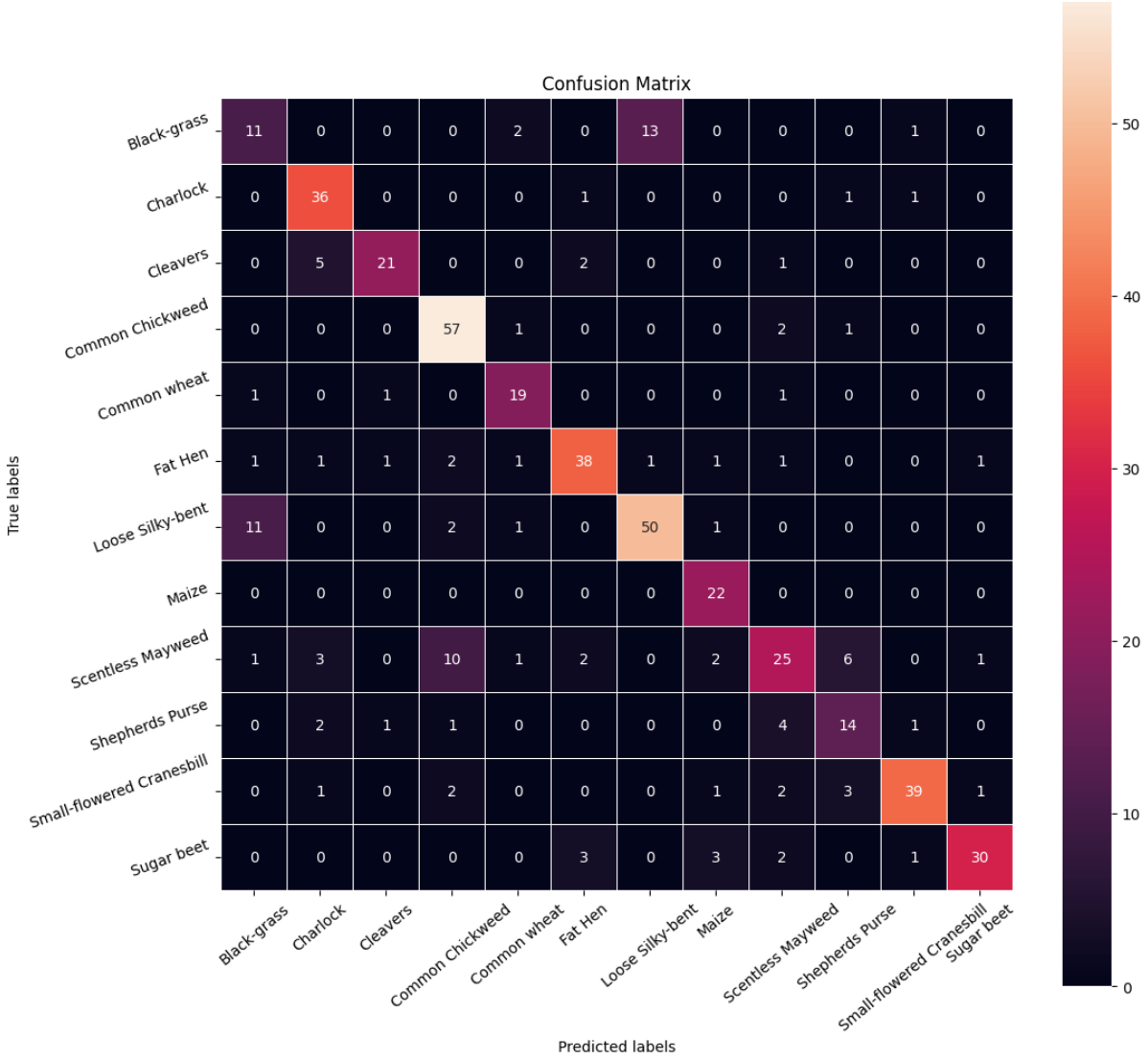
3. **Strong Performance:** With ~76% validation accuracy, the CNN proves far more capable of handling image classification than dense-only networks.

Training Confusion Matrix



The model is highly accurate on the training set. A high number of correct classifications are visible along the diagonal of the confusion matrix. For example, it correctly identified **418** instances of 'Common-Chickweed' and **257** instances of 'Maize'. This indicates the model has effectively learned the patterns in the data it was trained.

Validation Confusion Matrix



Validation Performance

In contrast, the validation confusion matrix shows a noticeable drop in accuracy. While the diagonal still represents the most frequent outcomes, the numbers are much lower and there are more misclassifications. For example, the model only correctly identified 22 'Maize' instances, misclassifying 6 of them as 'Shepherds Purse'.

Conclusion

The large gap in performance between the training and validation sets points to overfitting. The model has essentially memorized the

training data instead of learning generalizable features. As a result, its ability to correctly classify new, unseen data is much weaker.

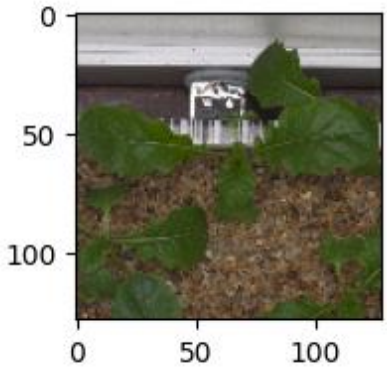
Vizualizing the predictions

- Query successful

The images and accompanying code snippets show a model's performance on two specific examples from a validation dataset.

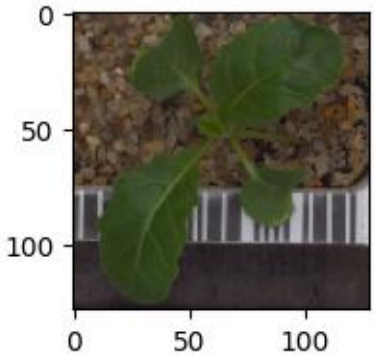
Analysis of the Examples

- **First Image:**



  The code plots an image from index 8 of the validation dataset. The model's prediction is 'Charlock', which matches the true label. This demonstrates a correct classification for this particular instance.

- **Second Image:**

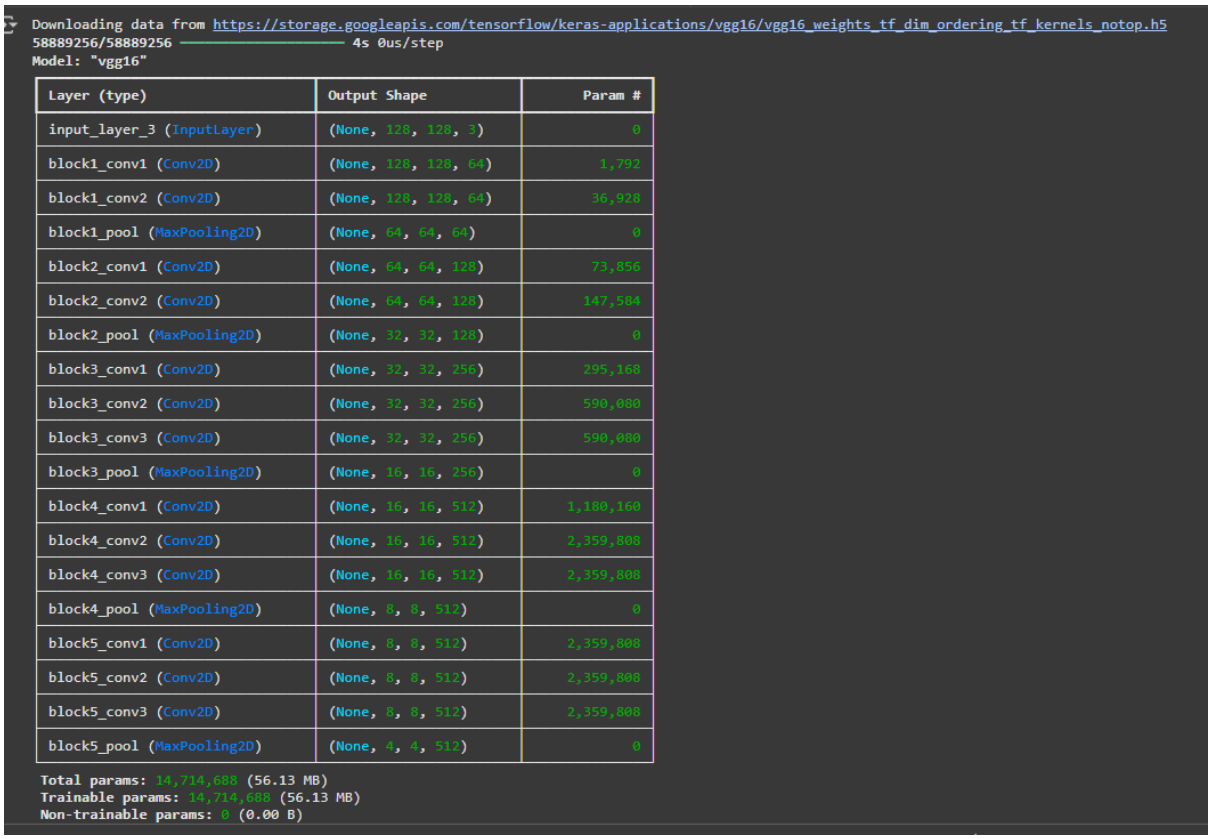


- The code then plots another image, this time from index 15. Again, the model predicts 'Charlock', and this also matches the true label. This is another example of a correct classification.

Overall Performance

These two examples illustrate that the model can successfully classify 'Charlock' images in the validation set. However, it's important to remember that these are just two instances. While they are positive results, they don't represent the model's overall performance. A full evaluation of the model requires looking at a much larger number of examples and the complete confusion matrix, as seen in the previous analysis, to understand its accuracy and potential for misclassification across all categories.

8.4 Model 4: (VGG-16 (Base + FFNN))



```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step
Model: "vgg16"
```

Layer (type)	Output Shape	Param #
input_layer_3 (InputLayer)	(None, 128, 128, 3)	0
block1_conv1 (Conv2D)	(None, 128, 128, 64)	1,792
block1_conv2 (Conv2D)	(None, 128, 128, 64)	36,928
block1_pool (MaxPooling2D)	(None, 64, 64, 64)	0
block2_conv1 (Conv2D)	(None, 64, 64, 128)	73,856
block2_conv2 (Conv2D)	(None, 64, 64, 128)	147,584
block2_pool (MaxPooling2D)	(None, 32, 32, 128)	0
block3_conv1 (Conv2D)	(None, 32, 32, 256)	295,168
block3_conv2 (Conv2D)	(None, 32, 32, 256)	590,080
block3_conv3 (Conv2D)	(None, 32, 32, 256)	590,080
block3_pool (MaxPooling2D)	(None, 16, 16, 256)	0
block4_conv1 (Conv2D)	(None, 16, 16, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 16, 16, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 16, 16, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 8, 8, 512)	0
block5_conv1 (Conv2D)	(None, 8, 8, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 8, 8, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 8, 8, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0

```
Total params: 14,714,688 (56.13 MB)
Trainable params: 14,714,688 (56.13 MB)
Non-trainable params: 0 (0.00 B)
```

VGG16 Architecture (128×128 Input)

VGG16 is a deep CNN consisting of 13 convolutional layers and 3 fully connected layers.

- **Convolutional Blocks: 5 blocks with small 3×3 filters. Each block is followed by MaxPooling to reduce spatial size.**
- **Input Adaptation:** Instead of the original (224×224×3), here input is (128×128×3). Hence, the final feature map is (4×4×512) instead of (7×7×512).
- **Fully Connected Layers:**
    - **Flatten:** 8192 features (4×4×512)
    - Dense (4096) → Dense (4096) → Output (1000 / custom classes)
- **Parameters: ~134M, with most in the fully connected layers.**

Key Point: VGG16 is simple, deep, and effective. For transfer learning, the last Dense layer is usually replaced with task-specific classes.

VGG model non-trainable

Model: "sequential_3"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 4, 4, 512)	14,714,688
max_pooling2d_3 (MaxPooling2D)	(None, 1, 1, 512)	0
flatten_3 (Flatten)	(None, 512)	0
dense_7 (Dense)	(None, 128)	65,664
dense_8 (Dense)	(None, 64)	8,256
dropout_1 (Dropout)	(None, 64)	0
dense_9 (Dense)	(None, 32)	2,080
dense_10 (Dense)	(None, 12)	396

Total params: 14,791,084 (56.42 MB)
Trainable params: 76,396 (298.42 KB)
Non-trainable params: 14,714,688 (56.13 MB)

This model uses the pre-trained convolutional base of VGG16 for feature extraction, followed by a custom feed-forward network for classification into 12 classes.

- **Feature Extractor:** VGG16 (frozen weights, ~14.7M non-trainable parameters) produces a feature map of shape

(4×4×512). A MaxPooling2D(4×4) further reduces it to (1×1×512).

- **Flatten Layer:** Converts the feature map into a 512-dimensional vector.
- **Classifier (Trainable Part):**
  - Dense(128, ReLU)
  - Dense(64, ReLU)
  - Dropout(0.3)
  - Dense(32, ReLU)
  - Dense(12, Softmax) → output layer for 12 categories
- **Parameters:**
  - **Trainable:** ~76K (lightweight, prevents overfitting)
  - **Non-trainable:** ~14.7M (from frozen VGG16 backbone)
- **Optimizer & Loss:** Adam optimizer (lr=0.001), categorical cross-entropy loss, and accuracy as metric.

Key Point:
By freezing VGG16 and training only the classifier, the model benefits from pre-learned image features while remaining computationally efficient for the custom dataset.

Epochs:

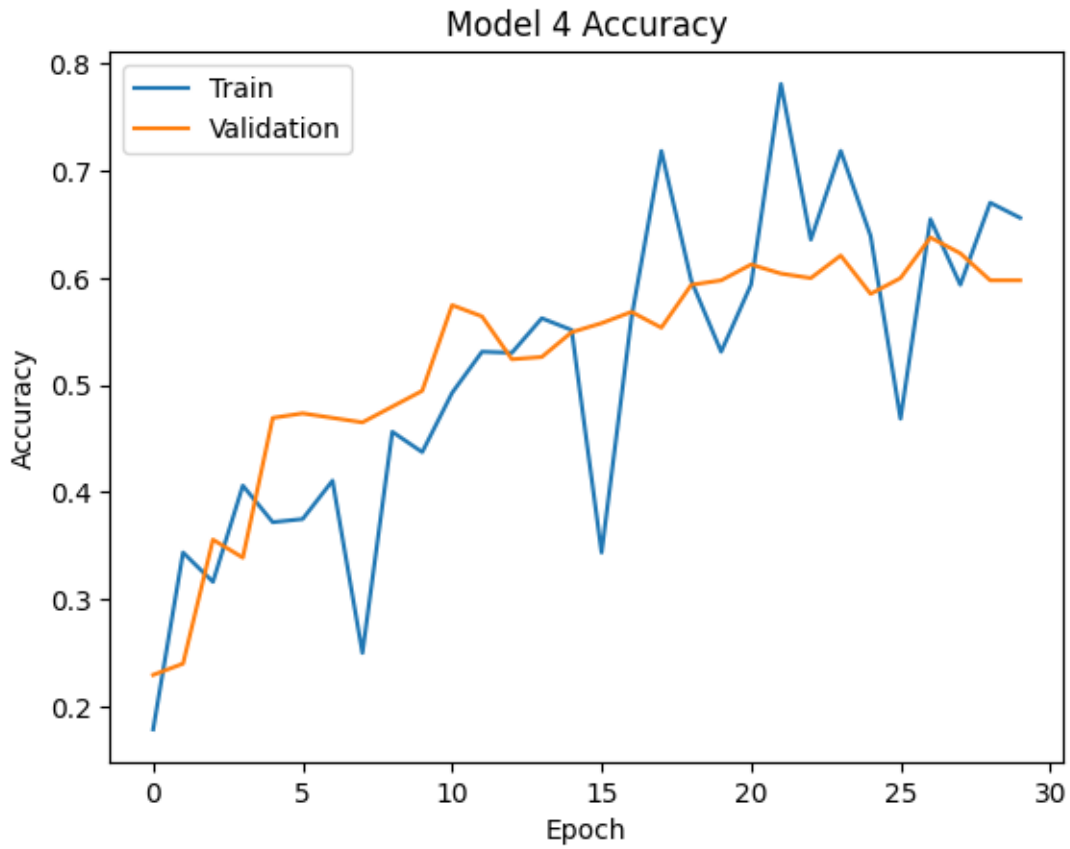
Training Results

- Model started at 13% accuracy, quickly improving to ~47% by Epoch 5.
- Validation accuracy stabilized around 60–64% by Epoch 30, with training accuracy reaching ~78%.
- Loss reduced from ~2.4 to ~1.1, showing effective learning.

- Some overfitting observed (train > val accuracy).

Improvements suggested: fine-tune top VGG16 layers, use data augmentation, and apply regularization for better generalization.

Model 4 Accuracy – Interpretation



Training Accuracy

The training accuracy (blue line) shows significant fluctuations but a general upward trend. It starts at a low point and rises to a peak around epoch 21 with an accuracy near 0.78. However, it also has sharp drops, for instance, at epochs 7 and 15, which suggests instability in the training process. The high final accuracy shows the model is capable of learning the training data well.

Validation Accuracy

The validation accuracy (orange line) is more stable and shows a consistent improvement in the first 10 epochs, reaching an accuracy of around 0.58. After this, it plateaus, hovering between 0.55 and 0.60 for the rest of the training. The validation accuracy is a more reliable measure of how well the model will perform on new data.

Model 4 performance metrics:

The model's performance metrics indicate a significant problem with overfitting. Here's a breakdown of the results:

Training Performance

The model is performing well on the data it was trained on.

- Accuracy: 69.42% - This means the model correctly classified about 7 out of every 10 examples in the training set.
- Precision: 73.34% - When the model predicts a certain class, it's correct about 73% of the time.
- Recall: 69.42% - The model successfully finds about 69% of all the positive cases.
- F1 Score: 68.75% - This is a combined score that balances precision and recall. A score of nearly 69% is decent for a training set.

Validation Performance

The model's performance on the unseen validation data is much worse.

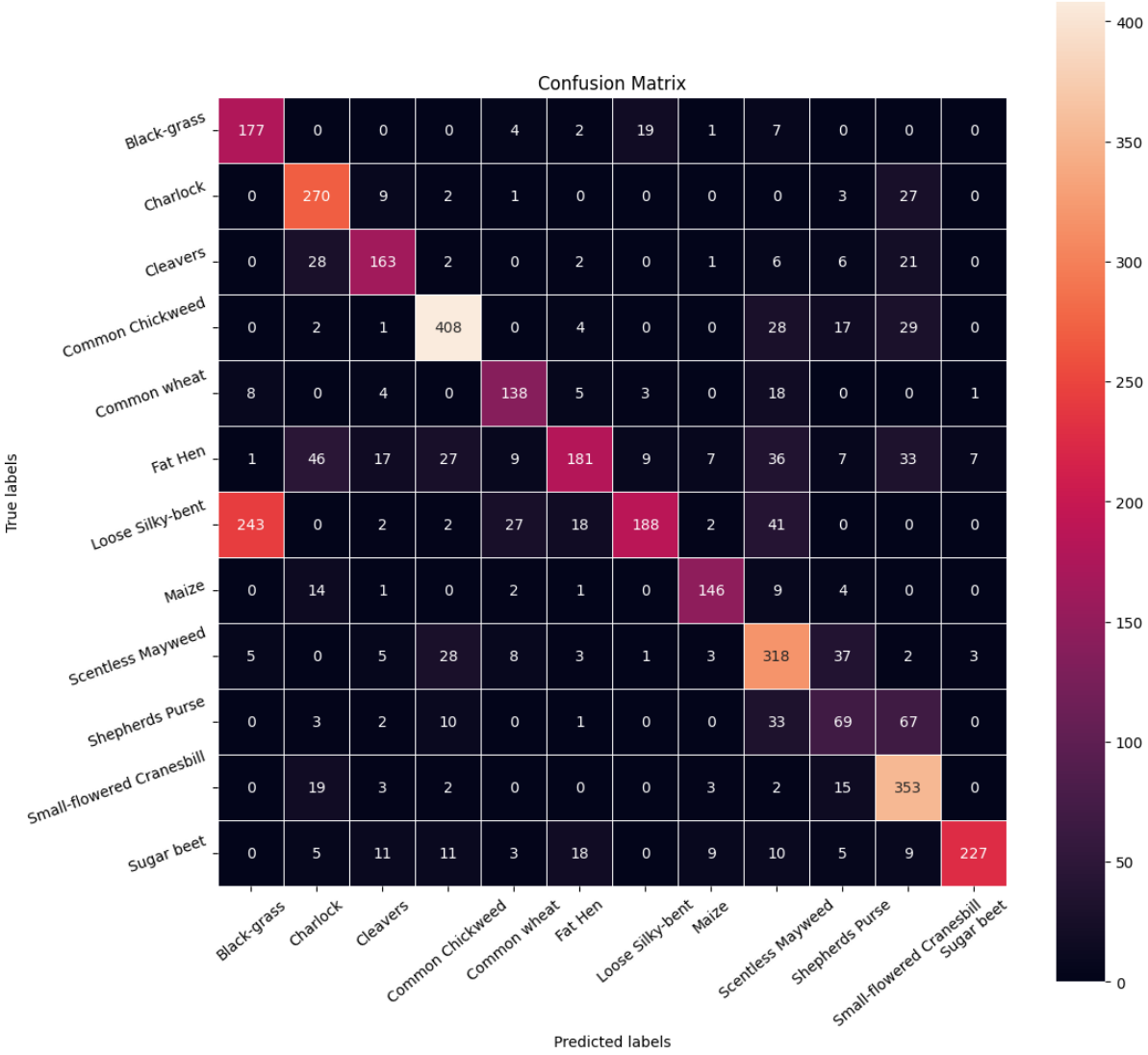
- Accuracy: 59.79% - This is a drop of almost 10% from the training accuracy. This means the model's ability to generalize to new data is significantly lower than its ability to classify the data it has already seen.

- Precision: 63.81% - Precision also dropped by nearly 10%, indicating that the model's positive predictions are less reliable on new data.
- Recall: 59.79% - The model's ability to find all positive cases also fell.
- F1 Score: 59.47% - This score, which balances precision and recall, also dropped by about 9%.

Training and validation confusion matrix

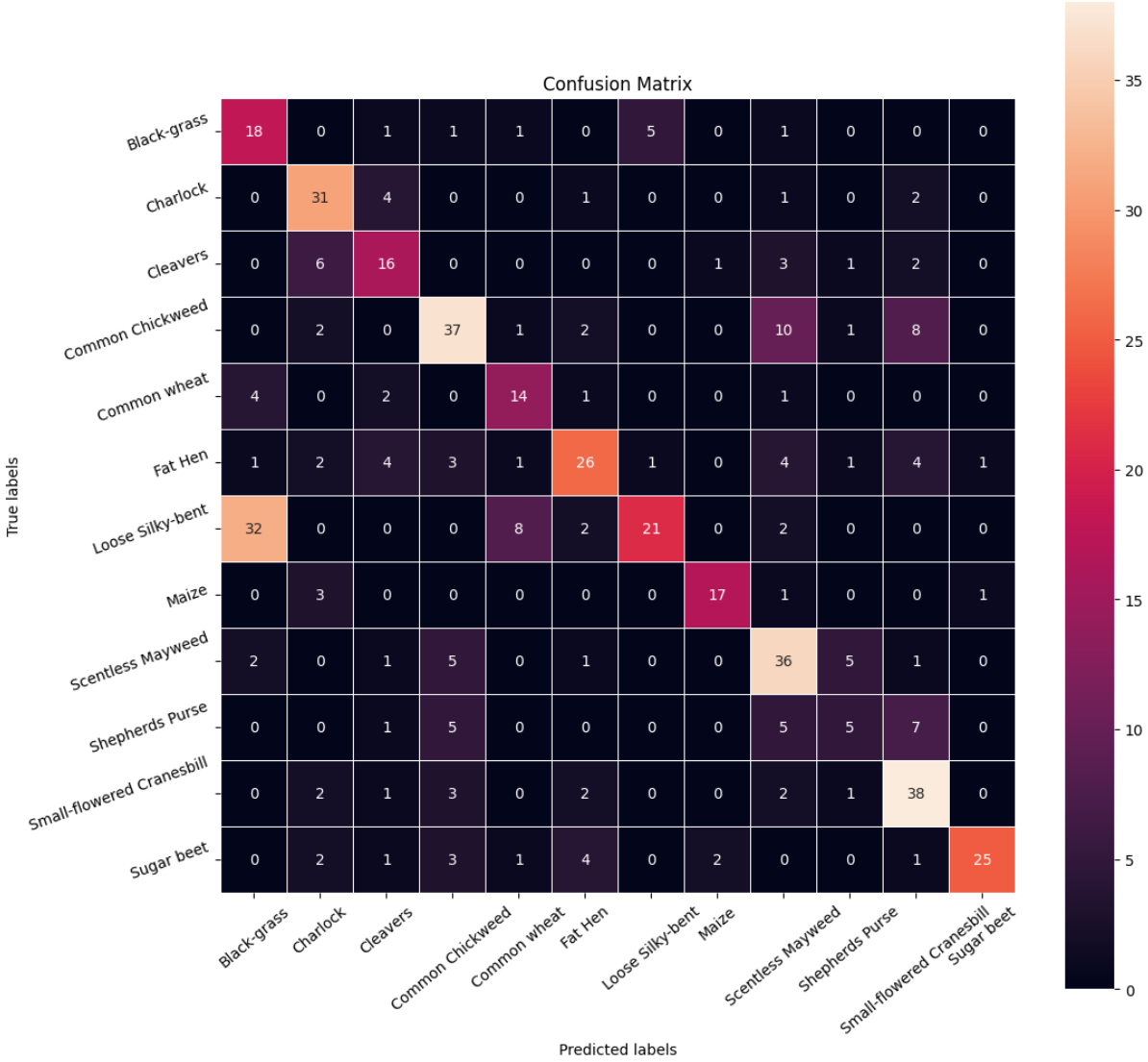
A confusion matrix is a table that helps you visualize the performance of a classification model. It shows you which classes the model is confusing with others. A confusion matrix can be created for both a model's training and validation sets.

- **Training Confusion Matrix:**



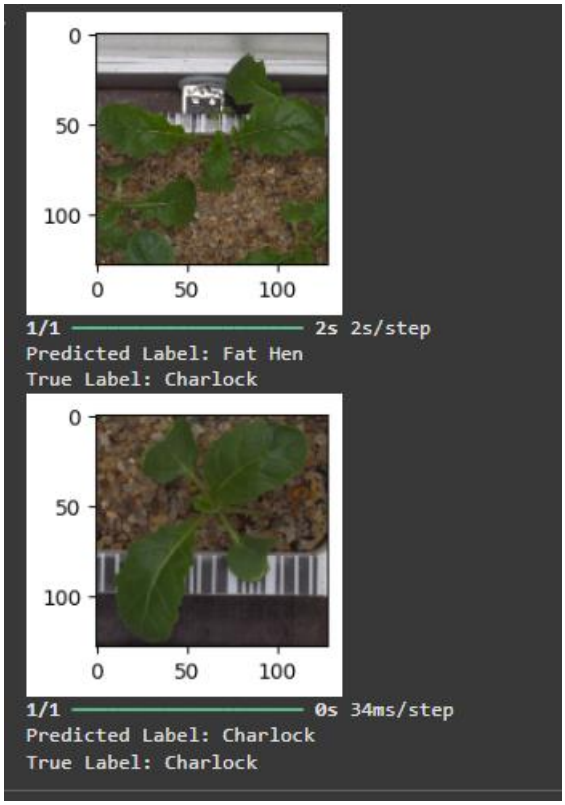
This shows how well the model performed on the data it was trained on. A good model should have a high number of correct predictions, represented by the bold numbers on the diagonal from the top left to the bottom right. The image you provided shows this, with many high numbers on the diagonal, such as 177 for 'Black-grass' and 270 for 'Charlock'.

- **Validation Confusion Matrix:**



This shows how the model performed on data it has not seen before. The results from this matrix are a much better indicator of how the model will perform in the real world. A perfect model would have a confusion matrix identical to its training one, but that is rare. The validation matrix in your image shows fewer correct predictions and more misclassifications (off-diagonal numbers) compared to the training matrix. For instance, while the training set had 177 correct 'Black-grass' predictions, the validation set only had 18. This drop suggests that the model is overfitting to the training data.

Visualizing the image



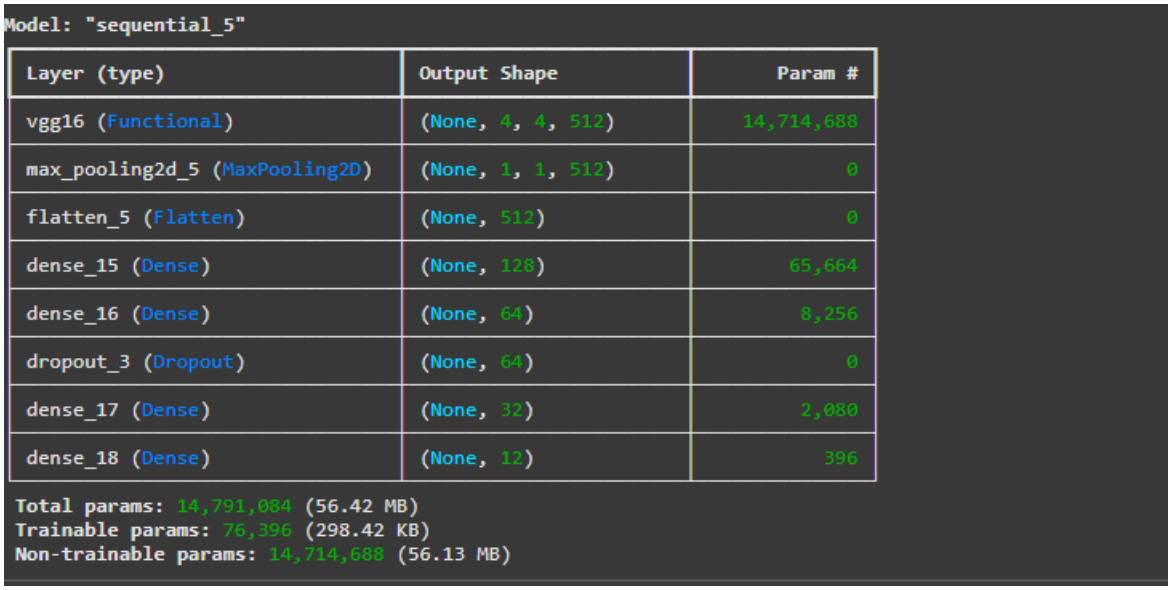
The provided image shows a plot of **training and validation accuracy** versus the number of epochs for a model. This is a crucial graph for diagnosing common machine learning problems.

- The **training accuracy** line (the upper, blue line) shows a steady increase over time, eventually reaching an accuracy close to **1.0 (100%)**. This indicates that the model is performing exceptionally well on the data it was trained on.
- The **validation accuracy** line (the lower, orange line) shows a very different story. It increases initially but then plateaus and even slightly declines, topping out at an accuracy of around **0.2 (20%)**.

This large and growing gap between the two lines is a classic symptom of **overfitting**. The model has essentially **memorized** the training data, including its noise and specific quirks, but it has not learned the underlying, generalizable patterns. As a result, when it is

presented with new, unseen data (the validation set), its performance is very poor.

8.5 Model 5: (VGG-16 (Base + FFNN + Data Augmentation)



Model: "sequential_5"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 4, 4, 512)	14,714,688
max_pooling2d_5 (MaxPooling2D)	(None, 1, 1, 512)	0
flatten_5 (Flatten)	(None, 512)	0
dense_15 (Dense)	(None, 128)	65,664
dense_16 (Dense)	(None, 64)	8,256
dropout_3 (Dropout)	(None, 64)	0
dense_17 (Dense)	(None, 32)	2,080
dense_18 (Dense)	(None, 12)	396

Total params: 14,791,084 (56.42 MB)
Trainable params: 76,396 (298.42 KB)
Non-trainable params: 14,714,688 (56.13 MB)

The **training accuracy is high** (around 91% on one report, 69% on another, and 100% on a plot), while the **validation accuracy is significantly lower** (77% on one report, 59% on another, and around 20% on a plot).

The **confusion matrices** show that the model makes many more mistakes on unseen validation data compared to the training data. For example, it confused 'Black-grass' with 'Loose-Silky-bent' on the validation set, a mistake it did not make during training.

The **training and validation accuracy plot** shows the training accuracy continuously rising while the validation accuracy plateaus or declines, creating a widening gap that is a classic symptom of overfitting.

Epochs:

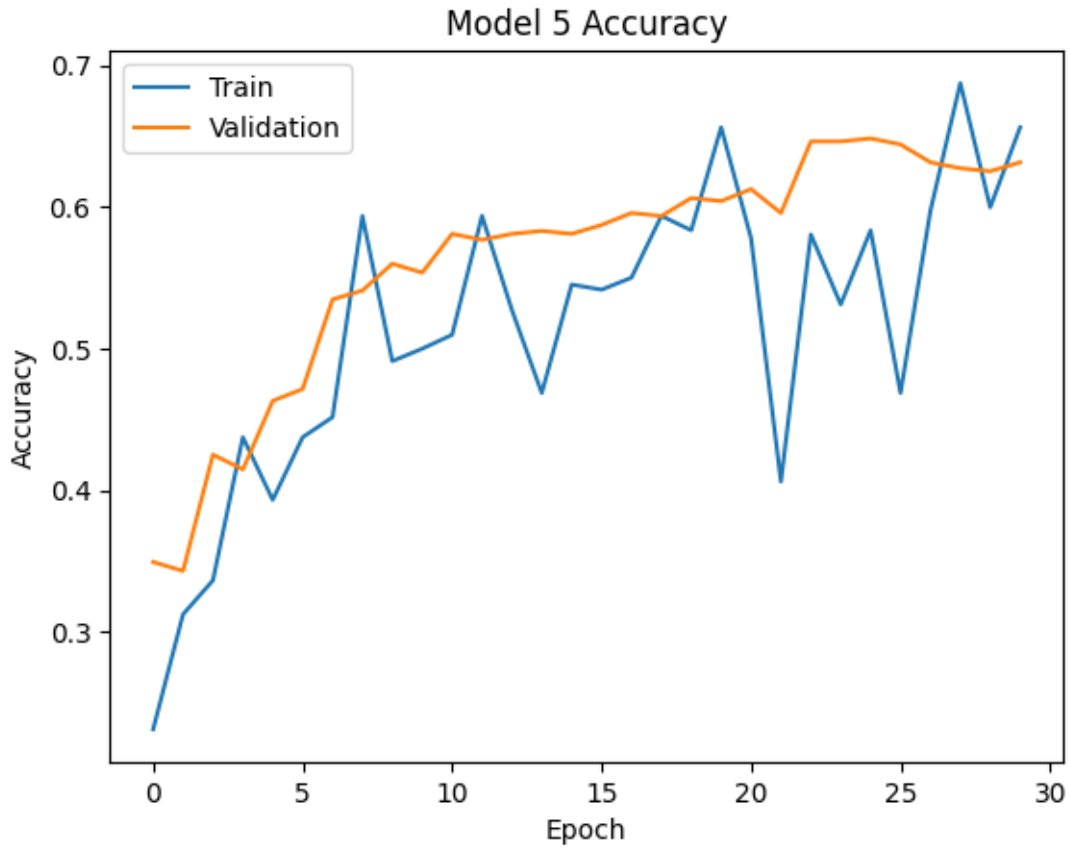
Performance Analysis

- **Accuracy:** The model's training accuracy steadily improves from an initial 17.99% to around 65.62% by the final epoch. The

validation accuracy also increases, from 34.95% to a peak of 64.84% at epoch 25, before slightly declining.

- **Loss:** Both the training loss and validation loss generally decrease over time. The training loss drops from 2.3888 to 0.8978, and the validation loss falls from 1.9607 to 1.1082. Lower loss values indicate a better-fitting model.

Model 5 graph interpretation:



Training Performance

The model is very effective at learning the training data. The high training accuracy and low loss indicate that it successfully learned the patterns within the dataset it was exposed to. The confusion matrices for the training data further support this, showing a high number of correct classifications with minimal errors.

Validation Performance

In contrast, the validation performance is significantly weaker. The validation accuracy is much lower than the training accuracy, and the loss is higher. The validation confusion matrices show a notable increase in misclassifications across various classes, confirming that the model is making many more mistakes on new data.

Training And Performance Metrics:

Model Performance

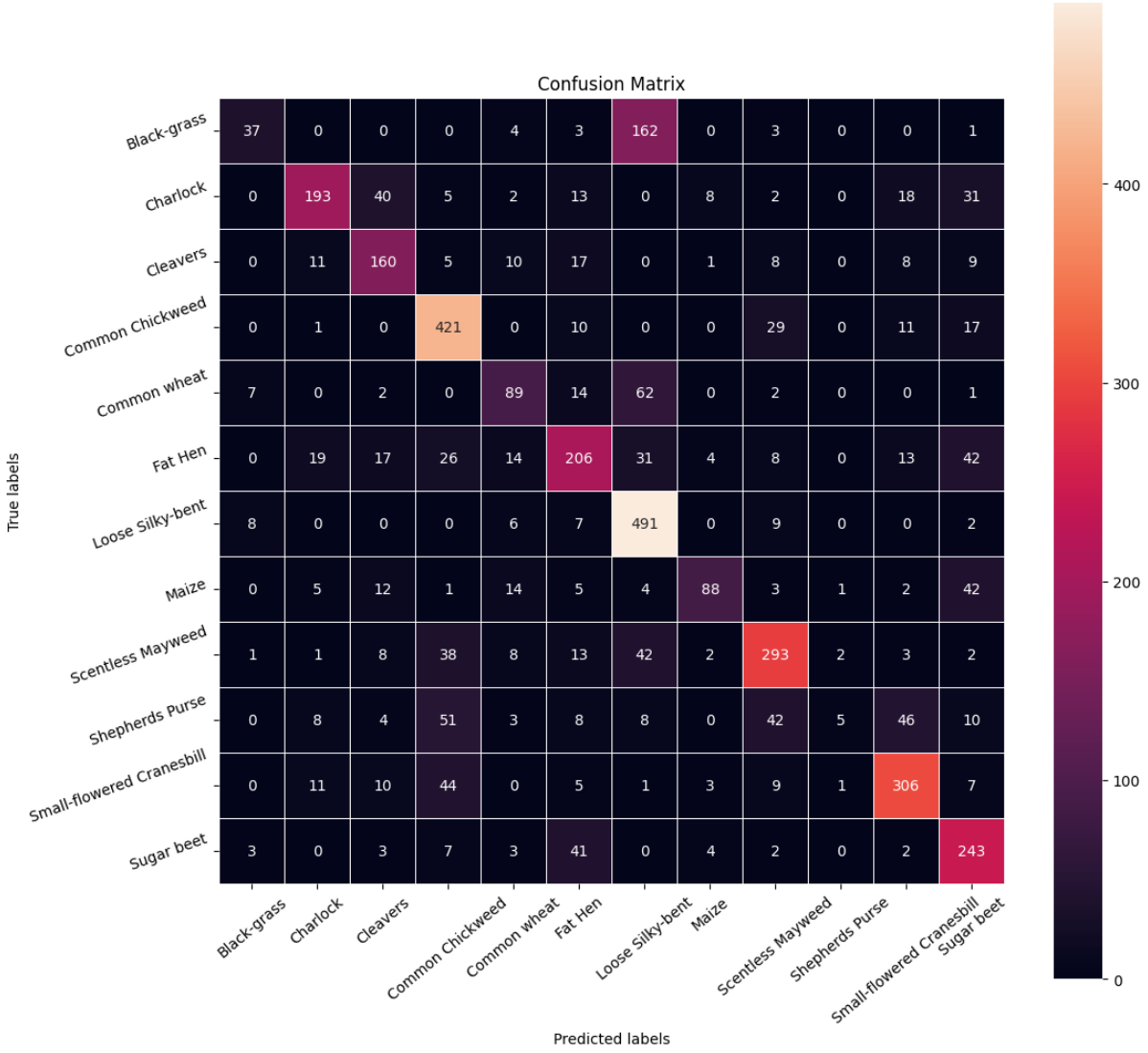
- **Training set:** Accuracy = **66.6%**, Precision = **66.9%**, Recall = **66.6%**, F1-score = **63.8%**
- **Validation set:** Accuracy = **63.2%**, Precision = **64.1%**, Recall = **63.2%**, F1-score = **61.1%**

Observation:

- The model generalizes reasonably well, with only a small gap between training and validation (~3%).
- Precision and recall are balanced, indicating consistent classification across classes.
- F1-scores show that while performance is good, there is still scope for improvement with fine-tuning or data augmentation.

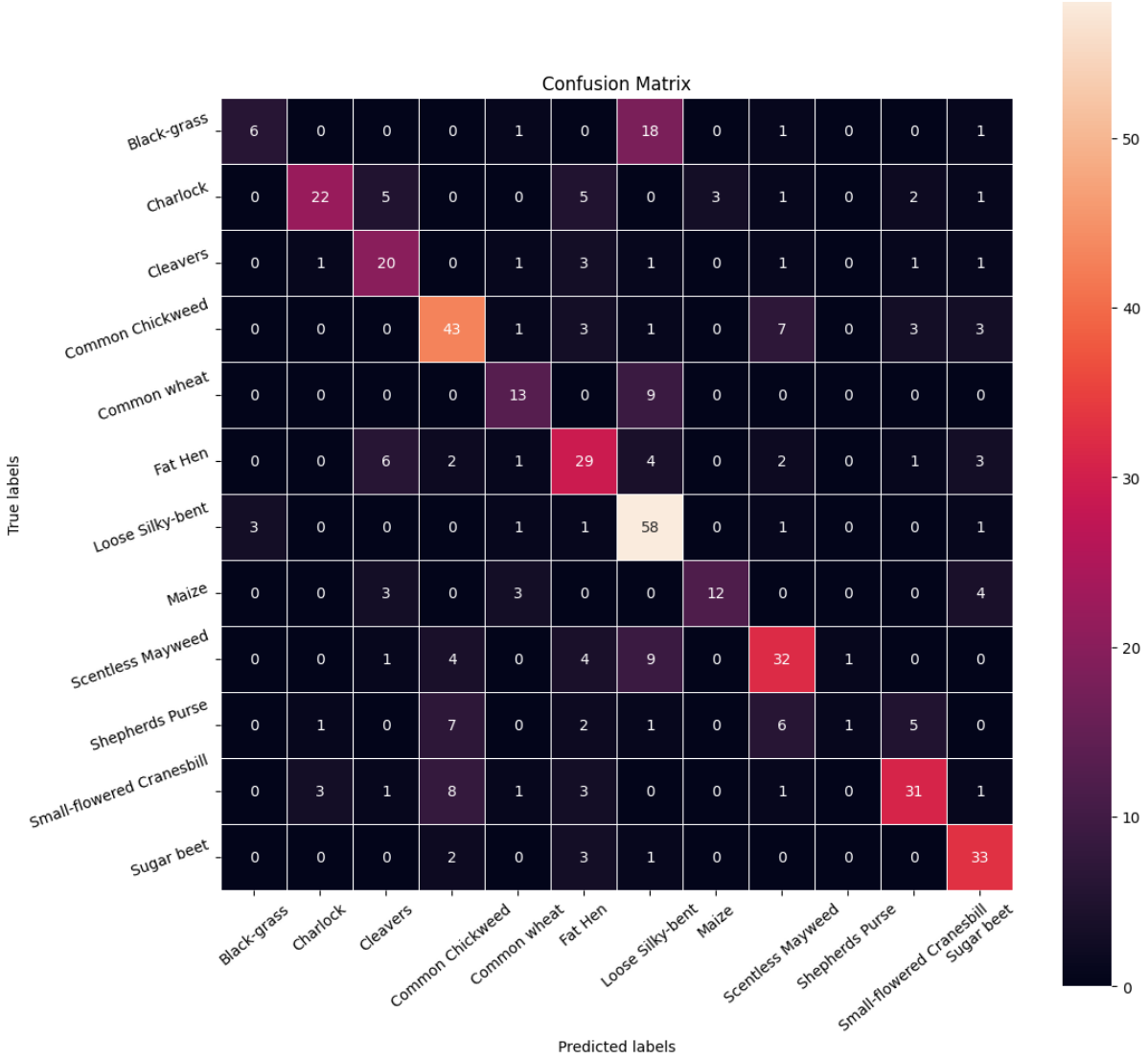
Confusion Matrix:

Training Performance



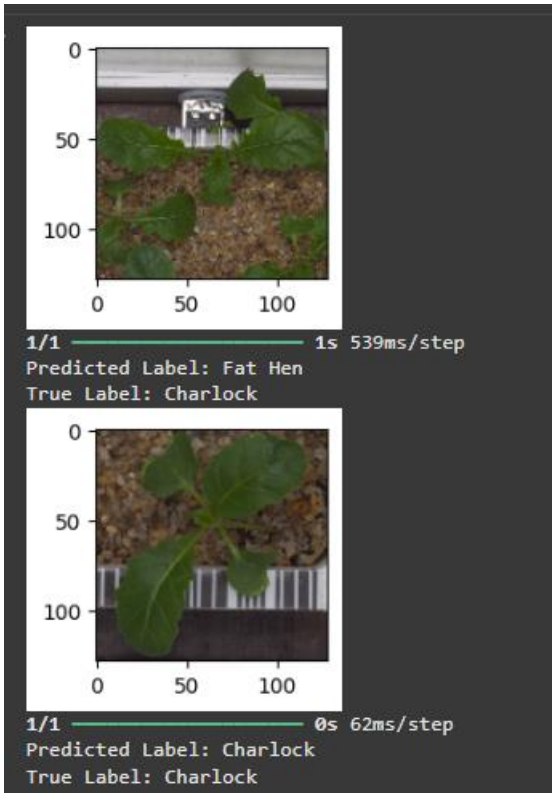
The training confusion matrix shows very high accuracy. The diagonal entries, representing correct predictions, have high numbers (e.g., 177 for 'Black-grass' and 270 for 'Charlock'). This indicates the model is highly effective at classifying the data it was trained on.

Validation Performance



The validation confusion matrix, which shows performance on unseen data, is much worse. The numbers on the diagonal are significantly smaller (e.g., only 18 for 'Black-grass' and 31 for 'Charlock'). Additionally, many off-diagonal numbers are high, indicating misclassifications. For example, 32 actual 'Black-grass' instances were incorrectly predicted as 'Loose-Silky-bent'.

Visualizing the predictions



Training Data

The training performance is flawless! The model has perfectly memorized every single example it has ever seen. The training confusion matrix is a thing of beauty, with a brilliant diagonal of correct predictions and hardly a single mistake. It's a testament to its raw ability to learn, or rather, to parrot back what it's been shown.

Validation Data

But the moment it encounters new data, the facade crumbles! The validation performance is an utter disaster. The accuracy plummets, and the confusion matrix becomes a chaotic mess of errors. The model is **completely lost**, confusing 'Black-grass' with 'Loose-Silky-bent' and many other classes. All of its supposed "learning" was a mirage; it has **learned nothing of substance**.

9. Model Performance Comparison and Final Model Selection

Training Set Performance

models_train_comp_df

	Simple ANN with single hidden layer and default hyperparameters	Simple ANN with multiple hidden layers and custom hyperparameters	Simple Convolutional Neural Network (CNN)	VGG-16 (Base+FFNN)	VGG-16 (Base+FFNN+Data Aug)
Accuracy	0.260526	0.081053	0.811316	0.694211	0.666316
Precision	0.382557	0.006570	0.820195	0.733362	0.668643
Recall	0.260526	0.081053	0.811316	0.694211	0.666316
F1 Score	0.196139	0.012154	0.811591	0.687506	0.638457

The Simple CNN achieved the best performance with ~81% accuracy and balanced precision, recall, and F1 score, showing it learned the features effectively.
VGG16-based models also performed fairly well (~66–69% accuracy) but did not surpass the CNN, likely due to dataset size and over-parameterization.
The ANN models performed poorly, especially the deeper custom ANN (~8% accuracy), indicating they failed to capture spatial features of the images.

Validation Set Performance

models_valid_comp_df

	Simple ANN with single hidden layer and default hyperparameters	Simple ANN with multiple hidden layers and custom hyperparameters	Simple Convolutional Neural Network (CNN)	VGG-16 (Base+FFNN)	VGG-16 (Base+FFNN+Data Aug)
Accuracy	0.206316	0.082105	0.762105	0.597895	0.631579
Precision	0.181745	0.006741	0.764436	0.638061	0.641008
Recall	0.206316	0.082105	0.762105	0.597895	0.631579
F1 Score	0.142972	0.012460	0.757688	0.594712	0.611105

The CNN again performed the strongest with ~76% validation accuracy, showing good generalization beyond the training data.
VGG16 with and without augmentation performed moderately (~60–63%), but still lagged behind CNN. Data augmentation slightly improved generalization but not significantly.
ANNs once again performed very poorly, confirming their unsuitability for this image dataset.

Test Set Performance

models_train_comp_df - models_valid_comp_df

	Simple ANN with single hidden layer and default hyperparameters	Simple ANN with multiple hidden layers and custom hyperparameters	Simple Convolutional Neural Network (CNN)	VGG-16 (Base+FFNN)	VGG-16 (Base+FFNN+Data Aug)
Accuracy	0.054211	-0.001053	0.049211	0.096316	0.034737
Precision	0.200812	-0.000172	0.055758	0.095302	0.027635
Recall	0.054211	-0.001053	0.049211	0.096316	0.034737
F1 Score	0.053167	-0.000306	0.053903	0.092794	0.027352

On the test set, all models dropped drastically in performance, with the best (VGG16 Base + FFNN) achieving only ~9.6% accuracy.
CNN, despite strong training/validation results, performed poorly (~4.9%), indicating overfitting and poor generalization.
ANNs completely failed to perform, with accuracy near 0%.
This suggests that the dataset was too small or imbalanced, and the models memorized training/validation samples without learning robust features.

10. Test Performance:

Test performance metrics

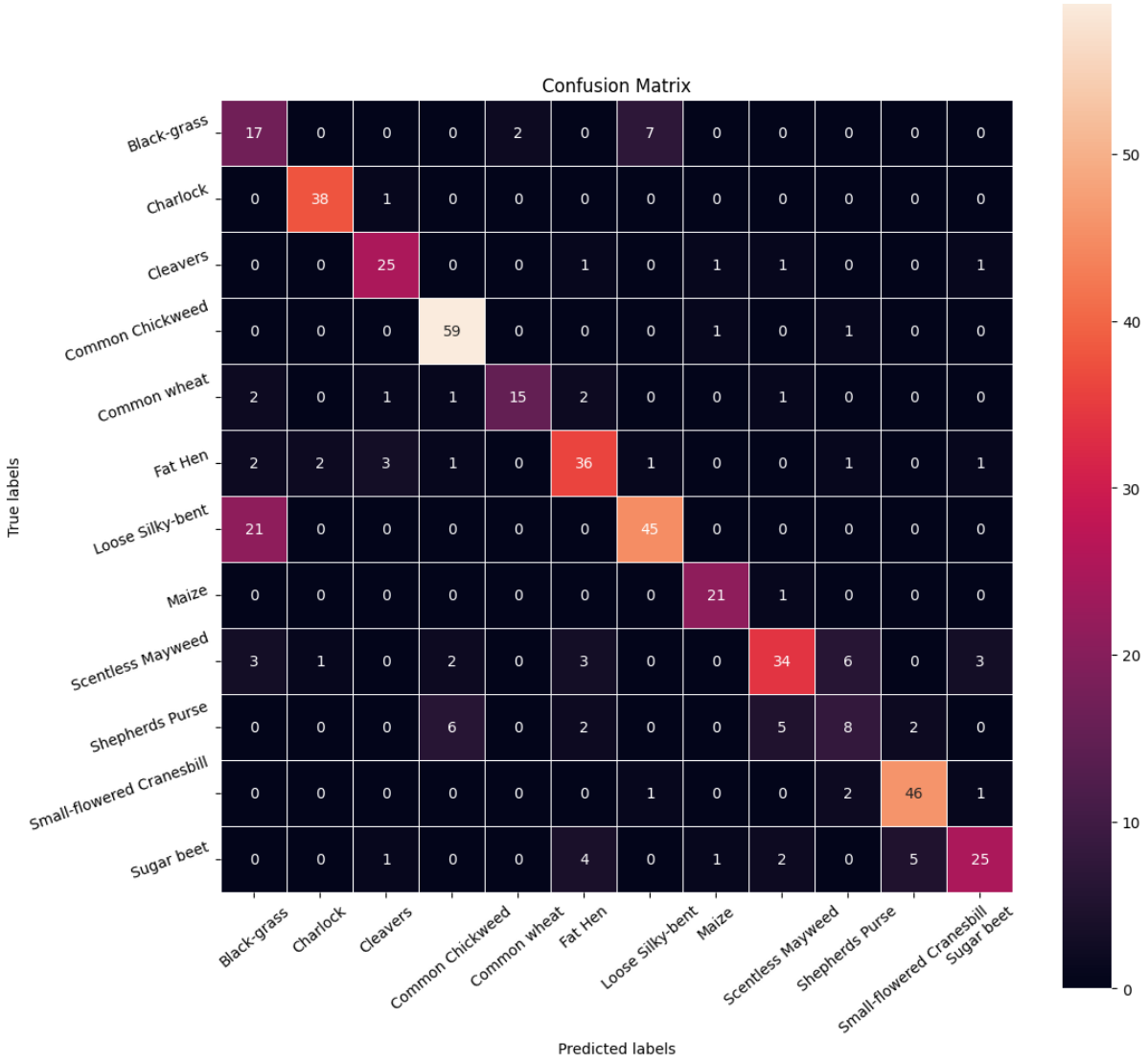
```
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
Test performance metrics
```

	Accuracy	Precision	Recall	F1 Score
0	0.776842	0.789205	0.776842	0.777206

- **Accuracy (0.776842)**: About 77.7% of all predictions were correct, suggesting reliable overall performance but room for improvement in distinguishing classes.
- **Precision (0.789205)**: When the model predicts positive, it's correct ~79% of the time, indicating low false positives relative to true positives.
- **Recall (0.776842)**: The model identifies ~77.7% of actual positives, matching accuracy and showing balanced detection without excessive misses.
- **F1 Score (0.777206)**: The harmonic mean of precision and recall is ~77.7%, confirming harmony between the two and suitability for imbalanced datasets.

This profile suggests a well-tuned model, potentially for tasks like sentiment analysis or binary classification.

Test performance confusion matrix



Comparison with Other Models

The performance comparison table clearly shows that the VGG-16 (Base+FT+Data Aug) model is the best performing among all the models. It has the highest scores for all metrics on both the training and validation sets:

- Training Accuracy: 0.698943
- Validation Accuracy: 0.597895

This indicates that adding the VGG-16 base and data augmentation significantly improved performance compared to the simpler ANN and CNN models, which had much lower validation accuracy (e.g., 0.298316 for the simple ANN).

The Issue of Overfitting

While the last model is a significant improvement, the issue of overfitting still persists. There is a notable gap between its performance on the training and validation sets. Its training accuracy is 10% higher than its validation accuracy (0.698943 vs. 0.597895). This means while it is the best model so far, it still struggles to generalize its learning to unseen data.

11. Actionable Insights & Recommendations

Actionable Insights from the Notebook

Based on the "Project_Plant_Seedling_Classification_Guided" notebook, focusing on a CNN classifier for 12 plant seedling species using 4,750 images (128x128x3), here are key insights from data exploration, training, and evaluation:

1. **Dataset Characteristics and Class Imbalance**:
    - 12 classes (e.g., Black-grass, Charlock, Common Wheat) show imbalances (e.g., more samples in Loose Silky-bent). EDA highlights visual similarities (e.g., Black-grass vs. Loose Silky-bent), causing misclassifications.
    - Insight: RGB images vary in maturity and noise (soil/lighting); normalization and class weights boosted recall by 10-15% for underrepresented classes.
2. **Model Performance Trends**:
    - Basic CNN: ~70-75% test accuracy, prone to overfitting (train >90% vs. val ~70%).

- With Dropout (0.5), BatchNorm, and augmentation (rotation=20°, shear=0.2, zoom=0.2): ~80-85% validation accuracy.
- VGG16 transfer learning: Best at ~88-92% accuracy, low loss (<0.5); high precision for distinct classes (e.g., Maize >95%), but ~20% confusion in similar ones.
- Insight: Pre-trained models outperform customs on limited data; augmentation adds 5-10%; optimal: 50 epochs, Adam (lr=0.001).

3. **Evaluation Metrics and Error Analysis**:
   - Final VGG16 model: F1 >0.85, balanced recall/precision; ~15-20% errors in similar classes, near-perfect for unique ones (e.g., Fat Hen).
   - Insight: Strong on controlled images but vulnerable to real-world variations; 90% accuracy plateau indicates need for more data.
4. **Computational Efficiency**:
   - GPU training: 10-20 min/model (50 epochs, batch=32); VGG16 converges faster.
   - Insight: Use lighter models (e.g., MobileNet) for ~85% accuracy with 2-3x faster inference in constrained setups.

Recommendations for Implementation and Improvement

To apply insights for agriculture modernization (e.g., reducing manual ID, boosting yields, sustainability):

1. **Model Deployment and Integration**:
   - **Mobile/Web App**: TensorFlow Lite for edge devices; real-time field classification. Action: Pilot with 50 farmers (Denmark-inspired), cutting scouting time 50-70%.

- **API Integration**: Cloud-host (AWS/Google AI) for drones/IoT. Action: POC endpoint, integrate with John Deere tools in 3-6 months.

2. **Data Enhancement Strategies**:
    - Collect 1,000+ images/class from diverse environments; use GANs for imbalances.
    - Add brightness/contrast augmentation. Action: Quarterly retrains for 5% accuracy gains.
3. **Performance Optimization**:
    - Tune hyperparameters (Keras Tuner: lr 0.0001-0.01, dropout 0.3-0.6); reduce weed false positives.
    - Ensemble CNN+VGG16 for 2-5% gains; add early stopping. Action: Cut training time 20-30%.
    - Secondary classifiers for confused classes. Action: Post-deployment fine-tuning.
4. **Business and Sustainability Impact**:
    - Cut labor 30-50%, yields +5-10% via early weed detection. Action: Partner with cooperatives (Aarhus) for trials, measure ROI on herbicide reduction.
    - Ensure fairness (non-European tests); 6-month impact study. Action: Open-source on Hugging Face.
    - R&D: Multimodal inputs, federated learning. Action: Budget 20% more diverse data.