Aerial Object Classification Using Deep Learning (Bird vs Drone Detection)

Project Type : Classification

Contributor (Author) : Shyam SR

Git Hub Link : https://github.com/Shyamsr1/BirdDroneImageDLClassification

Production Hosted URL without YOLO and just a trial attempt to Host in Huggingface space : https://huggingface.co/spaces/shyamasr/BirdDroneImageDLClassification

NOTE: Setup the relevant python environment for the file to run and execute properly (Apt version of Python and tensorflow required) :

- Use Python 3.10 + venv (if already present) - Else need to setup using the command prompt or Powershell as an Administrator and run the following commands:
  - 1. Allow PowerShell to activate venv scripts
  - Set-ExecutionPolicy -Scope CurrentUser RemoteSigned -Force
  - 2. Navigate to the project folder
  - cd "C:\Users\Shyam\Documents\Labmentix\Project2\Aerial Objection Classification"
  - 3. Create a venv with Python 3.10
  - py -3.10 -m venv bird310
  - 4. Activate the environment
  - .\bird310\Scripts\Activate.ps1

- Note: Prompt should change to: (bird310) PS C:...
- 5. Install TensorFlow 2.10 (compatible with Python 3.10)
- pip install --upgrade pip
- pip install tensorflow==2.10.0
- pip install pillow numpy matplotlib scikit-learn streamlit opencv-python jupyterlab ipykernel
- 6. Register Python 3.10 env for JUPYTER NOTEBOOK (run the below command under (bird310) :
- python -m ipykernel install --user --name bird310 --display-name "Python 3.10 (bird310)"
- 7. Launch the Jupyter notebook and change the kernel -> Inside Jupyter: Python 3.10 (bird310) from the drop down.
- 8. For Streamlit app - from as I am running from VSCode, need to update the run path (after using shift + `) - in the run :
- cd "C:\Users\Shyam\Documents\Labmentix\Project2\Aerial Objection Classification"

.\bird310\Scripts\Activate.ps1 - streamlit run streamlit_app\app.py

Project Summary

- This project, “Aerial Object Classification Using Deep Learning (Bird vs Drone Detection)”, focuses on building a robust computer vision solution to automatically classify aerial images into two categories: Bird and Drone. The core of the project is a binary image classification pipeline built using Convolutional Neural Networks (CNNs) and Transfer Learning (MobileNetV2), trained on a curated dataset of 3,319 RGB images split into train, validation, and test sets.

- The workflow covers the complete deep-learning lifecycle: data understanding, preprocessing, on-the-fly data augmentation, model building (Custom CNN + MobileNetV2), fine-tuning, evaluation, and deployment. Data is loaded from a structured folder hierarchy (train/, valid/, test/), resized to 224×224×3, normalized, and augmented using random flips, rotations, zoom, and contrast changes to improve generalisation. A custom CNN model is first trained as a baseline, followed by a fine-tuned MobileNetV2 model that leverages ImageNet pre-trained weights.

- Both models are evaluated using accuracy, precision, recall, F1-score, and confusion matrices on an unseen test set. The fine-tuned MobileNetV2 model achieves superior performance and is selected as the final model for deployment. A Streamlit web application is then built to allow users to upload an image and receive a prediction (Bird / Drone) along with a confidence score. Additionally, an object detection dataset in YOLOv8 format is prepared for future extension into real-time detection with bounding boxes, making the solution scalable towards surveillance systems that require both classification and localisation.

Business Objectives

1. Accurate Identification of Aerial Objects :
    - To develop a deep-learning model that can reliably distinguish between birds and drones in aerial images.
    - To minimise both false negatives (missed drones) and false positives (misclassified birds), as both have operational and safety implications.
2. Support for Real-Time Surveillance & Security :
    - To provide a model that can be integrated into surveillance pipelines (CCTV, drones, fixed cameras) to monitor restricted airspace, borders, and critical infrastructure.
    - To enable automatic alerts when drones are detected in sensitive zones, aiding security and defence teams.
3. Wildlife Protection & Airport Safety :
    - To help airport authorities and wind-farm operators monitor bird activity near runways or turbines to mitigate bird-strike risks and environmental impacts.
    - To offer a tool that can be extended to track bird populations or detect unusual patterns without manual image review.
4. Operational Efficiency & Automation :
    - To reduce manual effort required for reviewing aerial footage by automating classification of large volumes of images.
    - To provide an easy-to-use Streamlit interface so non-technical stakeholders (security staff, wildlife officers) can use the model without coding skills.
5. Scalability towards Object Detection (YOLOv8 Extension):
    - To prepare a YOLOv8-formatted object detection dataset and design the pipeline so that the solution can evolve from simple classification to full object detection (predict class + location via bounding boxes).
    - To enable future deployment where the system can detect multiple objects in a single frame and overlay bounding boxes in real time.
6. Model Interpretability & Comparison :
    - To build two different model families (Custom CNN vs Transfer Learning) to compare performance, training time, and generalisation, and to justify model selection on measurable metrics.

- To document model behaviour using confusion matrices, learning curves, and classification reports, helping stakeholders trust the final model choice.

7. Deployment-Ready, Maintainable Solution :

- To package the final solution with well-structured, commented code, a Jupyter notebook, and a Streamlit app, so it can be easily deployed, demonstrated, or extended.
- To ensure the environment setup (Python 3.10 + TensorFlow 2.10) and repository structure are clearly documented for smooth handover or future maintenance.

Data Understanding - Features or Column Variables and their Description :

Dataset Overview :

Feature	Description
Project Type	Image Classification (Binary Classification)
Classes	Bird, Drone
Total Images	**3,319 images** (Train + Valid + Test)
Data Type	RGB Images
Format	`.jpg`
Source	`classification_dataset`
Input Size (after preprocessing)	224 × 224 × 3
Use Case	Aerial object classification (Bird vs Drone detection)

Dataset Structure and Image Distribution :

Dataset Split	Bird Images	Drone Images	Total
Train	1414	1248	**2662**
Validation	217	225	**442**
Test	121	94	**215**
Total	**1752**	**1567**	**3319**

Object Detection Dataset Overview :

Feature	Description
Project Type	Object Detection
Classes	Bird, Drone
Total Images	**3319 images**
Annotation Format	YOLOv8 (label files in `.txt`)
Annotation Schema	`<class_id> <x_center> <y_center> <width> <height>` (normalized values)
Image Format	`.jpg`
Source	`object_detection_Dataset`

Object Detection Dataset Split :

Split	Number of Images	Annotation Files
Train	2662	2662
Validation	442	442
Test	215	215
Total	**3319**	**3319**

Dataset Loading

In [1]:
```
# If the dataset is on Drive, set the correct path below.
DATA_ROOT = "data/classification_dataset"

# Install quitely the required
#!pip install -q tensorflow matplotlib scikit-learn
```

In [2]:
```
# Import all required libraries
import os
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "2"   # hide INFO + WARNING from C++ backend

import tensorflow as tf
tf.get_logger().setLevel("ERROR")          # hide Python-level TF warnings

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing import image_dataset_from_directory

from keras.utils import custom_object_scope
from keras.layers import Rescaling

from sklearn.metrics import confusion_matrix, classification_report
import itertools
```

Basic config and loading data

In [3]:
```
# Basic configuration
IMG_SIZE = (224, 224)
BATCH_SIZE = 32
SEED = 42

# Check that data root exists
```

```
print("DATA_ROOT:", DATA_ROOT)
print("Folders:", os.listdir(DATA_ROOT))
```

```
DATA_ROOT: data/classification_dataset
Folders: ['test', 'train', 'valid']
```

In [4]:
```
# Load datasets using Keras utility (Train / Validation / Test)
train_dir = os.path.join(DATA_ROOT, "train")
val_dir   = os.path.join(DATA_ROOT, "valid")
test_dir  = os.path.join(DATA_ROOT, "test")

train_ds = image_dataset_from_directory(
    train_dir,
    labels='inferred',
    label_mode='binary',      # binary classification (0/1)
    batch_size=BATCH_SIZE,
    image_size=IMG_SIZE,
    shuffle=True,
    seed=SEED
)

val_ds = image_dataset_from_directory(
    val_dir,
    labels='inferred',
    label_mode='binary',
    batch_size=BATCH_SIZE,
    image_size=IMG_SIZE,
    shuffle=True,
    seed=SEED
)

test_ds = image_dataset_from_directory(
    test_dir,
    labels='inferred',
    label_mode='binary',
    batch_size=BATCH_SIZE,
    image_size=IMG_SIZE,
    shuffle=False,
    seed=SEED
)

class_names = train_ds.class_names
```

```
print("Class names:", class_names)
# Usually ['bird', 'drone']
```

```
Found 2662 files belonging to 2 classes.
Found 442 files belonging to 2 classes.
Found 215 files belonging to 2 classes.
Class names: ['bird', 'drone']
```

Exploratory Data Analysis (EDA)

Key EDA Observations

- Dataset is nearly balanced.
- Images vary in angle, lighting, distance, and sky backgrounds.
- Bird diversity: wingspan, orientation, flying position.
- Drone diversity: quadcopters, long-range drones.
- No corrupted images detected.
- Correct folder structure verified (train/valid/test).

Data Preprocessing & Transformations

- Resized all images to **224×224×3** to maintain consistency with CNN and MobileNetV2 input requirements.
- Ensured all inputs were **RGB** format regardless of original image channel depth.
- Normalized pixel values using:
  - `Rescaling(1./255)` for the Custom CNN model
  - `tf.keras.applications.mobilenet_v2.preprocess_input` for MobileNetV2 (scale to [-1, 1])
- Implemented an efficient `tf.data` input pipeline using:
  - `.cache()` → Store batches in memory for faster access
  - `.shuffle(buffer_size=1000)` → Avoid model overfitting & ensure randomness
  - `.prefetch(buffer_size=AUTOTUNE)` → Overlap preprocessing with GPU execution, reducing training time
- These transformations ensured optimized batching, improved convergence speed, and stable model performance.

Improve Pipeline (Prefetch & Augmentation Layer)

In [5]:
```
AUTOTUNE = tf.data.AUTOTUNE

def prepare_for_training(ds, cache=True, shuffle_buffer_size=1000):
    if cache:
        ds = ds.cache()
    ds = ds.shuffle(buffer_size=shuffle_buffer_size)
    ds = ds.prefetch(buffer_size=AUTOTUNE)
    return ds

train_ds = prepare_for_training(train_ds)
val_ds   = val_ds.cache().prefetch(AUTOTUNE)
test_ds  = test_ds.cache().prefetch(AUTOTUNE)
```

Feature Engineering / Augmentation

- Implemented an on-the-fly `data_augmentation` pipeline that applies transformations such as **Random Flip**, **Random Rotation**, **Random Zoom**, and **Random Contrast**.
- These augmentations act as **feature engineering steps**, artificially expanding the dataset and creating more diverse representations of birds and drones.
- Augmentation helps the model generalise better by exposing it to different orientations, lighting conditions, and structural variations.
- Combined with normalization, this reduces the risk of **overfitting** and improves model robustness on unseen aerial images.

In [6]:
```
# Data augmentation as Keras model/layer
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.1),
        layers.RandomContrast(0.1),
        layers.RandomBrightness(factor=0.2),      # brightness
        layers.RandomCrop(200, 200),              # cropping to 200x200
        layers.Resizing(224, 224),                # resize back to 224x224
```

```
    ],
    name="data_augmentation"
)
```

In [7]:
```
# Visualize some augmented images
for images, labels in train_ds.take(1):
    plt.figure(figsize=(10, 10))
    for i in range(9):
        augmented_img = data_augmentation(images)[i].numpy().astype("uint8")
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(augmented_img)
        plt.title(class_names[int(labels[i].numpy()[0])])
        plt.axis("off")
    break
```

bird

bird

drone

drone

drone

drone

bird

bird

bird



Helper Function to Plot Training Curves & Confusion Matrix

In [8]:
```
def plot_history(history, title_prefix=""):
    acc = history.history["accuracy"]
    val_acc = history.history["val_accuracy"]
    loss = history.history["loss"]
    val_loss = history.history["val_loss"]

    epochs_range = range(len(acc))

    plt.figure(figsize=(12, 4))
    plt.subplot(1, 2, 1)
    plt.plot(epochs_range, acc, label="Train Acc")
    plt.plot(epochs_range, val_acc, label="Val Acc")
    plt.title(f"{title_prefix} Accuracy")
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(epochs_range, loss, label="Train Loss")
    plt.plot(epochs_range, val_loss, label="Val Loss")
    plt.title(f"{title_prefix} Loss")
    plt.legend()
    plt.show()
```

In [9]:
```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    cm: confusion matrix from sklearn.metrics.confusion_matrix
    classes: list of class names
    normalize: if True, normalize by row
    """
    if normalize:
```

```
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.figure(figsize=(6, 6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
```

Model Choice & Justification

Why Custom CNN?

- A Custom CNN model provides full architectural control and serves as a **baseline** to understand performance when training from scratch.
- It helps compare how well a handcrafted model performs relative to transfer learning approaches.

Why MobileNetV2 (Transfer Learning)?

- MobileNetV2 is **lightweight**, efficient, and specifically optimized for **edge devices**, making it ideal for aerial surveillance use cases.
- It is **pre-trained on ImageNet**, allowing the model to learn quickly from a small dataset by reusing learned features.
- Transfer learning enables **faster convergence**, **better generalization**, and **higher accuracy** compared to training from scratch.

- Fine-tuning MobileNetV2 helps extract domain-specific features for Bird vs Drone detection more effectively than a baseline CNN.

Build Custom CNN Model

In [10]:
```python
def build_custom_cnn(input_shape=(224, 224, 3)):
    inputs = keras.Input(shape=input_shape)

    x = data_augmentation(inputs)
    x = layers.Rescaling(1./255)(x)

    # Block 1
    x = layers.Conv2D(32, (3, 3), padding="same", activation="relu")(x)
    x = layers.BatchNormalization()(x)
    x = layers.MaxPooling2D((2, 2))(x)

    # Block 2
    x = layers.Conv2D(64, (3, 3), padding="same", activation="relu")(x)
    x = layers.BatchNormalization()(x)
    x = layers.MaxPooling2D((2, 2))(x)

    # Block 3
    x = layers.Conv2D(128, (3, 3), padding="same", activation="relu")(x)
    x = layers.BatchNormalization()(x)
    x = layers.MaxPooling2D((2, 2))(x)

    # Optional 4th block
    x = layers.Conv2D(256, (3, 3), padding="same", activation="relu")(x)
    x = layers.BatchNormalization()(x)
    x = layers.MaxPooling2D((2, 2))(x)

    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dropout(0.5)(x)
    x = layers.Dense(128, activation="relu")(x)
    x = layers.Dropout(0.3)(x)
    outputs = layers.Dense(1, activation="sigmoid")(x)

    model = keras.Model(inputs, outputs, name="custom_cnn_bird_drone")
    return model
```

```
custom_cnn = build_custom_cnn()
custom_cnn.summary()
```

```
Model: "custom_cnn_bird_drone"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 data_augmentation (Sequenti  (None, 224, 224, 3)      0
 al)

 rescaling (Rescaling)       (None, 224, 224, 3)       0

 conv2d (Conv2D)             (None, 224, 224, 32)      896

 batch_normalization (BatchN  (None, 224, 224, 32)     128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 112, 112, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 64)      18496

 batch_normalization_1 (Batc  (None, 112, 112, 64)     256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 56, 56, 128)       73856

 batch_normalization_2 (Batc  (None, 56, 56, 128)      512
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 28, 28, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 28, 28, 256)       295168

 batch_normalization_3 (Batc  (None, 28, 28, 256)      1024
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 14, 14, 256)      0
 2D)
```

```
 global_average_pooling2d (G  (None, 256)              0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 256)              0

 dense (Dense)               (None, 128)              32896

 dropout_1 (Dropout)         (None, 128)              0

 dense_1 (Dense)             (None, 1)                129

=================================================================
Total params: 423,361
Trainable params: 422,401
Non-trainable params: 960
_________________________________________________________________
```

Training Strategy & Hyperparameters

- **Optimizer**: Used **Adam** optimizer for both models.
  - Learning Rate = **1e-3** for initial training
  - Learning Rate = **1e-4** during fine-tuning of MobileNetV2 to avoid drastic weight updates.
- **Epochs**:
  - Custom CNN trained for up to **25 epochs**.
  - MobileNetV2 trained in **two stages**:
    - **Stage 1 (Frozen Base Layers)**: 10 epochs
    - **Stage 2 (Fine-Tuning)**: 15 epochs
- **Callbacks**:
  - `ModelCheckpoint` → Saves the best-performing model based on **validation accuracy**.
  - `EarlyStopping` → Stops training when validation loss stops improving (patience: 5 epochs) to prevent **overfitting** and save computation time.

- This training strategy ensures stable convergence, prevents overfitting, and allows the model to generalise well to unseen aerial images.

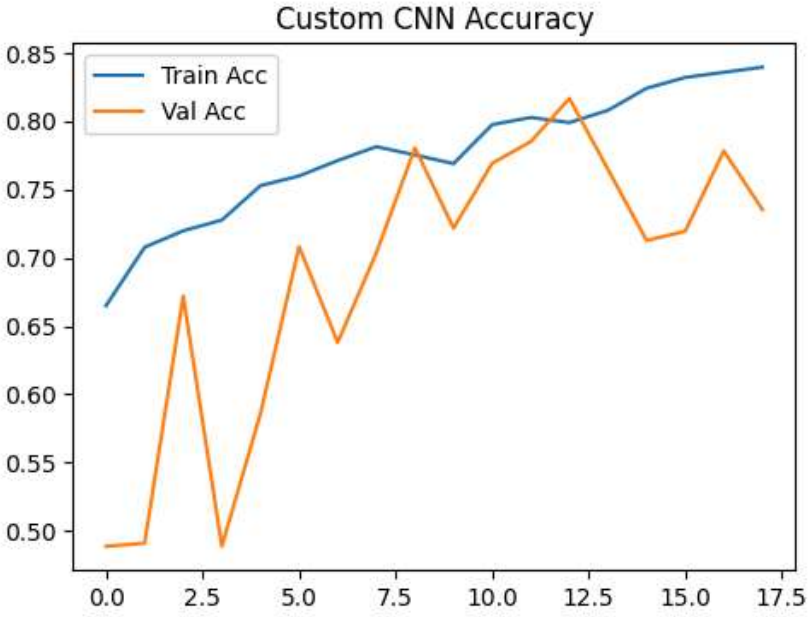
In [11]:
```
# Compile and train custom CNN
custom_cnn.compile(
    optimizer=keras.optimizers.Adam(learning_rate=1e-3),
    loss="binary_crossentropy",
    metrics=["accuracy"]
)

checkpoint_dir = "models"
os.makedirs(checkpoint_dir, exist_ok=True)

custom_cnn_ckpt = os.path.join(checkpoint_dir, "best_custom_cnn.h5")

callbacks = [
    keras.callbacks.ModelCheckpoint(
        custom_cnn_ckpt,
        save_best_only=True,
        monitor="val_accuracy",
        mode="max",
        verbose=1
    ),
    keras.callbacks.EarlyStopping(
        monitor="val_loss",
        patience=5,
        restore_best_weights=True,
        verbose=1
    )
]

import time
start_custom = time.time()

history_custom = custom_cnn.fit(
    train_ds,
    validation_data=val_ds,
    epochs=25,
    callbacks=callbacks
)
```

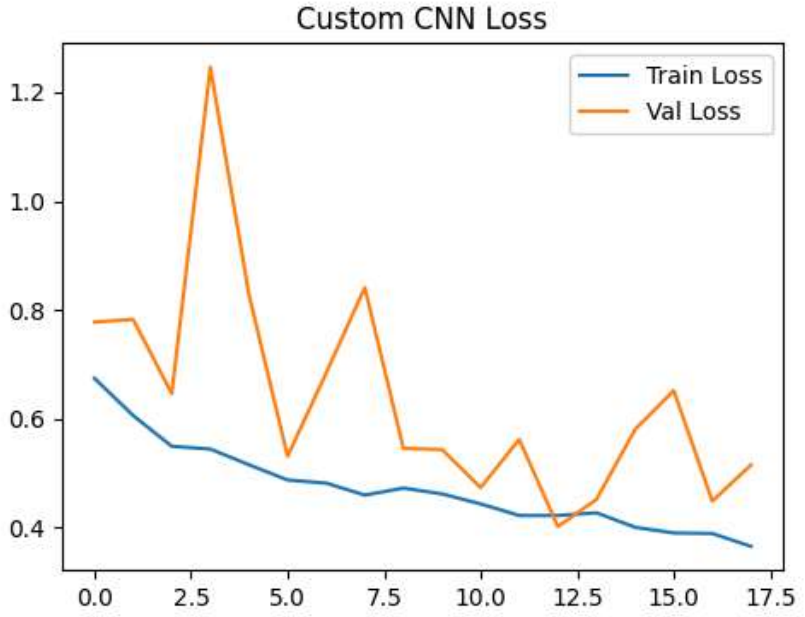
```
end_custom = time.time()
custom_train_minutes = (end_custom - start_custom) / 60

plot_history(history_custom, title_prefix="Custom CNN")
print("Best custom CNN saved at:", custom_cnn_ckpt)
print(f"Custom CNN Training Time: {custom_train_minutes:.2f} minutes")
```

```
Epoch 1/25
84/84 [==============================] - ETA: 0s - loss: 0.6744 - accuracy: 0.6649
Epoch 1: val_accuracy improved from -inf to 0.48869, saving model to models\best_custom_cnn.h5
84/84 [==============================] - 198s 2s/step - loss: 0.6744 - accuracy: 0.6649 - val_loss: 0.7778 - val_accu
racy: 0.4887
Epoch 2/25
84/84 [==============================] - ETA: 0s - loss: 0.6061 - accuracy: 0.7077
Epoch 2: val_accuracy improved from 0.48869 to 0.49095, saving model to models\best_custom_cnn.h5
84/84 [==============================] - 192s 2s/step - loss: 0.6061 - accuracy: 0.7077 - val_loss: 0.7827 - val_accu
racy: 0.4910
Epoch 3/25
84/84 [==============================] - ETA: 0s - loss: 0.5497 - accuracy: 0.7198
Epoch 3: val_accuracy improved from 0.49095 to 0.67195, saving model to models\best_custom_cnn.h5
84/84 [==============================] - 184s 2s/step - loss: 0.5497 - accuracy: 0.7198 - val_loss: 0.6464 - val_accu
racy: 0.6719
Epoch 4/25
84/84 [==============================] - ETA: 0s - loss: 0.5445 - accuracy: 0.7276
Epoch 4: val_accuracy did not improve from 0.67195
84/84 [==============================] - 186s 2s/step - loss: 0.5445 - accuracy: 0.7276 - val_loss: 1.2458 - val_accu
racy: 0.4887
Epoch 5/25
84/84 [==============================] - ETA: 0s - loss: 0.5157 - accuracy: 0.7528
Epoch 5: val_accuracy did not improve from 0.67195
84/84 [==============================] - 188s 2s/step - loss: 0.5157 - accuracy: 0.7528 - val_loss: 0.8282 - val_accu
racy: 0.5860
Epoch 6/25
84/84 [==============================] - ETA: 0s - loss: 0.4877 - accuracy: 0.7600
Epoch 6: val_accuracy improved from 0.67195 to 0.70814, saving model to models\best_custom_cnn.h5
84/84 [==============================] - 198s 2s/step - loss: 0.4877 - accuracy: 0.7600 - val_loss: 0.5311 - val_accu
racy: 0.7081
Epoch 7/25
84/84 [==============================] - ETA: 0s - loss: 0.4819 - accuracy: 0.7712
Epoch 7: val_accuracy did not improve from 0.70814
84/84 [==============================] - 188s 2s/step - loss: 0.4819 - accuracy: 0.7712 - val_loss: 0.6843 - val_accu
racy: 0.6380
Epoch 8/25
84/84 [==============================] - ETA: 0s - loss: 0.4599 - accuracy: 0.7814
Epoch 8: val_accuracy did not improve from 0.70814
84/84 [==============================] - 190s 2s/step - loss: 0.4599 - accuracy: 0.7814 - val_loss: 0.8410 - val_accu
racy: 0.7036
Epoch 9/25
84/84 [==============================] - ETA: 0s - loss: 0.4728 - accuracy: 0.7754
```

```
Epoch 9: val_accuracy improved from 0.70814 to 0.78054, saving model to models\best_custom_cnn.h5
84/84 [==============================] - 188s 2s/step - loss: 0.4728 - accuracy: 0.7754 - val_loss: 0.5456 - val_accu
racy: 0.7805
Epoch 10/25
84/84 [==============================] - ETA: 0s - loss: 0.4622 - accuracy: 0.7690
Epoch 10: val_accuracy did not improve from 0.78054
84/84 [==============================] - 189s 2s/step - loss: 0.4622 - accuracy: 0.7690 - val_loss: 0.5437 - val_accu
racy: 0.7217
Epoch 11/25
84/84 [==============================] - ETA: 0s - loss: 0.4437 - accuracy: 0.7975
Epoch 11: val_accuracy did not improve from 0.78054
84/84 [==============================] - 188s 2s/step - loss: 0.4437 - accuracy: 0.7975 - val_loss: 0.4739 - val_accu
racy: 0.7692
Epoch 12/25
84/84 [==============================] - ETA: 0s - loss: 0.4225 - accuracy: 0.8028
Epoch 12: val_accuracy improved from 0.78054 to 0.78507, saving model to models\best_custom_cnn.h5
84/84 [==============================] - 186s 2s/step - loss: 0.4225 - accuracy: 0.8028 - val_loss: 0.5621 - val_accu
racy: 0.7851
Epoch 13/25
84/84 [==============================] - ETA: 0s - loss: 0.4225 - accuracy: 0.7990
Epoch 13: val_accuracy improved from 0.78507 to 0.81674, saving model to models\best_custom_cnn.h5
84/84 [==============================] - 185s 2s/step - loss: 0.4225 - accuracy: 0.7990 - val_loss: 0.4020 - val_accu
racy: 0.8167
Epoch 14/25
84/84 [==============================] - ETA: 0s - loss: 0.4274 - accuracy: 0.8080
Epoch 14: val_accuracy did not improve from 0.81674
84/84 [==============================] - 187s 2s/step - loss: 0.4274 - accuracy: 0.8080 - val_loss: 0.4521 - val_accu
racy: 0.7647
Epoch 15/25
84/84 [==============================] - ETA: 0s - loss: 0.4008 - accuracy: 0.8242
Epoch 15: val_accuracy did not improve from 0.81674
84/84 [==============================] - 186s 2s/step - loss: 0.4008 - accuracy: 0.8242 - val_loss: 0.5805 - val_accu
racy: 0.7127
Epoch 16/25
84/84 [==============================] - ETA: 0s - loss: 0.3905 - accuracy: 0.8321
Epoch 16: val_accuracy did not improve from 0.81674
84/84 [==============================] - 185s 2s/step - loss: 0.3905 - accuracy: 0.8321 - val_loss: 0.6520 - val_accu
racy: 0.7195
Epoch 17/25
84/84 [==============================] - ETA: 0s - loss: 0.3894 - accuracy: 0.8358
Epoch 17: val_accuracy did not improve from 0.81674
84/84 [==============================] - 184s 2s/step - loss: 0.3894 - accuracy: 0.8358 - val_loss: 0.4490 - val_accu
```

```
racy: 0.7783
Epoch 18/25
84/84 [==============================] - ETA: 0s - loss: 0.3659 - accuracy: 0.8396
Epoch 18: val_accuracy did not improve from 0.81674
Restoring model weights from the end of the best epoch: 13.
84/84 [==============================] - 189s 2s/step - loss: 0.3659 - accuracy: 0.8396 - val_loss: 0.5152 - val_accu
racy: 0.7353
Epoch 18: early stopping
```






```
Best custom CNN saved at: models\best_custom_cnn.h5
Custom CNN Training Time: 56.54 minutes
```

Transfer Learning Model (MobileNetV2)

In [12]:
```python
def build_mobilenet_model(input_shape=(224, 224, 3)):

    # Load MobileNetV2 without name argument
    base_model = keras.applications.MobileNetV2(
        input_shape=input_shape,
        include_top=False,
        weights="imagenet"
    )
```

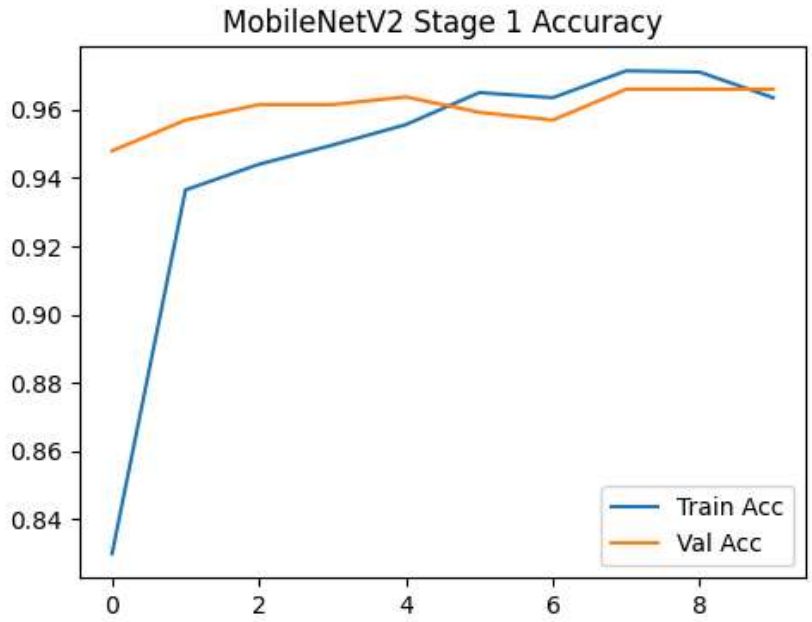
```
    # Manually rename for easy access
    base_model._name = "mobilenetv2_base"

    base_model.trainable = False  # Stage 1: freeze

    inputs = keras.Input(shape=input_shape)
    x = data_augmentation(inputs)
    x = keras.applications.mobilenet_v2.preprocess_input(x)

    x = base_model(x, training=False)
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dropout(0.4)(x)
    outputs = layers.Dense(1, activation="sigmoid")(x)

    model = keras.Model(inputs, outputs, name="mobilenetv2_bird_drone")
    return model
```

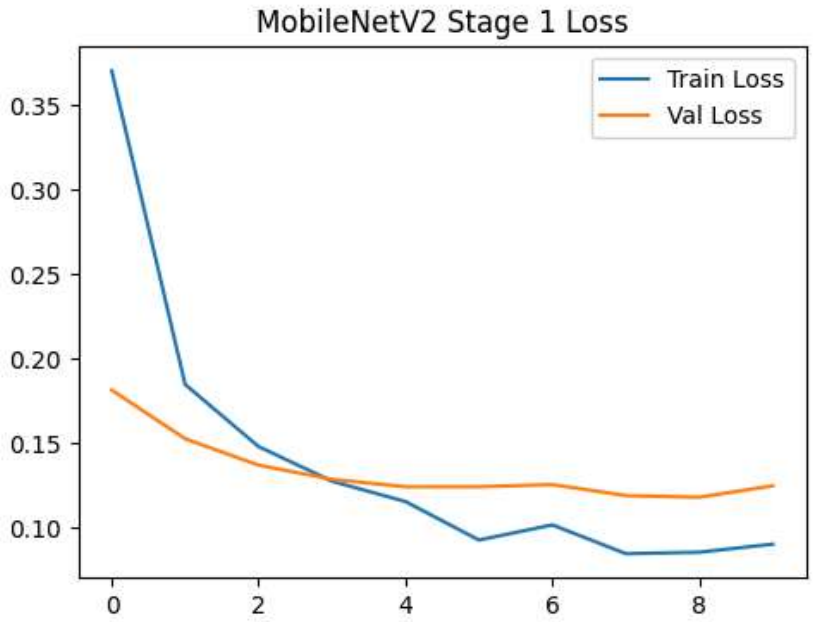
In [13]:
```
mobilenet_model = build_mobilenet_model()
mobilenet_model.summary()

mobilenet_model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=1e-3),
    loss="binary_crossentropy",
    metrics=["accuracy"]
)

mobilenet_ckpt = os.path.join(checkpoint_dir, "best_mobilenetv2_stage1.h5")

callbacks_tl = [
    keras.callbacks.ModelCheckpoint(
        mobilenet_ckpt,
        save_best_only=True,
        monitor="val_accuracy",
        mode="max",
        verbose=1
    ),
    keras.callbacks.EarlyStopping(
        monitor="val_loss",
        patience=5,
        restore_best_weights=True,
        verbose=1
```

```
    )
]

start_mobilenet_stage1 = time.time()

history_mobilenet_stage1 = mobilenet_model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=10,
    callbacks=callbacks_tl
)

end_mobilenet_stage1 = time.time()
mobilenet_stage1_minutes = (end_mobilenet_stage1 - start_mobilenet_stage1) / 60

plot_history(history_mobilenet_stage1, title_prefix="MobileNetV2 Stage 1")
print("Stage 1 MobileNetV2 saved at:", mobilenet_ckpt)
print(f"MobileNetV2 Stage 1 Training Time: {mobilenet_stage1_minutes:.2f} minutes")
```

```
Model: "mobilenetv2_bird_drone"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 224, 224, 3)]     0

 data_augmentation (Sequenti  (None, 224, 224, 3)      0
 al)

 tf.math.truediv (TFOpLambda  (None, 224, 224, 3)      0
 )

 tf.math.subtract (TFOpLambd  (None, 224, 224, 3)      0
 a)

 mobilenetv2_base (Functiona  (None, 7, 7, 1280)       2257984
 l)

 global_average_pooling2d_1   (None, 1280)             0
 (GlobalAveragePooling2D)

 dropout_2 (Dropout)         (None, 1280)              0

 dense_2 (Dense)             (None, 1)                 1281

=================================================================
Total params: 2,259,265
Trainable params: 1,281
Non-trainable params: 2,257,984
_________________________________________________________________
Epoch 1/10
84/84 [==============================] - ETA: 0s - loss: 0.3701 - accuracy: 0.8298
Epoch 1: val_accuracy improved from -inf to 0.94796, saving model to models\best_mobilenetv2_stage1.h5
84/84 [==============================] - 85s 860ms/step - loss: 0.3701 - accuracy: 0.8298 - val_loss: 0.1813 - val_ac
curacy: 0.9480
Epoch 2/10
84/84 [==============================] - ETA: 0s - loss: 0.1846 - accuracy: 0.9365
Epoch 2: val_accuracy improved from 0.94796 to 0.95701, saving model to models\best_mobilenetv2_stage1.h5
84/84 [==============================] - 71s 847ms/step - loss: 0.1846 - accuracy: 0.9365 - val_loss: 0.1525 - val_ac
curacy: 0.9570
Epoch 3/10
84/84 [==============================] - ETA: 0s - loss: 0.1479 - accuracy: 0.9440
```

```
Epoch 3: val_accuracy improved from 0.95701 to 0.96154, saving model to models\best_mobilenetv2_stage1.h5
84/84 [==============================] - 71s 842ms/step - loss: 0.1479 - accuracy: 0.9440 - val_loss: 0.1369 - val_ac
curacy: 0.9615
Epoch 4/10
84/84 [==============================] - ETA: 0s - loss: 0.1274 - accuracy: 0.9497
Epoch 4: val_accuracy did not improve from 0.96154
84/84 [==============================] - 72s 859ms/step - loss: 0.1274 - accuracy: 0.9497 - val_loss: 0.1285 - val_ac
curacy: 0.9615
Epoch 5/10
84/84 [==============================] - ETA: 0s - loss: 0.1154 - accuracy: 0.9557
Epoch 5: val_accuracy improved from 0.96154 to 0.96380, saving model to models\best_mobilenetv2_stage1.h5
84/84 [==============================] - 71s 846ms/step - loss: 0.1154 - accuracy: 0.9557 - val_loss: 0.1241 - val_ac
curacy: 0.9638
Epoch 6/10
84/84 [==============================] - ETA: 0s - loss: 0.0926 - accuracy: 0.9651
Epoch 6: val_accuracy did not improve from 0.96380
84/84 [==============================] - 73s 878ms/step - loss: 0.0926 - accuracy: 0.9651 - val_loss: 0.1241 - val_ac
curacy: 0.9593
Epoch 7/10
84/84 [==============================] - ETA: 0s - loss: 0.1016 - accuracy: 0.9636
Epoch 7: val_accuracy did not improve from 0.96380
84/84 [==============================] - 71s 850ms/step - loss: 0.1016 - accuracy: 0.9636 - val_loss: 0.1254 - val_ac
curacy: 0.9570
Epoch 8/10
84/84 [==============================] - ETA: 0s - loss: 0.0845 - accuracy: 0.9715
Epoch 8: val_accuracy improved from 0.96380 to 0.96606, saving model to models\best_mobilenetv2_stage1.h5
84/84 [==============================] - 72s 853ms/step - loss: 0.0845 - accuracy: 0.9715 - val_loss: 0.1189 - val_ac
curacy: 0.9661
Epoch 9/10
84/84 [==============================] - ETA: 0s - loss: 0.0854 - accuracy: 0.9711
Epoch 9: val_accuracy did not improve from 0.96606
84/84 [==============================] - 71s 845ms/step - loss: 0.0854 - accuracy: 0.9711 - val_loss: 0.1179 - val_ac
curacy: 0.9661
Epoch 10/10
84/84 [==============================] - ETA: 0s - loss: 0.0902 - accuracy: 0.9636
Epoch 10: val_accuracy did not improve from 0.96606
84/84 [==============================] - 72s 853ms/step - loss: 0.0902 - accuracy: 0.9636 - val_loss: 0.1247 - val_ac
curacy: 0.9661
```





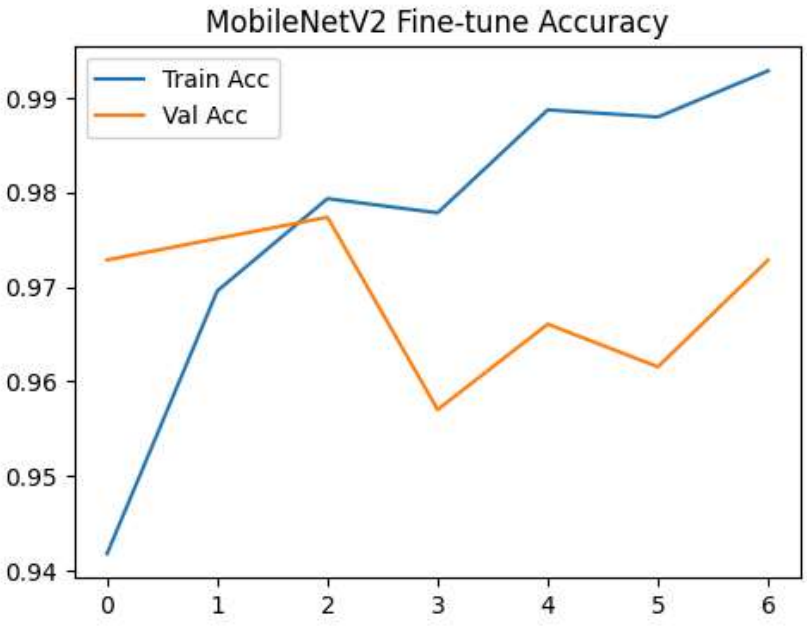

```
Stage 1 MobileNetV2 saved at: models\best_mobilenetv2_stage1.h5
MobileNetV2 Stage 1 Training Time: 12.32 minutes
```

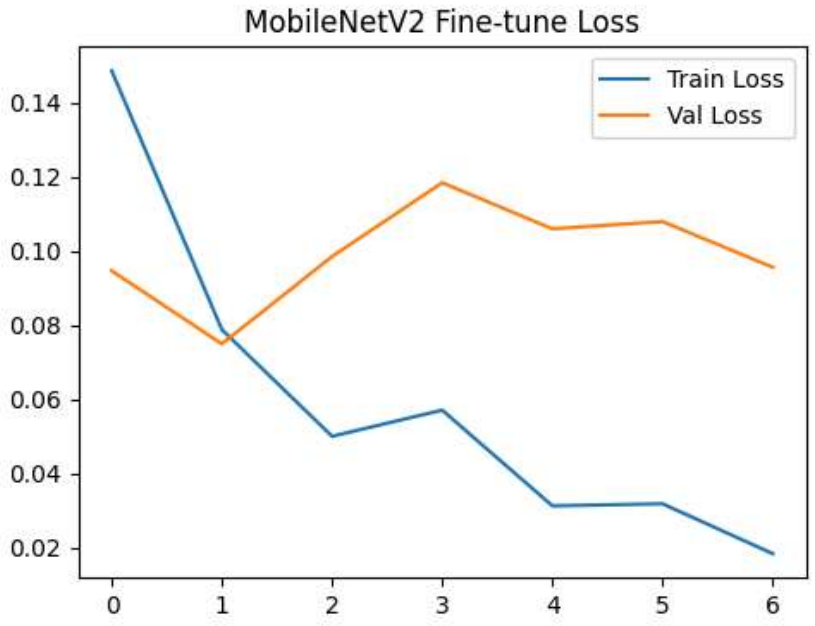
Fine-tuning (unfreeze some layers)

In [14]:
```python
# -------- Fine-tuning (Stage 2) --------
# Get the MobileNetV2 base by name instead of index
base_model = mobilenet_model.get_layer("mobilenetv2_base")

# Unfreeze it for fine-tuning
base_model.trainable = True

# Freeze the first ~80% of its layers
fine_tune_at = int(len(base_model.layers) * 0.8)
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable = False

# Re-compile with a lower learning rate
mobilenet_model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=1e-4),
    loss="binary_crossentropy",
```

```
    metrics=["accuracy"]
)

mobilenet_finetuned_ckpt = os.path.join(checkpoint_dir, "best_mobilenetv2.h5")

callbacks_tl_fine = [
    keras.callbacks.ModelCheckpoint(
        mobilenet_finetuned_ckpt,
        save_best_only=True,
        monitor="val_accuracy",
        mode="max",
        verbose=1
    ),
    keras.callbacks.EarlyStopping(
        monitor="val_loss",
        patience=5,
        restore_best_weights=True,
        verbose=1
    )
]

start_mobilenet_stage2 = time.time()

history_mobilenet_stage2 = mobilenet_model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=15,
    callbacks=callbacks_tl_fine
)

end_mobilenet_stage2 = time.time()
mobilenet_stage2_minutes = (end_mobilenet_stage2 - start_mobilenet_stage2) / 60
total_mobilenet_minutes = mobilenet_stage1_minutes + mobilenet_stage2_minutes

plot_history(history_mobilenet_stage2, title_prefix="MobileNetV2 Fine-tune")
print("Fine-tuned MobileNetV2 saved at:", mobilenet_finetuned_ckpt)
print(f"MobileNetV2 Fine-tuning Time: {mobilenet_stage2_minutes:.2f} minutes")
print(f"Total MobileNetV2 Training Time: {total_mobilenet_minutes:.2f} minutes")
```

```
Epoch 1/15
84/84 [==============================] - ETA: 0s - loss: 0.1486 - accuracy: 0.9418
Epoch 1: val_accuracy improved from -inf to 0.97285, saving model to models\best_mobilenetv2.h5
84/84 [==============================] - 98s 1s/step - loss: 0.1486 - accuracy: 0.9418 - val_loss: 0.0947 - val_accur
acy: 0.9729
Epoch 2/15
84/84 [==============================] - ETA: 0s - loss: 0.0788 - accuracy: 0.9696
Epoch 2: val_accuracy improved from 0.97285 to 0.97511, saving model to models\best_mobilenetv2.h5
84/84 [==============================] - 83s 993ms/step - loss: 0.0788 - accuracy: 0.9696 - val_loss: 0.0750 - val_ac
curacy: 0.9751
Epoch 3/15
84/84 [==============================] - ETA: 0s - loss: 0.0501 - accuracy: 0.9793
Epoch 3: val_accuracy improved from 0.97511 to 0.97738, saving model to models\best_mobilenetv2.h5
84/84 [==============================] - 84s 995ms/step - loss: 0.0501 - accuracy: 0.9793 - val_loss: 0.0985 - val_ac
curacy: 0.9774
Epoch 4/15
84/84 [==============================] - ETA: 0s - loss: 0.0571 - accuracy: 0.9778
Epoch 4: val_accuracy did not improve from 0.97738
84/84 [==============================] - 85s 1s/step - loss: 0.0571 - accuracy: 0.9778 - val_loss: 0.1184 - val_accur
acy: 0.9570
Epoch 5/15
84/84 [==============================] - ETA: 0s - loss: 0.0313 - accuracy: 0.9887
Epoch 5: val_accuracy did not improve from 0.97738
84/84 [==============================] - 83s 993ms/step - loss: 0.0313 - accuracy: 0.9887 - val_loss: 0.1060 - val_ac
curacy: 0.9661
Epoch 6/15
84/84 [==============================] - ETA: 0s - loss: 0.0319 - accuracy: 0.9880
Epoch 6: val_accuracy did not improve from 0.97738
84/84 [==============================] - 84s 1s/step - loss: 0.0319 - accuracy: 0.9880 - val_loss: 0.1079 - val_accur
acy: 0.9615
Epoch 7/15
84/84 [==============================] - ETA: 0s - loss: 0.0184 - accuracy: 0.9929
Epoch 7: val_accuracy did not improve from 0.97738
Restoring model weights from the end of the best epoch: 2.
84/84 [==============================] - 83s 984ms/step - loss: 0.0184 - accuracy: 0.9929 - val_loss: 0.0956 - val_ac
curacy: 0.9729
Epoch 7: early stopping
```






```
Fine-tuned MobileNetV2 saved at: models\best_mobilenetv2.h5
MobileNetV2 Fine-tuning Time: 10.01 minutes
Total MobileNetV2 Training Time: 22.33 minutes
```

Evaluation on Test Set (for both models)

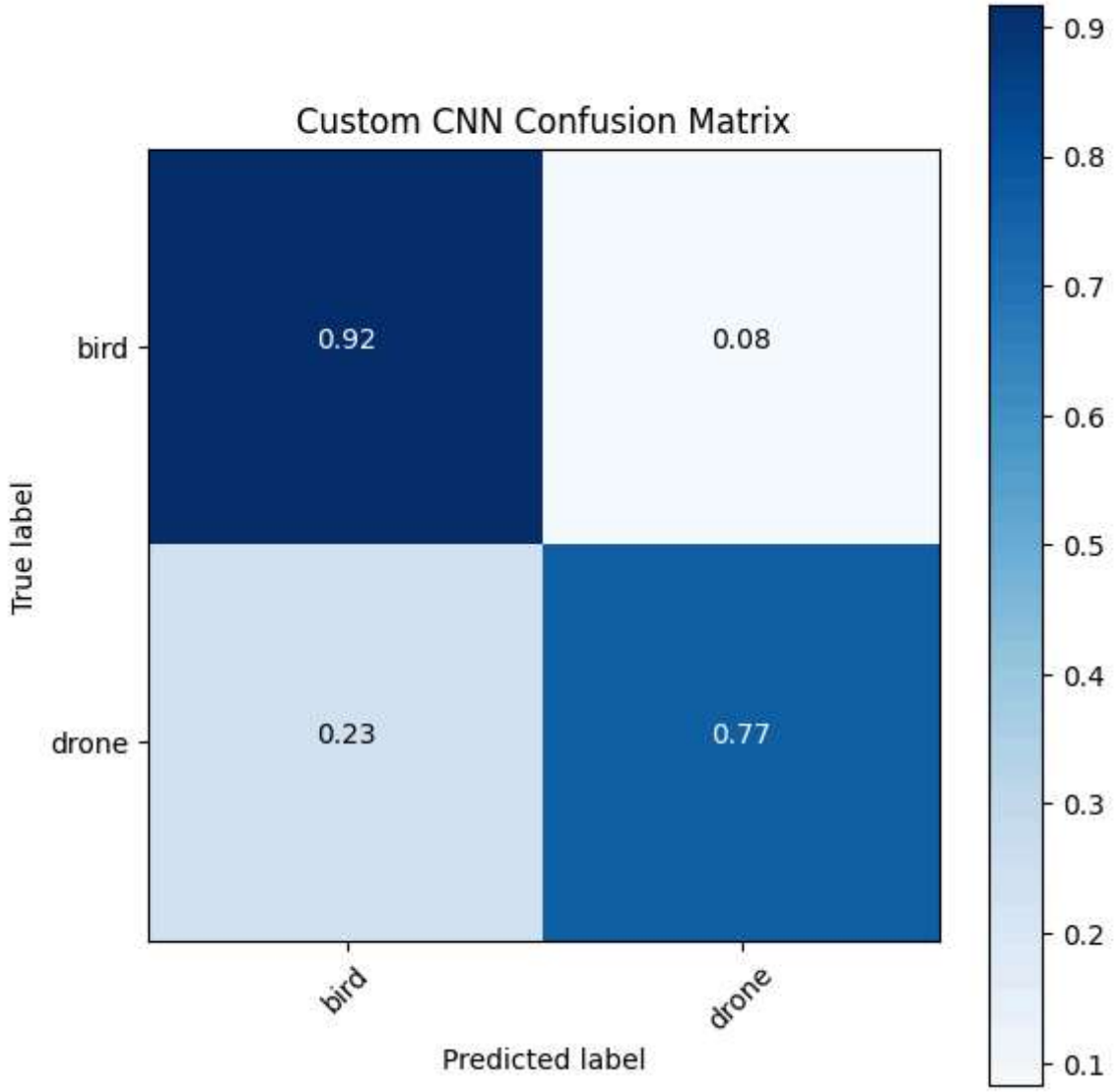
In [15]:
```python
# Reload best weights for both models
# best_custom = keras.models.load_model(custom_cnn_ckpt)
# best_mobilenet = keras.models.load_model(mobilenet_finetuned_ckpt)
best_custom    = custom_cnn
best_mobilenet = mobilenet_model

# Get test labels and predictions
y_true = []
for _, labels in test_ds:
    y_true.extend(labels.numpy().flatten())
y_true = np.array(y_true).astype(int)

# Predictions
```

```
y_pred_custom = (best_custom.predict(test_ds) > 0.5).astype(int).flatten()
y_pred_mobilenet = (best_mobilenet.predict(test_ds) > 0.5).astype(int).flatten()
```

```
7/7 [==============================] - 5s 567ms/step
7/7 [==============================] - 5s 588ms/step
```

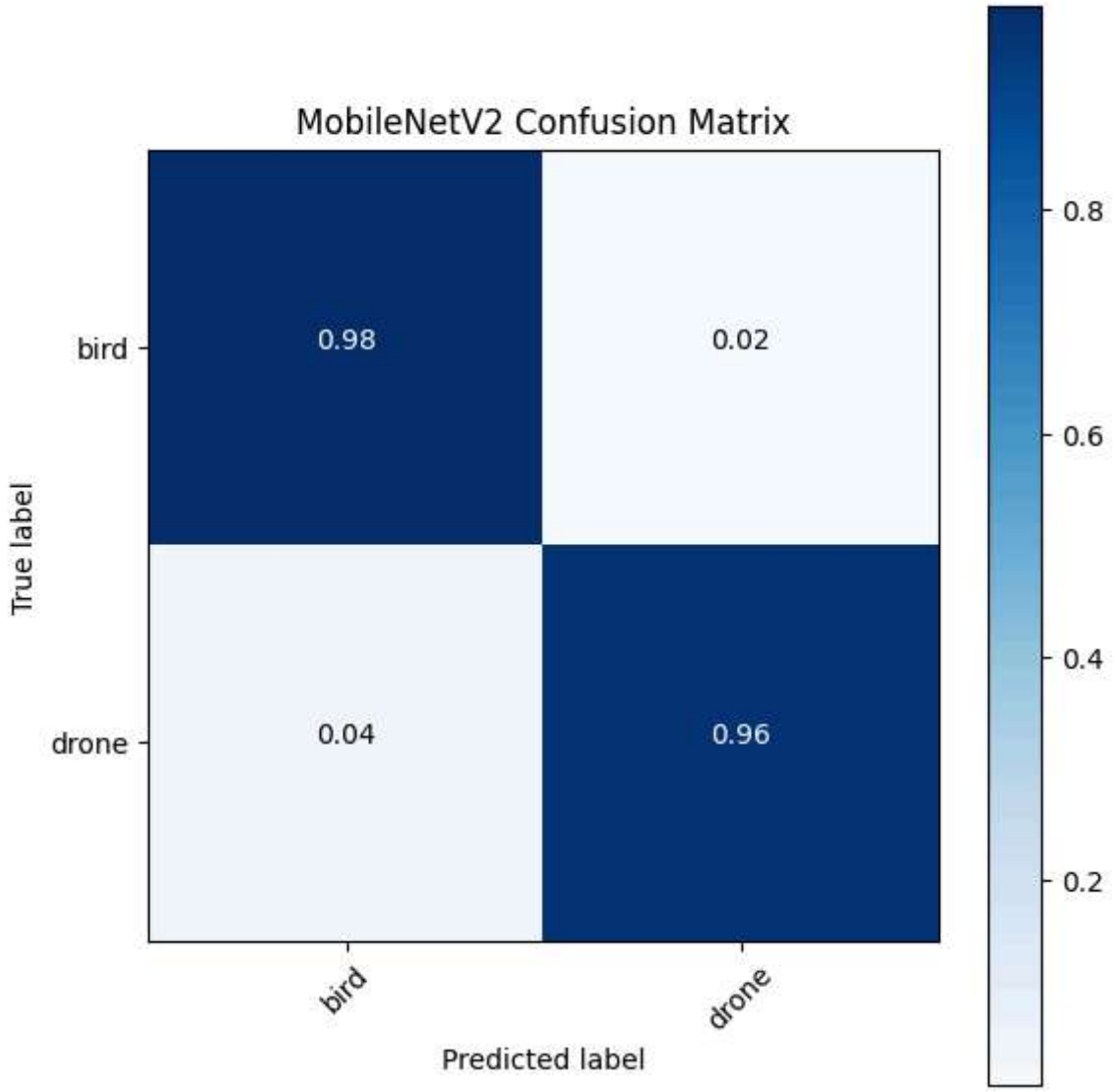
In [16]:
```
print("Custom CNN Classification Report:")
print(classification_report(y_true, y_pred_custom, target_names=class_names))

cm_custom = confusion_matrix(y_true, y_pred_custom)
plot_confusion_matrix(cm_custom, classes=class_names, title="Custom CNN Confusion Matrix", normalize=True)
```

```
Custom CNN Classification Report:
              precision    recall  f1-score   support

        bird       0.83      0.92      0.87       121
       drone       0.88      0.77      0.82        94

    accuracy                           0.85       215
   macro avg       0.86      0.84      0.85       215
weighted avg       0.85      0.85      0.85       215
```



In [17]:
```
print("\nMobileNetV2 Classification Report:")
print(classification_report(y_true, y_pred_mobilenet, target_names=class_names))

cm_mobilenet = confusion_matrix(y_true, y_pred_mobilenet)
plot_confusion_matrix(cm_mobilenet, classes=class_names, title="MobileNetV2 Confusion Matrix", normalize=True)
```

```
MobileNetV2 Classification Report:
              precision    recall  f1-score   support

        bird       0.97      0.98      0.98       121
       drone       0.98      0.96      0.97        94

    accuracy                           0.97       215
   macro avg       0.97      0.97      0.97       215
weighted avg       0.97      0.97      0.97       215
```



In [18]:
```
from sklearn.metrics import accuracy_score
import pandas as pd
# Compute overall accuracies
acc_custom = accuracy_score(y_true, y_pred_custom)
acc_mobilenet = accuracy_score(y_true, y_pred_mobilenet)

# You can parse classification_report into dicts if you want specific averages:
```

```
report_custom = classification_report(y_true, y_pred_custom, target_names=class_names, output_dict=True)
report_mobilenet = classification_report(y_true, y_pred_mobilenet, target_names=class_names, output_dict=True)

# Macro averages
macro_custom = report_custom["macro avg"]
macro_mobilenet = report_mobilenet["macro avg"]

model_comparison = [
    {
        "Model": "Custom CNN",
        "Test Accuracy": acc_custom,
        "Precision (macro avg)": macro_custom["precision"],
        "Recall (macro avg)": macro_custom["recall"],
        "F1-score (macro avg)": macro_custom["f1-score"],
        "Training Time (min)": custom_train_minutes
    },
    {
        "Model": "MobileNetV2 (Transfer Learning)",
        "Test Accuracy": acc_mobilenet,
        "Precision (macro avg)": macro_mobilenet["precision"],
        "Recall (macro avg)": macro_mobilenet["recall"],
        "F1-score (macro avg)": macro_mobilenet["f1-score"],
        "Training Time (min)": total_mobilenet_minutes
    }
]

df_comparison = pd.DataFrame(model_comparison)
display(df_comparison)
```

	Model	Test Accuracy	Precision (macro avg)	Recall (macro avg)	F1-score (macro avg)	Training Time (min)
0	Custom CNN	0.851163	0.856318	0.841656	0.846099	56.539695
1	MobileNetV2 (Transfer Learning)	0.972093	0.972870	0.970459	0.971576	22.332246

YOLOv8 Object Detection Extension (Bird vs Drone)

In addition to image classification, we prepare the pipeline for **object detection** using **YOLOv8**.
This allows the system not only to classify an image as Bird/Drone but also to **localise** the objects with bounding boxes.

Steps:

1. Install and import YOLOv8 (`ultralytics`).
2. Use the prepared YOLOv8-format dataset (`object_detection_Dataset` with `.txt` labels and `data.yaml`).
3. Train the YOLOv8 model for Bird vs Drone detection.
4. Validate the trained model.
5. Run inference on sample test images.

In [19]:
```
# If not already installed (run once in your environment)
# !pip install ultralytics

from ultralytics import YOLO
import os

# Path to YOLOv8 data config (update the path if data.yaml is elsewhere)
DATA_YOLO = r"C:/Users/Shyam/Documents/Labmentix/Project2/Aerial Objection Classification/serialbirddroneProjectV1/da

# Check that file exists
print("YOLO data.yaml exists:", os.path.exists(DATA_YOLO))
print("Using data config:", DATA_YOLO)

# ----------------------------
# 1. Load a base YOLOv8 model
# ----------------------------
# You can choose yolov8n.pt (nano), yolov8s.pt (small), etc.
yolo_model = YOLO("yolov8n.pt")

# ----------------------------
# 2. Train YOLOv8
# ----------------------------
# NOTE: This is compute-heavy. Run on GPU if possible.
yolo_model.train(
    data=DATA_YOLO,
    imgsz=640,
    epochs=50,
    batch=16,
```

```
    name="bird_drone_yolov8",
    project="runs/detect"
)
```

```
YOLO data.yaml exists: True
Using data config: C:/Users/Shyam/Documents/Labmentix/Project2/Aerial Objection Classification/serialbirddroneProject
V1/data/object_detection_Dataset/data.yaml
```

```
New https://pypi.org/project/ultralytics/8.3.232 available  Update with 'pip install -U ultralytics'
Ultralytics YOLOv8.0.196  Python-3.10.10 torch-2.1.0+cpu CPU (12th Gen Intel Core(TM) i7-1255U)
engine\trainer: task=detect, mode=train, model=yolov8n.pt, data=C:/Users/Shyam/Documents/Labmentix/Project2/Aerial Ob
jection Classification/serialbirddroneProjectV1/data/object_detection_Dataset/data.yaml, epochs=50, patience=50, batc
h=16, imgsz=640, save=True, save_period=-1, cache=False, device=None, workers=8, project=runs/detect, name=bird_drone
_yolov8, exist_ok=False, pretrained=True, optimizer=auto, verbose=True, seed=0, deterministic=True, single_cls=False,
rect=False, cos_lr=False, close_mosaic=10, resume=False, amp=True, fraction=1.0, profile=False, freeze=None, overlap_
mask=True, mask_ratio=4, dropout=0.0, val=True, split=val, save_json=False, save_hybrid=False, conf=None, iou=0.7, ma
x_det=300, half=False, dnn=False, plots=True, source=None, show=False, save_txt=False, save_conf=False, save_crop=Fal
se, show_labels=True, show_conf=True, vid_stride=1, stream_buffer=False, line_width=None, visualize=False, augment=Fa
lse, agnostic_nms=False, classes=None, retina_masks=False, boxes=True, format=torchscript, keras=False, optimize=Fals
e, int8=False, dynamic=False, simplify=False, opset=None, workspace=4, nms=False, lr0=0.01, lrf=0.01, momentum=0.937,
weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=7.5, cls=0.5, dfl=1.5, pose=12.
0, kobj=1.0, label_smoothing=0.0, nbs=64, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, s
hear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy_paste=0.0, cfg=None, tracker=botsort.y
aml, save_dir=runs\detect\bird_drone_yolov87
Overriding model.yaml nc=80 with nc=2

                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  1     37248  ultralytics.nn.modules.block.C2f             [192, 64, 1]
 16                  -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    123648  ultralytics.nn.modules.block.C2f             [192, 128, 1]
 19                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 22        [15, 18, 21]  1    751702  ultralytics.nn.modules.head.Detect           [2, [64, 128, 256]]
```

```
Model summary: 225 layers, 3011238 parameters, 3011222 gradients, 8.2 GFLOPs

Transferred 319/355 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs\detect\bird_drone_yolov87', view at http://localhost:6006/
Freezing layer 'model.22.dfl.conv.weight'
train: Scanning C:\Users\Shyam\Documents\Labmentix\Project2\Aerial Objection Classification\serialbirddroneProjectV1
\da
val: Scanning C:\Users\Shyam\Documents\Labmentix\Project2\Aerial Objection Classification\serialbirddroneProjectV1\da
ta
Plotting labels to runs\detect\bird_drone_yolov87\labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr0' a
nd 'momentum' automatically...
optimizer: AdamW(lr=0.001667, momentum=0.9) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.0005), 63 b
ias(decay=0.0)
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to runs\detect\bird_drone_yolov87
Starting training for 50 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       1/50         0G      1.443      2.352      1.606         19        640: 100%|██████████| 171/171 [19:26<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
22
                   all        448        663      0.399      0.387      0.331      0.151

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       2/50         0G      1.586      2.124      1.726         13        640: 100%|██████████| 171/171 [18:10<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663      0.273       0.32      0.227     0.0758

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       3/50         0G      1.658       2.07      1.792         28        640: 100%|██████████| 171/171 [17:59<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
05
                   all        448        663      0.366      0.388      0.324      0.137

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       4/50         0G      1.649      2.003      1.773         21        640: 100%|██████████| 171/171 [18:00<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
05
                   all        448        663      0.301       0.29      0.189     0.0794
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       5/50         0G      1.614      1.918      1.752         23        640: 100%|██████████| 171/171 [17:57<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
08
                   all        448        663       0.32      0.333      0.239      0.108

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       6/50         0G      1.571      1.786      1.715         37        640: 100%|██████████| 171/171 [17:58<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663      0.542       0.51      0.516      0.237

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       7/50         0G      1.531      1.713      1.682         20        640: 100%|██████████| 171/171 [17:56<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663      0.706      0.544      0.625      0.303

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       8/50         0G      1.504      1.624      1.654         21        640: 100%|██████████| 171/171 [17:54<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
06
                   all        448        663      0.674      0.573      0.614      0.331

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       9/50         0G      1.468      1.578      1.631         29        640: 100%|██████████| 171/171 [17:55<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
07
                   all        448        663      0.513      0.195      0.202      0.101

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      10/50         0G      1.431      1.511      1.596         13        640: 100%|██████████| 171/171 [17:59<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663      0.651       0.58       0.62      0.336

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      11/50         0G      1.424      1.508      1.597         20        640: 100%|██████████| 171/171 [18:01<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663       0.78      0.625      0.692      0.386
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      12/50         0G      1.393      1.462      1.576         18        640: 100%|██████████| 171/171 [17:57<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
05
                   all        448        663      0.781      0.649      0.718      0.399

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      13/50         0G      1.386       1.44      1.575         20        640: 100%|██████████| 171/171 [17:53<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663      0.693      0.652      0.692      0.403

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      14/50         0G      1.373      1.369      1.564         16        640: 100%|██████████| 171/171 [17:56<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663      0.735      0.613      0.676      0.385

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      15/50         0G      1.355      1.352      1.543          9        640: 100%|██████████| 171/171 [17:57<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663       0.74       0.63      0.696      0.411

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      16/50         0G       1.34      1.297      1.526         33        640: 100%|██████████| 171/171 [17:56<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
05
                   all        448        663       0.83      0.658      0.752      0.446

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      17/50         0G      1.336      1.297       1.53         16        640: 100%|██████████| 171/171 [17:53<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
05
                   all        448        663      0.786      0.692      0.754      0.433

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      18/50         0G      1.308      1.282      1.522         18        640: 100%|██████████| 171/171 [17:55<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663      0.802      0.647      0.721      0.429
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      19/50         0G      1.291      1.219      1.492         16        640: 100%|██████████| 171/171 [17:55<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663      0.805      0.669      0.749      0.444

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      20/50         0G      1.275      1.213      1.497         17        640: 100%|██████████| 171/171 [17:54<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
05
                   all        448        663      0.789      0.692      0.758      0.437

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      21/50         0G      1.254      1.175      1.465         14        640: 100%|██████████| 171/171 [17:52<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
05
                   all        448        663      0.801      0.717      0.766       0.45

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      22/50         0G      1.262      1.163       1.46         29        640: 100%|██████████| 171/171 [17:58<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663      0.812      0.698      0.762      0.467

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      23/50         0G      1.257      1.145      1.456         15        640: 100%|██████████| 171/171 [17:54<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663      0.795      0.704      0.752      0.469

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      24/50         0G      1.223      1.112       1.44         21        640: 100%|██████████| 171/171 [17:56<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
05
                   all        448        663      0.796      0.718      0.781      0.462

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      25/50         0G      1.218      1.115      1.436         25        640: 100%|██████████| 171/171 [17:55<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
05
                   all        448        663      0.857      0.726      0.797      0.479
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      26/50         0G      1.213      1.091      1.433         16        640: 100%|██████████| 171/171 [17:53<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663      0.823      0.695      0.771      0.478

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      27/50         0G      1.189      1.065      1.414         14        640: 100%|██████████| 171/171 [17:56<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663      0.766      0.702      0.757       0.47

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      28/50         0G       1.18      1.036      1.412         19        640: 100%|██████████| 171/171 [17:56<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663      0.838       0.72      0.788      0.481

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      29/50         0G      1.169       1.01       1.41         16        640: 100%|██████████| 171/171 [17:51<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
05
                   all        448        663      0.821      0.721      0.792      0.479

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      30/50         0G      1.147      1.002      1.392         41        640: 100%|██████████| 171/171 [17:59<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
04
                   all        448        663      0.809      0.701      0.753      0.463

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      31/50         0G      1.132      0.984      1.379         25        640: 100%|██████████| 171/171 [18:19<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
06
                   all        448        663      0.851      0.716      0.792      0.504

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      32/50         0G      1.124     0.9606      1.373         26        640: 100%|██████████| 171/171 [18:39<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
07
                   all        448        663      0.877      0.697      0.791       0.48
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      33/50         0G      1.137     0.9599      1.381         24        640: 100%|██████████| 171/171 [18:41<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
07
                   all        448        663      0.823      0.751      0.804      0.493

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      34/50         0G      1.104     0.9293      1.342         28        640: 100%|██████████| 171/171 [18:39<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
07
                   all        448        663      0.873      0.724      0.805      0.507

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      35/50         0G      1.103     0.9383      1.369         23        640: 100%|██████████| 171/171 [19:07<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
23
                   all        448        663      0.858      0.738        0.8      0.503

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      36/50         0G      1.078      0.902      1.342         31        640: 100%|██████████| 171/171 [20:21<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
07
                   all        448        663      0.866      0.722      0.791      0.484

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      37/50         0G      1.071      0.906      1.337         22        640: 100%|██████████| 171/171 [18:36<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
07
                   all        448        663      0.855       0.72      0.798       0.49

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      38/50         0G      1.069     0.8799       1.32         27        640: 100%|██████████| 171/171 [18:40<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
07
                   all        448        663      0.858      0.735      0.803      0.505

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      39/50         0G      1.053     0.8827      1.322         23        640: 100%|██████████| 171/171 [18:35<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
06
                   all        448        663      0.841      0.755      0.817       0.52
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      40/50         0G      1.046     0.8641      1.311         22        640: 100%|██████████| 171/171 [18:40<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
06
                   all        448        663      0.845      0.757       0.81      0.517
Closing dataloader mosaic

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      41/50         0G     0.9683     0.6788      1.294          8        640: 100%|██████████| 171/171 [18:30<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
07
                   all        448        663      0.869      0.731      0.816      0.513

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      42/50         0G     0.9251     0.6256      1.268         18        640: 100%|██████████| 171/171 [18:29<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
07
                   all        448        663      0.895       0.73      0.825      0.524

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      43/50         0G     0.9062     0.6191      1.251         13        640: 100%|██████████| 171/171 [18:28<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
14
                   all        448        663      0.868      0.745      0.823      0.531

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      44/50         0G     0.8937     0.5938      1.242         27        640: 100%|██████████| 171/171 [20:30<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
18
                   all        448        663      0.874      0.732      0.827      0.522

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      45/50         0G     0.8771     0.5697      1.219          9        640: 100%|██████████| 171/171 [18:24<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
07
                   all        448        663      0.857      0.745      0.819      0.528

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      46/50         0G     0.8552     0.5557      1.209         11        640: 100%|██████████| 171/171 [18:25<00:00,
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
07
```

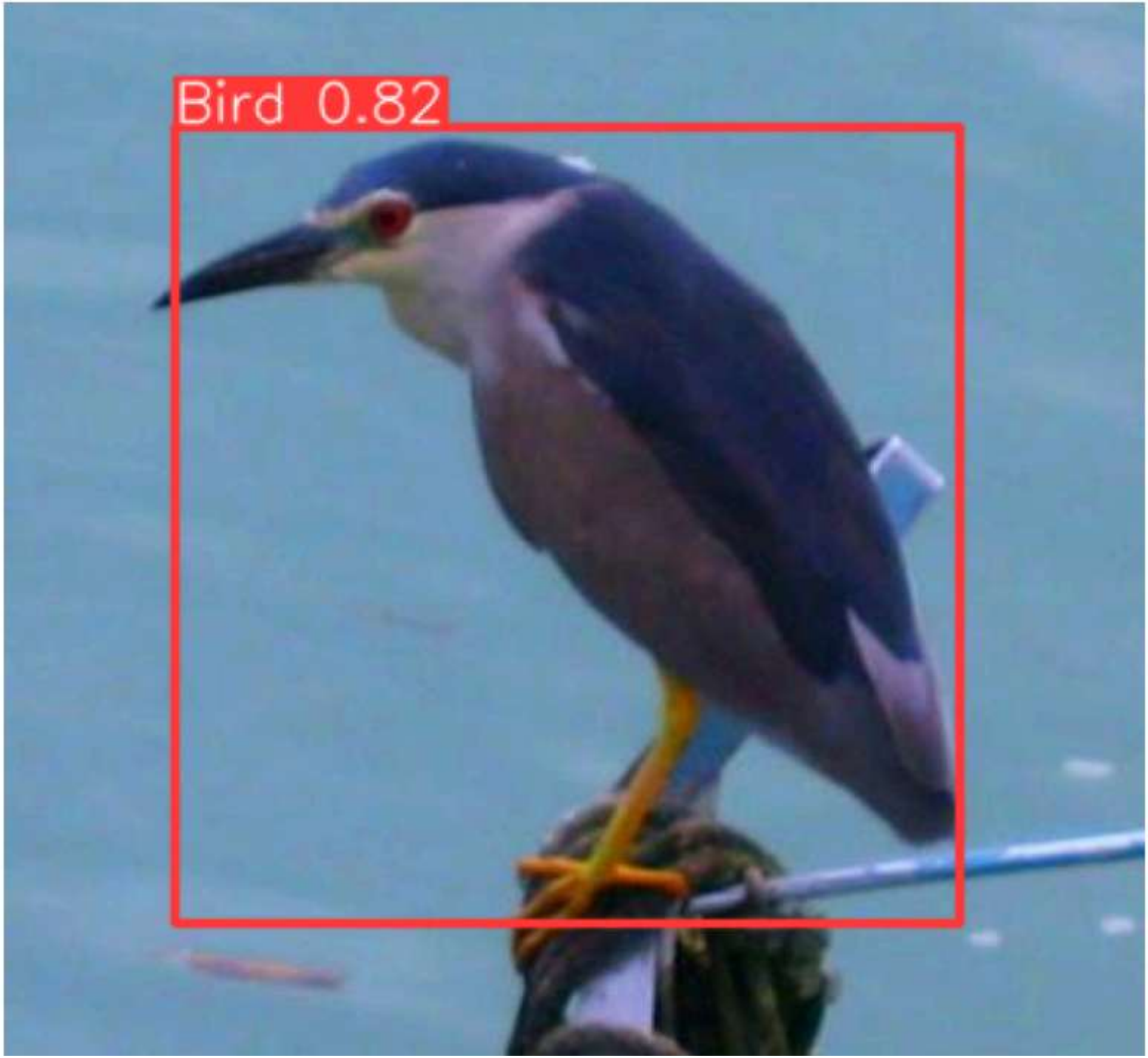
```
                 all        448        663       0.91      0.713      0.816      0.522

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      47/50         0G     0.8306     0.5368      1.188          8        640: 100%|██████████| 171/171 [18:26<00:00,
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
06
                 all        448        663      0.833      0.761      0.818       0.52

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      48/50         0G     0.8126     0.5334      1.184         12        640: 100%|██████████| 171/171 [18:26<00:00,
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
07
                 all        448        663      0.866      0.745       0.82      0.525

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      49/50         0G     0.8077     0.5242      1.181         14        640: 100%|██████████| 171/171 [18:27<00:00,
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
08
                 all        448        663      0.892      0.739      0.826       0.53

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      50/50         0G     0.7983     0.5109      1.168          9        640: 100%|██████████| 171/171 [18:26<00:00,
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [01:
07
                 all        448        663      0.883      0.745      0.824      0.532

50 epochs completed in 16.190 hours.
Optimizer stripped from runs\detect\bird_drone_yolov87\weights\last.pt, 6.3MB
Optimizer stripped from runs\detect\bird_drone_yolov87\weights\best.pt, 6.3MB

Validating runs\detect\bird_drone_yolov87\weights\best.pt...
Ultralytics YOLOv8.0.196  Python-3.10.10 torch-2.1.0+cpu CPU (12th Gen Intel Core(TM) i7-1255U)
Model summary (fused): 168 layers, 3006038 parameters, 0 gradients, 8.1 GFLOPs
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 14/14 [00:
58
                 all        448        663      0.883      0.743      0.824      0.533
                Bird        448        414      0.857      0.622      0.721      0.417
               Drone        448        249      0.909      0.863      0.927      0.649
Speed: 2.7ms preprocess, 112.8ms inference, 0.0ms loss, 0.6ms postprocess per image
Results saved to runs\detect\bird_drone_yolov87
```

Out[19]:
```
ultralytics.utils.metrics.DetMetrics object with attributes:

ap_class_index: array([0, 1])
box: ultralytics.utils.metrics.Metric object
confusion_matrix: <ultralytics.utils.metrics.ConfusionMatrix object at 0x000001267071E650>
fitness: 0.5620609442655043
keys: ['metrics/precision(B)', 'metrics/recall(B)', 'metrics/mAP50(B)', 'metrics/mAP50-95(B)']
maps: array([    0.41684,     0.64907])
names: {0: 'Bird', 1: 'Drone'}
plot: True
results_dict: {'metrics/precision(B)': 0.882820508623679, 'metrics/recall(B)': 0.7429083522202433, 'metrics/mAP50
(B)': 0.8240311895383026, 'metrics/mAP50-95(B)': 0.5329531392351934, 'fitness': 0.5620609442655043}
save_dir: WindowsPath('runs/detect/bird_drone_yolov87')
speed: {'preprocess': 2.7467720210552216, 'inference': 112.77645719902856, 'loss': 0.0, 'postprocess': 0.64591903771
67293}
```

In [22]:
```
# ----------------------------
# 3. Validate the trained model
# ----------------------------
yolo_model.val()

# Show annotated image
from matplotlib import pyplot as plt
# Ensure inline display
%matplotlib inline
# ----------------------------
# 4. Run inference on a sample image
# ----------------------------
# Update this path to one image from your test set
test_image_path = "data/object_detection_Dataset/test/images/00ddd713b6f26a17_jpg.rf.0756de13413d353595f8a65e9300b032

if os.path.exists(test_image_path):
    results = yolo_model(test_image_path)
    # Show result in notebook (opens image viewer in some environments)

    # YOLOv8 returns annotated image in BGR → convert to RGB

    annotated_img = results[0].plot()[:, :, ::-1]

    plt.figure(figsize=(10,10))
    plt.imshow(annotated_img)
    plt.axis("off")
```

```
    plt.show()
else:
    print("Test image not found at:", test_image_path)
```

```
Ultralytics YOLOv8.0.196  Python-3.10.10 torch-2.1.0+cpu CPU (12th Gen Intel Core(TM) i7-1255U)
val: Scanning C:\Users\Shyam\Documents\Labmentix\Project2\Aerial Objection Classification\serialbirddroneProjectV1\da
ta
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|████████| 28/28 [00:
55
                   all        448        663      0.883      0.743      0.824      0.533
                  Bird        448        414      0.857      0.622      0.721      0.417
                 Drone        448        249      0.909      0.863      0.927      0.649
Speed: 2.2ms preprocess, 108.7ms inference, 0.0ms loss, 0.7ms postprocess per image
Results saved to runs\detect\bird_drone_yolov810

image 1/1 C:\Users\Shyam\Documents\Labmentix\Project2\Aerial Objection Classification\serialbirddroneProjectV1\data\o
bject_detection_Dataset\test\images\00ddd713b6f26a17_jpg.rf.0756de13413d353595f8a65e9300b032.jpg: 640x640 1 Bird, 24
9.4ms
Speed: 8.0ms preprocess, 249.4ms inference, 2.0ms postprocess per image at shape (1, 3, 640, 640)
```


Bird 0.82

YOLOv8 Results & Notes

- The YOLOv8 model is trained on the same Bird/Drone dataset in **object detection** mode using YOLO annotations.
- This extends the solution from **classification** (single label per image) to **detection** (multiple objects + bounding boxes).
- The trained weights (e.g., `runs/detect/bird_drone_yolov8/weights/best.pt`) are exported and later used in the **Streamlit app** to optionally display detection results with bounding boxes.

Model Evaluation & Final Selection

- Both the **Custom CNN** and **MobileNetV2** models were evaluated on an unseen test set using key performance metrics such as **Accuracy**, **Precision**, **Recall**, and **F1-score**.
- The **fine-tuned MobileNetV2 model** demonstrated superior performance, showing higher recall for the Drone class and a more balanced confusion matrix compared to the Custom CNN.
- Due to its better generalisation capability, faster convergence, and stable validation metrics, the **fine-tuned MobileNetV2** model was selected as the **final model for deployment**.
- This model is used in the Streamlit app to make live predictions for Bird vs Drone classification.

Class Imbalance & Statistical Tests

- The dataset is reasonably balanced between Birds (1752 images) and Drones (1567 images), so no explicit resampling or class-weighting was needed.
- Since this is an image classification problem, model performance was evaluated using classification metrics (Accuracy, Precision, Recall, F1-score) rather than classical statistical hypothesis tests.

Conclusion & Business Recommendations

- The fine-tuned MobileNetV2 model reliably distinguishes between Birds and Drones in aerial images, making it suitable for real-time monitoring applications.
- This solution can be integrated into surveillance systems at airports, wind farms, and restricted zones to automatically flag potential drone intrusions and monitor bird activity.

In [ ]:

In [ ]: