Project Title	**Aerial Object Classification & Detection**
Skills take away From This Project	• Deep Learning • Computer Vision • Image Classification & Object Detection • Python • TensorFlow/Keras or PyTorch • Data Preprocessing & Augmentation • YOLOv8 (Optional – Object Detection) • Model Evaluation • Streamlit Deployment
Domain	**Aerial Surveillance, Wildlife Monitoring, Security & Defense Applications**

📌 Problem Statement

This project aims to develop a deep learning-based solution that can **classify** aerial images into two categories — **Bird** or **Drone** — and optionally perform **object detection** to locate and label these objects in real-world scenes.

The solution will help in **security surveillance**, **wildlife protection**, and **airspace safety** where accurate identification between drones and birds is critical. The project involves building a **Custom CNN classification model**, leveraging **transfer learning**, and optionally implementing **YOLOv8** for real-time object detection. The final solution will be deployed using **Streamlit** for interactive use.

📌 Real-Time Business Use Cases

1. **Wildlife Protection**
    - Detect birds near wind farms or airports to prevent accidents.
2. **Security & Defense Surveillance**
    - Identify drones in restricted airspace for timely alerts.
3. **Airport Bird-Strike Prevention**
    - Monitor runway zones for bird activity.
4. **Environmental Research**
    - Track bird populations using aerial footage without misclassification.

📌 Project Workflow

1. Understand the Dataset

- Inspect dataset folder structure
- Check number of images per class
- Identify class imbalance
- Visualize sample images

---

2. Data Preprocessing

- Normalize pixel values to **[0, 1]**
- Resize images to a fixed size (**224×224** for classification)

---

3. Data Augmentation

- Apply transformations: rotation, flipping, zoom, brightness, cropping

---

4. Model Building (Classification)

- **Custom CNN:** Conv layers, pooling, dropout, batch normalization, dense output layer
- **Transfer Learning:** Load models like ResNet50, MobileNet, EfficientNetB0 and fine-tune

---

5. Model Training

- Train both models
- Use EarlyStopping & ModelCheckpoint
- **Track metrics:** Accuracy, Precision, Recall, F1-score

---

6. Model Evaluation

- Evaluate test results with confusion matrix & classification report
- Plot accuracy/loss graphs

---

7. Model Comparison

- Compare accuracy, training time, and generalization performance
- Save the best performing model for Streamlit deployment

📌 Optional: Object Detection with YOLOv8

Steps:

1. Install YOLOv8.
2. Prepare dataset (images and YOLOv8-format `.txt` labels — already done).
3. Create a `data.yaml` configuration file for YOLOv8.
4. Train the YOLOv8 model.
5. Validate the trained model.
6. Run inference on test or new images.

📌 Streamlit Deployment

- Create a simple UI with image upload
- Display prediction (Bird / Drone) & confidence score
- (Optional) Show YOLOv8 detection results with bounding boxes

---

📌 Project Deliverables

1. Trained models (Custom CNN, Transfer Learning, YOLOv8 (optional))
2. Streamlit app for classification/detection
3. Scripts & notebooks for preprocessing, training, evaluation
4. Model comparison report

5. GitHub repository with documentation

6. Well-structured, commented code
7. Video

🛠️ **Technical Tags**

`Computer Vision, Deep Learning, Image Classification, Object Detection, CNN, YOLOv8, Transfer Learning, Data Augmentation, Model Evaluation, Streamlit Deployment, Aerial Surveillance AI`

📌 Datasets

📌 Classification Dataset

- **Source:** classification_dataset
- **Task:** Image Classification (Binary: Bird / Drone)
- **Data Type:** RGB Images
- **Format:** `.jpg`

Structure

- `TRAIN set:`
  - `- bird: 1414 images`
  - `- drone: 1248 images`
- `VALID set:`
  - `- bird: 217 images`
  - `- drone: 225 images`
- `TEST set:`
  - `- bird: 121 images`
  - `- drone: 94 images`

📌 Object Detection Dataset (YOLOv8 Format)

- **Source :** object_detection_Dataset
- The dataset contains 3319 images with corresponding YOLOv8-format annotations (`.txt` files).
- Each annotation file contains bounding boxes in the format:

```
<class_id> <x_center> <y_center> <width> <height>
```

- Data split: Train (2662), Validation (442), Test (215).

⏳ Timeline

The project should be completed and submitted **within 14 days** from the date it is assigned.

Reference

Streamlit recording (English)	Special session for STREAMLIT(11/08/2024)
Streamlit Reference doc	Streamlit API reference
Project Live Evaluation	Project Live Evaluation
Capstone Explanation Guideline	Capstone Explanation Guideline
GitHub Reference	How to Use GitHub.pptx

Deep learning material	Deep_Learning-study_material.pdf