**Military Soldier Safety and Weapon Detection using YOLO and Computer Vision**

**📌 Table of Contents**

1. **Project Overview**
2. **Objectives**
3. **Dataset Description**
4. **Preprocessing & Augmentation**
5. **Model Architecture & Training**
6. **Model Evaluation**
7. **App Development using Streamlit**
8. **Results & Screenshots**
9. **Challenges Faced**
10. **Conclusion & Future Scope**
11. **References**

**🧾 1. Project Overview**

This project focuses on building an object detection system capable of identifying and localizing military assets in images using the YOLOv8 deep learning model. A web-based application was built using Streamlit to allow users to upload images and visualize detections in real time.

**🎯 2. Objectives**

* Build an object detection model for military applications.
* Train a robust model using YOLOv8 on a custom dataset.
* Evaluate the model performance using appropriate metrics.
* Deploy a user-friendly web application for real-time object detection.
* Enable detection of 10 military-related object classes.

**🗂️ 3. Dataset Description**

* **Original dataset** had 12 classes:  
  camouflage\_soldier, weapon, military\_tank, military\_truck, military\_vehicle, civilian, soldier, civilian\_vehicle, military\_artillery, trench, military\_aircraft, military\_warship.
* **Final training classes (after removing small categories)**:  
  camouflage\_soldier, weapon, military\_tank, military\_truck, military\_vehicle, soldier, civilian\_vehicle, military\_artillery, military\_aircraft, military\_warship.
* **Annotation Format**: YOLO format (bounding box + class).
* **Sample Class Distribution:**

| **Class** | **Count** |
| --- | --- |
| military\_tank | 20059 |
| soldier | 7807 |
| camouflage\_soldier | 5376 |
| military\_aircraft | 8636 |
| weapon | 1568 |
| ... | ... |

**🛠️ 4. Preprocessing & Augmentation**

* **Data cleaning**: Removed very low-frequency classes (e.g., civilian, trench).
* **Augmentation Techniques**:
  + Horizontal/vertical flip
  + Random rotation
  + Brightness/contrast adjustments
  + Mosaic augmentation (YOLO-native)
* **Train/Validation Split**: 80/20

**🧠 5. Model Architecture & Training**

* **Model Used**: YOLOv8 Medium (yolov8m.pt) as base.
* **Framework**: Ultralytics YOLOv8.
* **Custom Training**: Using PyTorch backend via Ultralytics library.
* **Training Duration**: 50 epochs
* **Batch Size**: 16
* **Learning Rate**: 0.001 (default, cosine scheduler)
* **Image Size**: 640x640

**📊 6. Model Evaluation**

* **Performance Metrics**:
  + [mAP@0.5](mailto:mAP@0.5) 🡪 0.778
  + Precision 🡪 0.804
  + Recall 🡪 0.723
  + Confusion Matrix
  + PR Curve

**🧾 Key Evaluation Outputs:**

**A graph of a graph

AI-generated content may be incorrect.**

A screenshot of a computer

AI-generated content may be incorrect.

A graph of different colored lines

AI-generated content may be incorrect.

The model shows strong performance on high-occurrence classes like military\_tank and soldier, with acceptable precision-recall balance across all classes.

**Class-wise mAP@0.5:**

| **Class** | **mAP@0.5** |
| --- | --- |
| camouflage\_soldier | 0.856 |
| weapon | 0.705 |
| military\_tank | 0.894 |
| military\_truck | 0.751 |
| military\_vehicle | 0.657 |
| soldier | 0.821 |
| civilian\_vehicle | 0.515 |
| military\_artillery | 0.681 |
| military\_aircraft | 0.936 |
| military\_warship | 0.959 |

**💻 7. App Development using Streamlit**

A simple but interactive web application was built to allow users to upload an image and get back detection results.

**🔹 Key Features:**

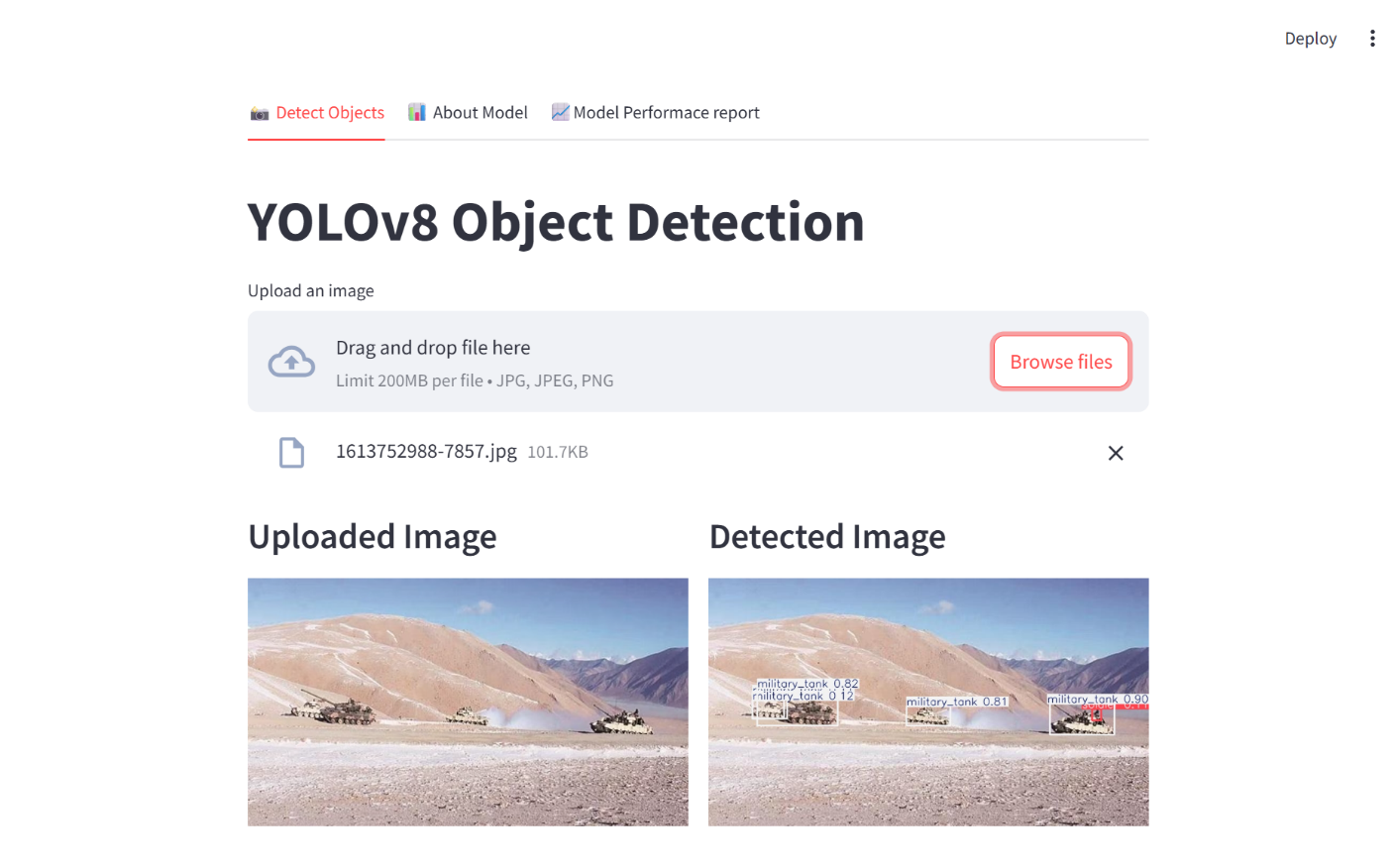
* Upload .jpg, .png images
* Visualize detections side-by-side (original vs predicted)
* Tabs for:
  + Object detection
  + Model insights
  + Evaluation metrics

**🔹 Tools Used:**

* streamlit
* ultralytics
* Pillow for image handling

**🖼️ 8. Results & Screenshots**

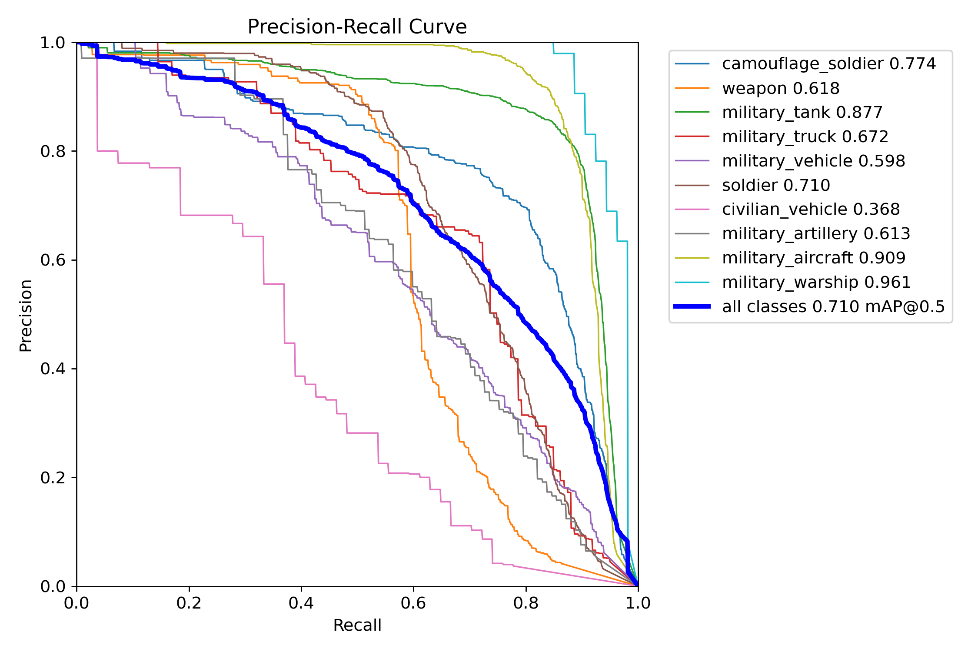
* **Uploaded Image** vs **Detected Image** examples
* Real-time detections with bounding boxes and class labels
* Preview in the app:



**🧱 9. Challenges Faced**

* Class imbalance in dataset (some classes had <100 samples).
* mAP score improvement to over 0.77 from 0.71 by improving poor performance classes

**First trial Model performance**



**Final Model performance**

A graph of different colored lines

AI-generated content may be incorrect.

**🧩 10. Conclusion & Future Scope**

**✅ Achievements:**

* Successfully trained a YOLOv8 model to detect 10 military object types.
* Deployed a real-time, user-friendly web app using Streamlit.
* Model performs well in precision and recall for major classes.

**🔮 Future Scope:**

* Add support for video stream detection.
* Deploy to the cloud (e.g., Streamlit Cloud or Hugging Face Spaces).
* Expand dataset for underrepresented classes.
* Incorporate GPS tagging or threat scoring.

**📚 11. References**

* Ultralytics YOLOv8 Documentation
* Streamlit Documentation
* Research papers and datasets on military object detection from open access sources.