Automated Classification of Military Aircraft Using Deep Learning

Submitted By:

Soumya Sharma

Table of Contents **Page No**

Project Title

Automated Classification of Military Aircraft Using Deep Learning

Our project is centered on detecting military aircraft through machine learning and computer vision techniques. It aims to identify and classify aircraft in images, enabling accurate detection of military aircraft types for applications in security, defense, and surveillance.

Technologies to be used

- **Software Platform:** Python, Kaggle Notebook
- **Front-end:** Not applicable for this model-only project
- **Hardware Platform:** Kaggle Notebook (cloud GPU), with requirements for at least 10GB of RAM for data processing.

Hardware platform:

- Ram: 16GB
- Processor: Intel Core i5

- Hard Disk: 1TB SSD

Tools

Language used:

Python: Python was used for its simplicity and powerful ecosystem of libraries that streamlined complex tasks like data handling, image processing, and machine learning. Its readable syntax allowed for rapid development and seamless integration of various tools and libraries, making it an ideal choice for building and deploying the military aircraft detection model.

Libraries used:

- **pandas**: This library was used for data manipulation, particularly for reading, processing, and managing CSV files. It provided efficient tools to handle large amounts of data with ease, allowing quick analysis and formatting, which was essential for preparing and managing labeled data for the model.

- **numpy**: As a foundational library for numerical computing in Python, numpy was used to handle arrays and perform mathematical operations on image data. It supported efficient manipulation of coordinates and bounding boxes, which are central to training and predicting with the YOLO model.

- **OpenCV (cv2)**: OpenCV is an open-source computer vision library used extensively in this project to read, process, and manipulate images. OpenCV enabled loading images, converting color spaces, and drawing bounding boxes on detected aircraft, making it essential for visualizing and processing the data.

- **matplotlib.pyplot**: This library was used for visualizing images and results. By plotting images with detected aircraft and bounding boxes, matplotlib.pyplot made it possible to verify and interpret the model's predictions, ensuring clarity in results for both training and testing phases.

- **glob**: The glob library was used for file pattern matching, which allowed efficient loading of multiple files, such as images and prediction results, from specified directories. This was helpful in handling large datasets and managing the file structure dynamically during the project.

- **os and shutil**: The os and shutil libraries were essential for creating, modifying, and organizing file directories. They supported managing project files systematically, allowing for operations like moving and copying images and creating directories as needed, which streamlined the data preparation pipeline.

- **tqdm**: tqdm added progress bars to the data processing and model training phases. This visual feedback was helpful for monitoring progress, especially during long-running tasks like data processing and model training.

- **ultralytics (YOLO)**: The ultralytics library provides an implementation of the YOLO model, a popular object detection framework. In this project, ultralytics was used for both training a YOLO model on labeled aircraft images and for performing detection on new images. This library simplified the integration of a complex deep learning model, providing tools to configure and optimize the model for military aircraft detection.

- **yaml**: The yaml library was used to manage configurations in YAML format. This was essential for defining training parameters for the YOLO model, including paths to datasets and class labels, ensuring that model configurations were clear, organized, and easily editable.

Problem Statement

The project's goal is to create a deep learning-based system capable of classifying images of military aircraft into different classes based on visual features. This system will utilize the YOLOv10 (You Only Look Once version 10) model, a state-of-the-art object detection algorithm, to analyze and categorize the aircraft images according to their types. By leveraging the advanced capabilities of YOLOv10, the system aims to achieve real-time detection and classification, which is essential for various defense and surveillance applications.

Literature Survey

1. **YOLOv10: Real-Time Object Detection**
   This paper introduces YOLOv10, focusing on improved real-time detection by removing non-maximum suppression (NMS) and optimizing model design. It features a lightweight classification head and efficient downsampling, achieving faster inference and reduced computational cost while maintaining high accuracy.

2. **Aircraft Type Recognition with YOLOv8**
   This paper evaluates YOLOv8 for aircraft type recognition, comparing different versions to find the best fit. Using a CSPDarkNet53 backbone and decoupled head, the large model variant (YOLOv8l) achieved high accuracy, proving effective in detecting fine-grained aircraft features in real-world conditions.

3. **INNAR: Aircraft Recognition Using Similarity Learning**
   This study presents a novel approach for aircraft recognition using similarity learning,

overcoming limitations of traditional classification methods in identifying unknown aircraft types. The proposed INNAR framework employs few-shot learning and a robust embedding space, enhancing recognition of both known and novel classes, especially in remote sensing applications.

Project Description

The project involves designing a robust model to classify 73 types of military aircraft i.e.[A-10, A-400M, AG-600, AH-64, AV-8B, An-124, An-22, An-225, An-72, B-1, B-2, B-21, B-52, Be-200, C-130, C-17, C-2, C-390, etc] utilizing image data to enhance real-time categorization. The system's structure includes modules for data preprocessing, model design, and testing. The high-level context diagram illustrates data flow from input to classification output.

Project Modules: Design/Algorithm

The core implementation of our project integrates several steps and methodologies to detect and classify military aircraft from images. The project relies on machine learning and object detection techniques**.**

- **Data Preprocessing**: The dataset contains images and CSV files with bounding box annotations. We extract the bounding box coordinates, normalize them, and save them in YOLO format. The dataset is split into training and validation sets, and images are organized for easy access during model training. Libraries like pandas and shutil are used for data manipulation and file organization.
- **Model Training**: We train the YOLO (You Only Look Once) model for aircraft detection using the training dataset, validating with a separate validation set. The ultralytics YOLOv10 implementation is used for training, with hyperparameters like learning rate and image size tuned for performance. The model is trained and saved locally, ready for future use but not yet deployed.
- **Prediction and Inference**: After training, the model is used to make predictions on new images. The model identifies aircraft by predicting bounding boxes and class labels. Results are visualized by overlaying bounding boxes on the images using cv2 and matplotlib.
- **Model Status**: The model is trained and ready for deployment but has not been exported or deployed yet. It remains in its trained state, ready for integration into future applications.

Implementation Methodology

The military aircraft detection project follows a structured methodology, from data preprocessing to model evaluation. This section outlines each stage, detailing the tools, technologies, and techniques used to build an effective detection system.

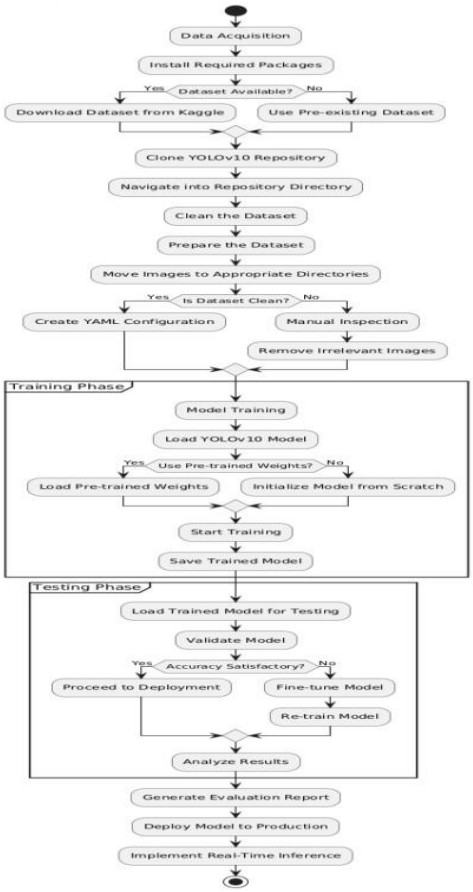
1. **Data Collection and Preprocessing**
    - **Objective**: Prepare the dataset for model training by extracting and organizing images and annotations.
    - **Process**: The dataset consists of images of military aircraft and corresponding CSV files with bounding box annotations. In this stage, the CSV files are parsed, and bounding box coordinates are normalized based on image dimensions. The dataset is then split into training and validation sets using a 90/10 ratio. The images are moved to appropriate directories for model training. Python libraries like pandas and shutil are used for data manipulation and organization.
    - **Output**: A structured dataset with images and annotation files in YOLO format, ready for training.
2. **Model Selection and Training**
    - **Objective**: Train a YOLO-based object detection model to classify and localize military aircraft in images.
    - **Process**: The YOLO (You Only Look Once) algorithm is used for real-time object detection. This model is selected due to its high accuracy and speed in detecting objects in images. The model is trained using the ultralytics YOLOv10 implementation, with hyperparameters like learning rate, batch size, and image resolution tuned for optimal performance. The dataset is fed into the model, and training is conducted over a set number of epochs, with validation data used to assess performance during training.
    - **Output**: A trained YOLO model capable of detecting military aircraft in unseen images, with weights saved for future use.
3. **Model Inference and Prediction**
    - **Objective**: Use the trained model to predict aircraft in new images.
    - **Process**: After training, the model is used for inference on new images. The model predicts bounding boxes and class labels for detected aircraft. The results are

visualized by overlaying bounding boxes on the original images. The cv2 and matplotlib libraries are used to draw bounding boxes and display the results.

- **Output**: Visualized predictions showing detected aircraft with bounding boxes in the input images.

4. **Model Evaluation and Analysis**

   - **Objective**: Evaluate the performance of the trained model on the validation dataset.
   - **Process**: The model's accuracy is assessed using standard metrics for object detection, such as Intersection over Union (IoU) and mean Average Precision (mAP). These metrics help to quantify how well the model performs in terms of both detection and localization of military aircraft. The validation dataset is used to measure the model's generalization capability.
   - **Output**: Evaluation metrics that provide insight into the model's accuracy and performance.



Result & Conclusion

- **Quantitative Performance Metrics:** The YOLOv10 model was evaluated using several key performance metrics, which provide insights into its effectiveness in classifying and detecting military aircraft.

- **Accuracy, Precision, Recall, and F1-Score**: The model achieved an overall accuracy of 76.4% across all classes. Precision and recall were calculated for each aircraft type, with notable results:

- **EF2000**: Precision: 0.649, Recall: 0.620, F1-Score: 0.634

- **Tu95**: Precision: 0.926, Recall: 0.833, F1-Score: 0.877

- **F15: Precision**: 0.833, Recall: 0.706, F1-Score: 0.765

  The mean F1-score across all classes was approximately 0.687, indicating balanced performance in terms of precision and recall.

- **Intersection over Union (IoU):** The average IoU for detected bounding boxes was approximately 0.764, reflecting the model's accuracy in localizing aircraft within images.

- **Mean Average Precision (mAP):** The mAP at an IoU threshold of 0.5 was calculated to be 76.4%, while the mAP across multiple IoU thresholds (from 0.5 to 0.95) was found to be 68.7%.

- **Confusion Matrix**: Two confusion matrices were generated to visualize classification results (see Figures X and Y).

- **Raw Confusion Matrix**: This matrix displays absolute counts of predictions, where darker colors along the diagonal indicate correct classifications, while off-diagonal cells represent misclassifications.

- **Normalized Confusion Matrix**: This matrix shows the percentage of instances for each true class that were predicted as each class, helping to identify which classes are most frequently confused with one another.

  Both matrices suggest a good degree of correct classification, indicated by prominent diagonal values, but also highlight areas for improvement in reducing misclassifications among certain aircraft types.

- **Training and Validation Losses:** The training process involved monitoring various loss metrics over 100 epochs (see Figures Z1-Z4):

- **Training Losses:**

Box regression loss decreased steadily, indicating improved localization.

Classification loss also showed a downward trend, reflecting better object classification.

Distribution Focal Loss (DFL) decreased as well, suggesting enhanced bounding box regression.

- **Validation Losses**:

Validation box loss followed a similar trend to training box loss, indicating good generalization.

Validation classification loss decreased consistently, further confirming model performance.

- **Performance Metrics Over Epochs**: Performance metrics such as precision and recall improved over epochs (see Figures A1-A2). The mean Average Precision (mAP) at various IoU thresholds showed a steady increase throughout training:

  mAP@50 reached an impressive value of approximately 76.4%.

  mAP@50-95 also increased, indicating robust performance across varying levels of overlap.

- **Detection Examples**: Qualitative results were illustrated through sample images showcasing both successful detections and failure cases (see Figures B1-BN). The model accurately classified and localized various aircraft in several instances but struggled with smaller or overlapping objects.

- **Error Analysis**: The primary types of errors encountered included:

  - Misclassifications between visually similar aircraft types.
  - Challenges with detecting smaller or partially obscured aircraft.

  These errors suggest potential areas for further refinement in the model's architecture or training data augmentation strategies.

- **Computational Efficiency**: The model demonstrated efficient inference capabilities, processing images at an average speed of approximately 3.8 ms per image, making it suitable for real-time applications in military contexts.

Summary of Key Findings

In summary, the YOLOv10 model exhibited strong performance in classifying and detecting military aircraft, achieving high precision and recall rates across various classes. While the average mAP and IoU scores indicate effective localization capabilities, challenges remain with certain

misclassifications and detection of smaller or overlapping objects. The model weights were manually saved as best_model.pt, ensuring that the trained parameters can be reused for future inference or further training.

Future Scope and further enhancement of the Project

- **Real-time Detection**: Implementing real-time detection using frameworks like TensorFlow Lite or YOLOv8 would enable deployment on drones or surveillance cameras, allowing for live monitoring of airspace.
- **Multi-view Recognition**: Enhancing the model to detect aircraft from multiple perspectives would improve accuracy. This can be achieved with multi-view datasets or 3D detection models.
- **Threat Analysis Integration**: Integrating a threat assessment system to classify detected aircraft by risk level would be valuable for defense applications, providing critical insights in real time.
- **Model Optimization for Edge Devices**: Streamlining the model for low-power edge devices, like NVIDIA Jetson, would allow deployment in remote or mobile environments, extending its usability.
- **Adaptability for Civilian Applications**: With modifications, this system could be adapted for civilian aviation or wildlife monitoring, broadening its practical applications.
- **Integration with Segmentation Models**: To enhance the precision of aircraft detection, future work will explore combining the current YOLOv10 model with a segmentation model. This integration would allow for more detailed object delineation, distinguishing finer features and improving performance in complex backgrounds.

Advantages of this Project

- **Enhanced Security and Surveillance**: This project allows for efficient detection and identification of military aircraft, improving airspace surveillance and security. Defense organizations can use it to monitor restricted zones, detect unauthorized aircraft, and respond to potential threats more effectively.
- **Real-time Threat Assessment**: With its potential for real-time deployment, the project enables timely detection and tracking of military aircraft, crucial for proactive threat assessment and response in sensitive areas.

- **Automated and Accurate Detection**: The object detection model provides high accuracy in identifying different aircraft types, reducing human error in monitoring tasks. This automation supports defense personnel by offering reliable and continuous tracking without manual oversight.
- **Broad Application Scope**: Beyond military applications, the system can be adapted for civilian uses such as air traffic control, aviation monitoring, and wildlife protection, making it versatile for multiple sectors.
- **Resource Efficiency**: By optimizing the model for edge devices, this project allows for efficient use of computational resources, making it cost-effective and suitable for deployment on mobile or remote systems.

References

1. Chen, H., Liu, L., Han, J., & Ding, G. (2024). YOLOv10: Real-Time End-to-End Object Detection. 38th Conference on Neural Information Processing Systems (NeurIPS). This work introduces YOLOv10, an advanced model for real-time object detection, emphasizing efficiency and reduced inference latency [38].

2. Saeed, A., Atif, H. B., Habib, U., & Bilal, M. (2024). Intelligent Known and Novel Aircraft Recognition - A Shift from Classification to Similarity Learning for Combat Identification. IEEE Xplore, demonstrating a method to distinguish between known and novel aircraft using similarity learning [39].

3. Lei, C., Zeng, J., Xia, Y., & Pang, F. (2024). Aircraft Type Recognition Based on YOLOv8. Journal of Physics: Conference Series, 2787, 012047. This paper applies YOLOv8 to aircraft recognition, identifying optimal YOLOv8 versions for real-time and fine-grained detection tasks in aviation [40].