Aircraft Classification from Aerial Imagery

Computer Vision CAP 5415
Course Project Report

Glaspey, Joshua
Dept. of Electrical and Computer Engineering
University of Central Florida
Orlando, United States
jo248954@ucf.edu

Green, Alexander
Dept. of Electrical and Computer Engineering
University of Central Florida
Orlando, United States
al101178@ucf.edu

***Abstract*— This project addresses the challenge of identifying military aircraft from aerial images using computer vision (CV) techniques. For this project we created a dataset of labeled aircraft using RoboFlow [1] to demonstrate the capabilities of our model to learn and recognize different vehicles. Then we applied data augmentation techniques to increase the effectiveness of our dataset and model.**

This project demonstrates the capabilities of three different CV models, YOLOv5, YOLOv8, and our own custom deep learning implementation using PyTorch. YOLOv5 [2] and YOLOv8 [3] are CV models designed and documented by Ultralytics. After successfully implementing both YOLO models, we created our own custom CV model to compare against the YOLO models which acted as a baseline for performance.

I. Introduction

The use of machine learning in computer vision is a cornerstone of artificial intelligence (AI), and recent advancements in accessible computing power have made it increasingly feasible for broader applications. As students immersed in this field, we sought to apply these principles to a practical project, demonstrating a thorough understanding of both their theoretical principles and real-world utility.

Aircraft detection in aerial imagery is a critical application for defense and surveillance. However, challenges such as image resolution, varying angles, and environmental conditions complicate accurate detection. To better reflect the complexities of real-life surveillance scenarios, we constructed a custom dataset tailored to our use case. Leveraging publicly available imagery, we carefully selected and labeled images to create a robust dataset tailored for this project.

Evaluating computer vision performance often relies on benchmarks established by existing models. To this end, we utilized two widely recognized object detection architectures, Ultralytics's YOLOv5 and YOLOv8, as baseline models. By leveraging both of these known models, we get a better understanding of not only YOLO's performance, but the amount of variance or improvement that can be expected from different CV models. These provided a solid reference for understanding the strengths and weaknesses of our own custom implementation.

This project aimed to identify the most effective method for accurate aircraft detection using realistic data. By comparing baseline models against our custom implementation, we gained valuable insights into the trade-offs between precision, recall, and accuracy in machine learning-driven computer vision tasks.

II. Model Architecture

A. *Initial Information*

Our custom model builds upon existing work in deep learning for aircraft recognition, drawing insights from the GitHub repository Deep Learning for Aircraft Recognition [4]. This repository provided an effective foundation, offering guidance on architecture and addressing some limitations found in previous CV models. However, we adapted and extended this architecture to better meet the specific needs of our project. The custom implementation incorporates a design that balances computational efficiency with detection accuracy, a necessary consideration for practical deployment in real-world scenarios.

In addition to the custom model, we also implemented YOLOv5 and YOLOv8 as baseline models to compare against our architecture. These models, known for their speed and accuracy, serve as benchmarks, providing valuable context for the performance of our custom model. We referenced the architecture and hyperparameters from the aforementioned GitHub repository but adapted them to the needs and performance of our own model.

B. *Custom Model Architecture*

Our custom model was designed with a focus on balancing computational efficiency with detection accuracy. Its architecture combines convolutional layers for feature extraction with fully connected layers for classification. The model architecture is summarized in Table 2.1 below. We kept the architecture relatively simple, using only three convolutional layers and three fully connected layers, to reduce the risk of overfitting while maintaining strong performance on our dataset.

The input to the model consists of RGB images (3 channels) with a size of 640x640 pixels. Each convolutional layer is followed by a ReLU activation function, which introduces non-linearity and allows the model to learn more complex patterns. Max-pooling layers are applied to reduce the spatial dimensions and computational cost. After the convolutional layers, the features are flattened and passed through fully connected layers, where the dimensionality is reduced to a final output layer with 5 neurons for classification, corresponding to the 5 types of aircraft in the dataset. This represents the output layer, which is used to predict the probability for each aircraft class. The highest probability neuron will be selected as the predicted label.

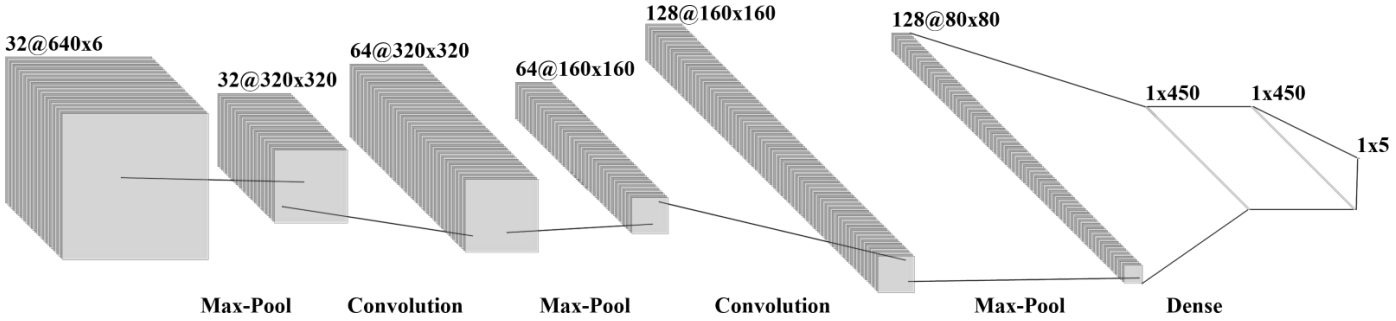

Figure 2.1

Table 2.1: Custom ConvNet Architecture

Layer (Type)	Output Shape	# of Parameters
Conv2D	bx32x640x640	896
ReLU	bx32x640x640	0
MaxPool2D	bx32x320x320	0
Conv2D	bx64x320x320	18,496
ReLU	bx64x320x320	0
MaxPool2D	bx64x160x160	0
Conv2D	bx128x160x160	73,856
ReLU	bx128x160x160	0
MaxPool2D	bx128x80x80	0
Flatten	bx819200	0
Linear (FC)	bx450	368,640,450
ReLU	bx450	0
Dropout	bx450	0
Linear (FC)	bx450	202,950
ReLU	bx450	0
Linear (FC)	bx5	2,255

Total Parameters = 368,938,903

This architecture was intentionally kept compact to avoid potential overfitting while ensuring that the model could still learn the complex patterns required for accurate aircraft detection. The decision to use three convolutional layers followed by three fully connected layers was based on a balance of accuracy and computational efficiency, ensuring that the model could generalize well while avoiding excessive training times.

III. METHOD

This section outlines the methodology used to achieve accurate aircraft detection, covering dataset preparation, data augmentation, model training, and evaluation.

We constructed a high-quality dataset and applied augmentation techniques to simulate real-world variability. YOLOv5, YOLOv8, and our custom model were trained using identical parameters for fair comparison. Performance was then evaluated using a range of metrics to assess detection accuracy and classification efficiency.

Each step is detailed below, highlighting the tools and techniques employed to develop and benchmark the models.

A. Dataset Preparation

The success of any machine learning project depends heavily on the quality and relevance of the dataset. For this project, we utilized a publicly available repository of aerial images hosted on Google Drive, which served as the foundation for constructing our dataset. These images, captured from a top-down perspective, were carefully selected to represent five distinct types of military aircraft: B-1, B-2, C-130, C-5, and E-3. These images were carefully selected to reflect real-world challenges, such as varying resolutions, angles, and environmental conditions, to ensure that our model could generalize effectively.

1) Labeling the Data

To ensure accurate and consistent annotations, we employed RoboFlow, a widely used tool for computer vision dataset management and preprocessing. RoboFlow provides an intuitive interface for labeling objects within images, which greatly simplified the process of identifying and marking aircraft in our dataset. With its automated preprocessing options, RoboFlow also allowed us to normalize image dimensions, augment the dataset for diversity, and export annotations in formats compatible with our models. We manually labeled 300 images for each aircraft type, resulting in a dataset of 1,500 images selected for this project's specific needs.

2) Dataset Splitting

To prepare the dataset for machine learning, we divided it into three distinct subsets: Training, Validation, and Testing. By creating this division, we can later use each set to train our model, evaluate its current performance as training continues, and test the final performance of the model.

- **Training Set (70%):** Comprising roughly 1,050 images, this subset was used to train the models and adjust their weights. The training set forms the backbone of the learning process, providing the data needed to identify patterns and features specific to aircraft detection.
- **Validation Set (20%):** Consisting of about 300 images, the validation set was employed during model training to evaluate intermediate performance and guide hyperparameter tuning. This subset helps to monitor overfitting and ensure that the model generalizes well to unseen data.

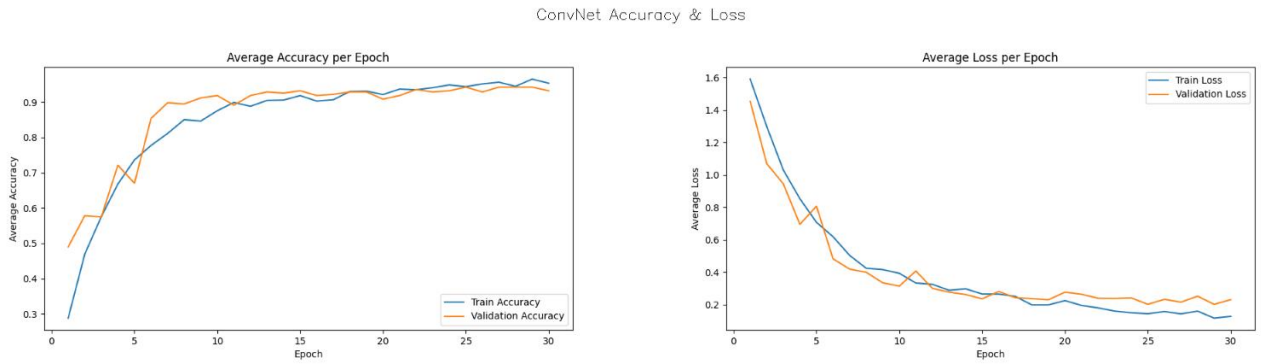

Figure 3.1

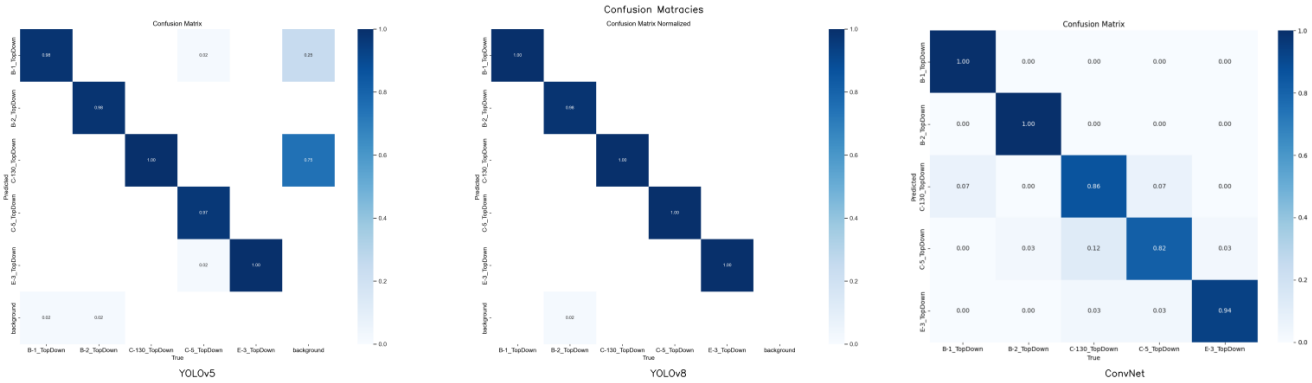

Figure 4.1

- **Testing Set (10%):** The remaining 150 images were reserved for final performance evaluation. This set provides an unbiased assessment of the model's ability to detect aircraft in new, unseen data and serves as the benchmark for comparing different models.

This structured approach to dataset preparation ensured a balanced representation of all five aircraft types across the splits, enabling fair and consistent evaluation of YOLOv5, YOLOv8, and our custom implementation. To label and split the data, we utilized RoboFlow again. RoboFlow's advanced labeling features and data splitting functionality allowed us to create our custom dataset in an incredibly small amount of time.

*3) **Data Augmentation***

To further enhance the quality and robustness of our dataset, we implemented data augmentation techniques. Data augmentation is a critical step in preparing datasets for machine learning, as it artificially expands the dataset by applying transformations to the original images. These transformations help simulate real-world variations, improving the model's ability to generalize across diverse conditions. The following augmentation techniques were applied:

- **Rotation:** To account for varying orientations of the aircraft.
- **Shearing:** To simulate changes in perspective due to camera angles.
- **Resizing:** To standardize input dimensions across the dataset.
- **Color Jitter**: To introduce slight variations in image contrast, hue, and saturation.

The purpose of data augmentation was twofold, to improve generalization, and enhance model robustness.

By introducing variability, the augmented dataset reduced the risk of overfitting, ensuring the model learned features that generalized well to unseen images.

Augmented data exposed the model to conditions that closely resembled real-world surveillance challenges, such as aircraft viewed from unconventional angles or under inconsistent lighting. This worked to make the model more robust in realistic environmental conditions.

With these augmentations, we effectively increased the diversity of our dataset without requiring additional data collection or labeling. This step was crucial in building models resilient to the complexities inherent in aerial aircraft detection tasks.

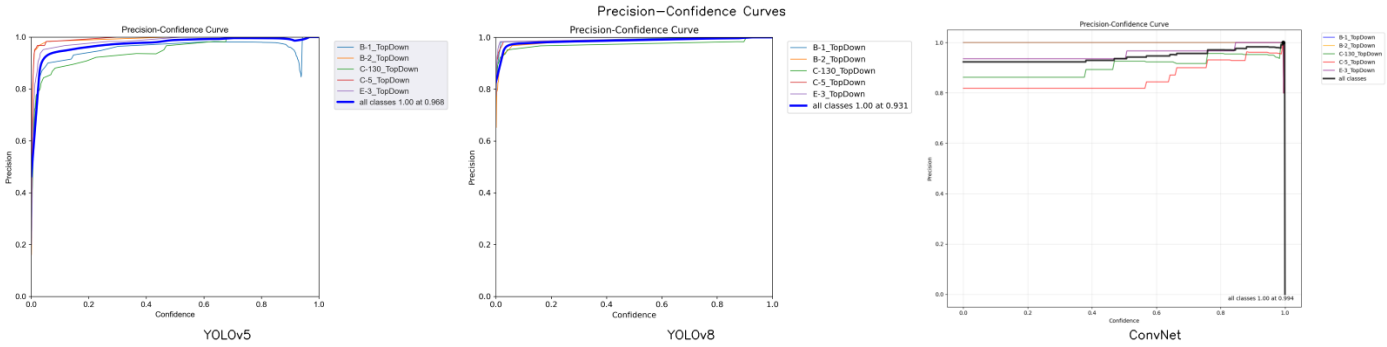

Figure 4.2

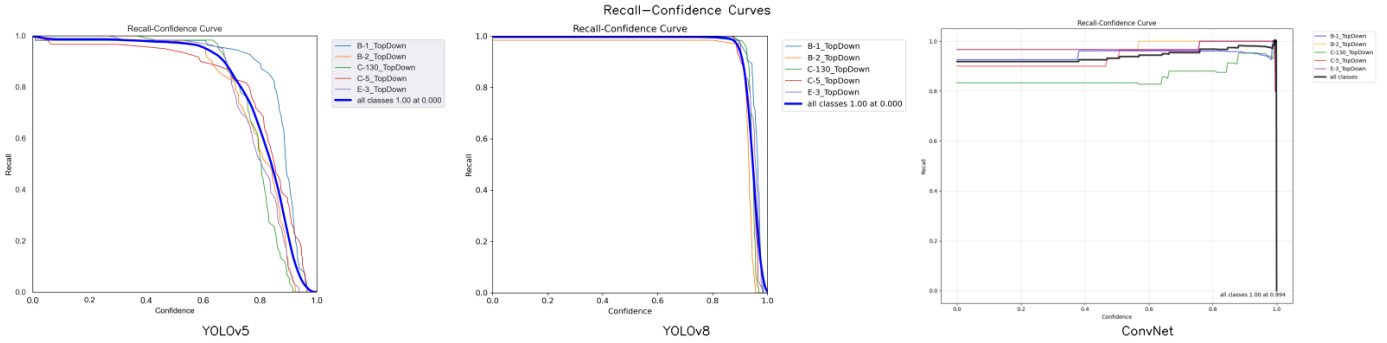

Figure 4.3

B. Model Training

To ensure a consistent training process, we made sure that all three model (YOLOv5, YOLOv8, and ConvNet) used the same number of epochs for training. This ensured that they all had an equal number of training cycles to achieve their final performance metrics.

In addition to standardizing the number of epochs, we trained all models on the same hardware. This ensured that each model used a batch size of 16 images per batch.

Finally, we chose the Adam optimizer for our ConvNet, as it is also used in both YOLOv5 and YOLOv8. Regardless of the optimizer used by those models, the Adam optimizer would have been our preferred choice due to its adaptive learning capabilities.

Figure 3.1 presents the training and validation loss trends for the custom ConvNet over 30 epochs. The model exhibits a steady decrease in both training and validation loss, with convergence occurring around the 20th epoch. This indicates effective learning and suggests that the model avoids significant overfitting, as the validation loss closely tracks the training loss throughout the process, indicating minimal overfitting.

The consistent alignment between training and validation trends highlights the robustness of the ConvNet's architecture and its ability to generalize effectively. Additionally, this performance reflects the impact of the data augmentation techniques applied during preprocessing, which helped enhance the model's ability to handle unseen data. These results confirm that the chosen training parameters and architectural design provided a strong balance between computational efficiency and detection accuracy.

C. Evaluation Metrics

Accurate evaluation of these models' performance is essential for comparing the baseline results to our custom implementation for aircraft detection. In this project, we use a variety of evaluation metrics to assess the models' effectiveness, focusing on their ability to detect and classify aircraft under diverse conditions. This section outlines each of the metrics, as well as discussions of their significance for this task. It will begin with basic measures (precision, recall, F1 score), then expand to more complex metrics and their relevance.

First, precision is a measure of proportionality between correctly detected aircraft out of all detections made. It reflects the model's ability to avoid false positives. Precision is important when minimizing false positives is crucial. In this scenario, it minimizes the scenarios of misidentifying aircraft classes in a real-world setting.

Second, recall measures the proportion of correctly detected aircraft against all true instances of that aircraft in the dataset. It assesses the model's ability to identify as many aircraft as possible. A high recall score ensures that the model is effective at identifying all relevant aircraft, which is important in this project to ensure that there are no missed targets during classification.

Third, F1 score is a balance between these precision and recall scores. It is useful as a metric when relating both these precision and recall cases as equally important.

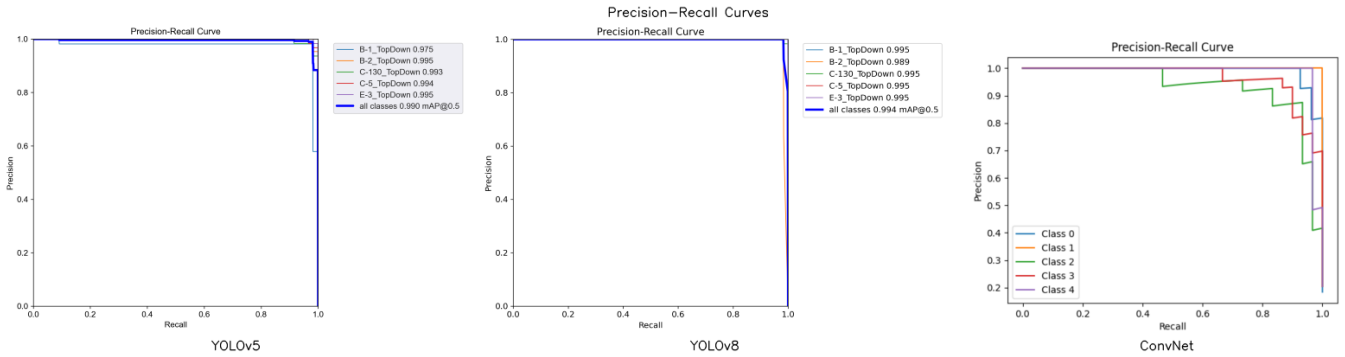

Figure 4.4

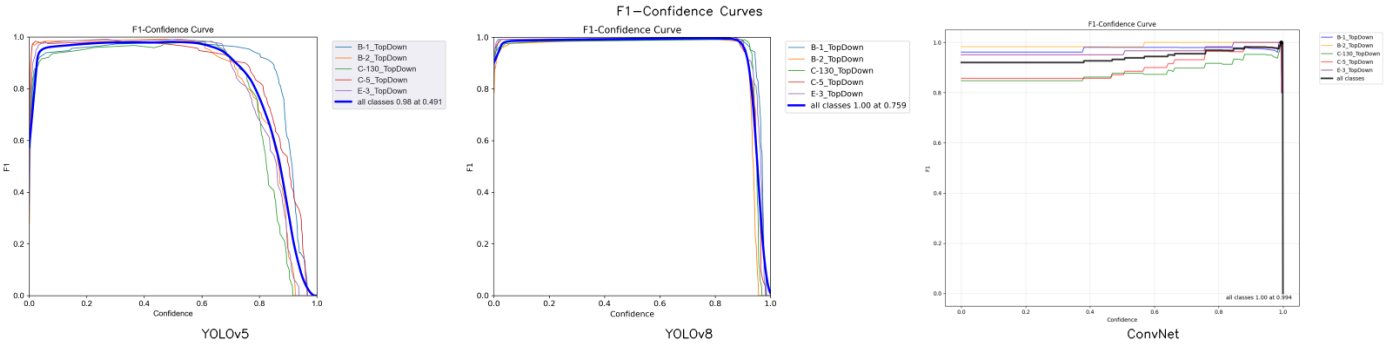

Figure 4.5

Precision, Recall, and F1 score can be seen in Figures 4.2 through 4.5. For this experiment, we will be focusing on all three metrics, as we are addressing the overall efficiency of our custom model in comparison to any shortcomings present in our baseline approach.

Another metric, Mean Average Precision (mAP), calculates the average precision across all object classes and intersection-over-union (IoU) thresholds. It represents a comprehensive metric that captures both precision and recall, providing a single score to summarize detection performance.

Our objective is to provide a method of detecting various aircraft classes, and mAP is critical for evaluating the level of ability that each model can distinguish between them. A high mAP score indicates robustness and accuracy in the multiclass classification problem, signifying that it is an important metric for this project.

Lastly, we measure the increase in performance that training our custom model has when compared to the baselines. Given that training is done using the same dataset and hyperparameters, the direct time it takes to train the model can be compared.

Ultimately, each model is compared using basic metrics – precision, recall, and F1 score – to evaluate its classification behavior. Then each model is compared using more complex metrics, such as confidence measures, mAP, and training duration to compare direct performance and behavior.

IV. Results

This section presents a comprehensive evaluation of three models: YOLOv5, YOLOv8, and a custom convolutional neural network (ConvNet), comparing their performance on a multi-class object detection task. The analysis focuses on metrics such as confusion matrices, F1-confidence curves, precision-recall curves, recall-confidence curves, and accuracy. These metrics provide insights into the models' detection capabilities, confidence calibration, and robustness across classes.

A. Performance Comparison

YOLOv5:

- **Confusion Matrix:** Detection performance is consistent across most classes, but there are slightly more false negatives for class B-1_TopDown, compared to others [Figure 4.1].
- **Precision-Confidence Curve:** Although there is a performance drop for B-1 classification just before the end, the model achieves a peak Precision score of 1.00 at a confidence threshold of 0.968 [Figure 4.2].
- **Recall-Confidence Curve:** Recall drops significantly as confidence thresholds increase beyond 0.6, revealing that YOLOv5 struggles to maintain recall for high-confidence detections. This model was able to achieve a peak recall score of 1.00 [Figure 4.3].
- **Precision-Recall Curve:** All classes exhibit tightly clustered, near-perfect precision and recall curves, consistent with the high mAP score. YOLOv5 achieved an

mAP@0.5 of 0.990, demonstrating strong detection capabilities across all classes [Figure 4.4].

- **F1-Confidence Curve:** The F1 score peaks at 0.98 at a confidence threshold of 0.491. Class-specific F1 curves show minor variability in the optimal confidence thresholds, indicating some differences in detection performance between classes [Figure 4.5].

YOLOv8:

- **Confusion Matrix:** YOLOv8 demonstrated highly consistent performance, with lower false negatives compared to YOLOv5. Class-specific performance was near-identical, reflecting excellent generalization [Figure 4.1].
- **Precision-Confidence Curve:** The performance of YOLOv8 is more stable than the performance of YOLOv5. This model achieves a Precision score of 1.00 at a confidence threshold of 0.931 [Figure 4.2].
- **Recall-Confidence Curve:** YOLOv8 maintained recall at higher confidence thresholds compared to YOLOv5, reflecting better robustness for high-confidence predictions. This model was able to achieve a peak recall score of 1.00 [Figure 4.3].
- **Precision-Recall Curve:** Class-specific curves display near-perfect alignment, indicating highly reliable performance across all classes.YOLOv8 outperformed YOLOv5, achieving an mAP@0.5 of 0.994 and showing excellent detection accuracy across all classes [Figure 4.4].
- **F1-Confidence Curve:** The F1 score peaks at 1.00 at a confidence threshold of 0.759, showcasing superior confidence calibration compared to YOLOv5. The higher confidence threshold (0.759 vs. 0.491 for YOLOv5) highlights YOLOv8's superior ability to correctly classify predictions with higher certainty [Figure 4.5].

Custom Implementation ConvNet:

- **Confusion Matrix:** YOLOv8 demonstrated highly consistent performance, with lower false negatives compared to YOLOv5. Class-specific performance was near-identical, reflecting excellent generalization [Figure 4.1].
- **Precision-Confidence Curve:** The performance of YOLOv8 is more stable than the performance of YOLOv5. This model achieves a Precision score of 1.00 at a confidence threshold of 0.931 [Figure 4.2].
- **Recall-Confidence Curve:** YOLOv8 maintained recall at higher confidence thresholds compared to YOLOv5, reflecting better robustness for high-confidence predictions. This model was able to achieve a peak recall score of 1.00 [Figure 4.3].
- **Precision-Recall Curve:** Class-specific curves display near-perfect alignment, indicating highly reliable performance across all classes.YOLOv8 outperformed YOLOv5, achieving an mAP@0.5 of 0.994 and showing excellent detection accuracy across all classes [Figure 4.4].
- **F1-Confidence Curve:** The F1 score peaks at 1.00 at a confidence threshold of 0.759, showcasing superior confidence calibration compared to YOLOv5. The higher confidence threshold (0.759 vs. 0.491 for YOLOv5) highlights YOLOv8's superior ability to correctly classify predictions with higher certainty [Figure 4.5].

B. Analysis

The custom ConvNet demonstrated superior performance over the YOLO models, particularly in confidence calibration and robustness. While YOLOv5 and YOLOv8 achieved high overall detection metrics, the ConvNet excelled in maintaining consistent stability across varying confidence thresholds, as highlighted by its F1-confidence, recall-confidence, and precision-confidence curves.

Although the YOLO models—especially YOLOv8—excelled in generalization and class consistency, making them suitable for real-time applications, the ConvNet distinguished itself with more refined performance. It exhibited fewer fluctuations across key metrics and thresholds, achieving strong stability and maintaining perfect scores in precision, recall, and F1 metrics. These results indicate that the ConvNet is well-optimized for the specific dataset and task requirements.

The ConvNet's consistent high performance, particularly in high-confidence scenarios, highlights its potential as a reliable and efficient alternative to the YOLO models for multi-class object detection tasks. While YOLOv8 demonstrated exceptional generalization and a high mAP@0.5, the ConvNet surpassed it in robustness and confidence calibration. Addressing minor classification errors through further architectural or regularization improvements could make the ConvNet even more effective in future implementations.

V. CONCLUSION

The comparative evaluation of YOLOv5, YOLOv8, and the custom convolutional neural network, ConvNet, provided valuable insights into their performance on multi-class object detection tasks. While both YOLO models showcased strong generalization and high mAP scores, the custom ConvNet demonstrated remarkable stability and robustness, excelling in confidence calibration across all metrics. The ConvNet's consistent performance in high-confidence scenarios highlights its potential for reliable and efficient object detection, especially when tailored to specific datasets and task requirements.

This study underscores the importance of task-specific model optimization and rigorous evaluation using diverse metrics such as confusion matrices, precision-recall curves, and F1-confidence curves. By identifying the strengths and limitations of each model, this work not only advances the understanding of performance trade-offs but also lays the groundwork for future improvements. Enhancing the ConvNet's regularization and further fine-tuning could address minor classification inconsistencies, driving even higher performance. Overall, the findings emphasize the critical role of custom implementations in achieving specialized detection goals and set a solid foundation for future research and development in object detection.

References

[1] Roboflow, Inc., “Roboflow: Go from Raw Images to a Trained Computer Vision Model in Minutes.,” roboflow.ai, 2024. https://roboflow.com/ (accessed Oct. 01, 2024).

[2] Ultralytics, “Ultralytics YOLOv5,” docs.ultralytics.com. https://docs.ultralytics.com/models/yolov5/ (accessed Oct. 01, 2024).

[3] Ultralytics YOLOv8, “YOLOv8,” docs.ultralytics.com, Nov. 12, 2023. https://docs.ultralytics.com/models/yolov8/ (accessed Oct. 01, 2024).

[4] S-Khos, “GitHub - S-Khos/Deep-Learning-for-Aircraft-Recognition: A CNN model trained to classify and identify various military aircraft through satellite imagery,” GitHub, 2021. https://github.com/S-Khos/Deep-Learning-for-Aircraft-Recognition?tab=readme-ov-file

[5] The Linux Foundation and The PyTorch Foundation, “PyTorch,” www.pytorch.org. https://pytorch.org/hub/ultralytics_yolov5/ (accessed Oct. 01, 2024).