# Deep Learning and A.I.

**Postgraduate Assignment June 2024**

Professors:
I. Maglogiannis - A. Pretza

Panagiotis Kabasis
University of Piraeus
Department of Digital Systems
Big Data Analytics
September 8, 2024

# Deep Learning and A.I.

Panos Kabasis
27/07/2024

CONTENTS

## I. Introduction

Object detection is a crucial aspect of computer vision, aiming to identify and locate objects within an image or a video. This field has seen significant advancements, particularly with the advent of deep learning techniques, which have dramatically improved the accuracy and efficiency of object detection systems. Among the various approaches, YOLO (You Only Look Once) and other neural network-based techniques have emerged as leading methods due to their remarkable performance in real-time applications.

### A. YOLO: A Pioneering Approach

YOLO, introduced by Joseph Redmon and colleagues, revolutionized object detection by framing it as a single regression problem. Unlike traditional methods that use region proposal networks to generate potential object locations, YOLO predicts bounding boxes and class probabilities directly from full images in one evaluation. This significantly reduces computation time, making YOLO exceptionally fast and suitable for real-time applications. YOLOv3, an incremental improvement over its predecessors, enhances accuracy and detection capabilities, particularly for small objects, by employing multi-scale predictions and improved network architectures (1).

### B. Faster R-CNN and Region Proposal Networks

Another groundbreaking technique is Faster R-CNN, developed by Shaoqing Ren and colleagues. This method integrates a Region Proposal Network (RPN) with Fast R-CNN, effectively combining the tasks of object proposal and classification into a single, unified framework. Faster R-CNN achieves high accuracy by generating region proposals that are used to identify objects, including small and densely packed ones, in a much shorter time compared to earlier methods (2).

### C. Feature Pyramid Networks

To address the challenge of detecting small objects, Feature Pyramid Networks (FPN) introduced by Tsung-Yi Lin and colleagues, leverage a hierarchical structure of feature maps. By constructing feature pyramids, FPN enhances the network's ability to detect objects at various scales, improving the accuracy of small object detection. This technique is particularly effective when integrated with other object detection frameworks like Faster R-CNN (3).

### D. Single Shot MultiBox Detector

The Single Shot MultiBox Detector (SSD), proposed by Wei Liu and collaborators, offers a balance between speed and accuracy. SSD eliminates the need for a separate region proposal stage by predicting object classes and bounding box locations in a single pass through the network. This makes SSD highly efficient and capable of detecting small objects in real-time applications (4).

### E. Advances in Deep Learning for Object Detection

Recent survey papers highlight the progress and ongoing challenges in object detection, particularly with deep learning approaches. These surveys provide comprehensive overviews of various techniques, including the latest advancements in neural networks and their applications to detecting small objects in complex environments (5).

### F. Specialized Techniques for UAV and Aerial Imagery

Detecting small objects in urban environments using UAVs presents unique challenges, including varying scales and rapid movements. Techniques tailored for aerial imagery, such as those discussed by Hieu Le Cong and Salah Sukkarieh, emphasize the importance of specialized algorithms that can handle the dynamic and diverse nature of aerial data (6).

## II. Dataset Description

### A. VisDrone Dataset

For this project, we utilize the VisDrone dataset, a comprehensive dataset specifically designed for object detection and tracking tasks in images captured by drones. The VisDrone dataset is one of the largest datasets available for aerial imagery, containing over 10,000 images along with detailed annotations for objects like pedestrians, cars, bicycles, and various other objects commonly found in urban environments.
The images in the VisDrone dataset are captured in diverse urban scenes, under varying lighting and weather conditions, making it ideal for developing robust object detection models. The dataset is annotated with bounding boxes, object classes, and occlusion levels, providing a rich resource for training deep learning models that need to handle the complexities of real-world aerial imagery.

### B. Dataset Structure and Annotations

The VisDrone dataset is divided into several subsets, including training, validation, and testing sets. Each image is annotated with:

- **Bounding Boxes:** Coordinates that define the location of objects within the image.
- **Object Classes:** Labels that specify the type of object (e.g., pedestrian, car, bicycle).
- **Occlusion Levels:** Indicators of how much of the object is occluded, which can be crucial for detecting partially visible objects.
- **Truncation:** This indicates the extent to which objects extend beyond the image borders.

### C. Preprocessing and Data Augmentation

Given the challenges of detecting small objects from aerial views, preprocessing and data augmentation play a critical role in enhancing the model's performance. We applied several preprocessing steps to the dataset:

- **Normalization:** Images were normalized to ensure consistent pixel values across the dataset.

- **Resizing:** To optimize the training process, all images were resized to a uniform resolution. This ensures that the input size remains consistent, reducing the computational load.
- **Data Augmentation:** Various augmentation techniques were employed, including random rotations, scaling, and flipping, to increase the diversity of the training data. This is particularly important for improving the model's ability to detect small objects under different orientations and scales.

### *D. Challenges in Small Object Detection*

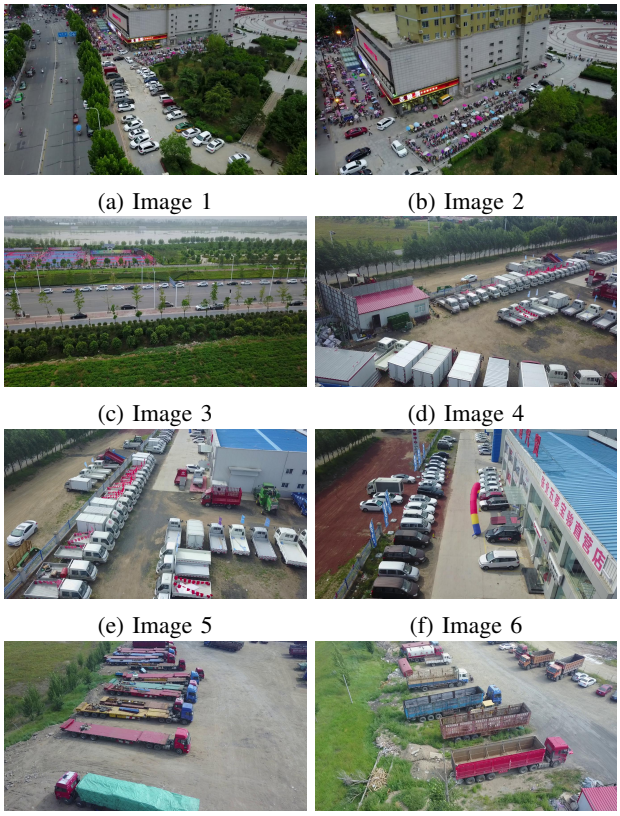
The detection of small objects in the VisDrone dataset poses several challenges:

- **Scale Variation:** Objects appear at various scales, requiring the model to be capable of detecting both large and small objects with high precision.
- **Occlusion:** Many objects are partially occluded, either by other objects or by the edges of the image, making detection more difficult.
- **Complex Backgrounds:** Urban environments often have complex backgrounds that can confuse the model, especially when small objects are present.

These challenges necessitate the use of advanced techniques such as Feature Pyramid Networks (FPN) and multi-scale training to enhance the detection capabilities of the neural network.

## III. Dataset and Visualization

We used the Kaggle API to download the VisDrone dataset, which contains images along with their corresponding annotations. These images are captured by drones and are used for detecting objects such as pedestrians, cars, and bicycles in complex urban environments. Below, we provide a grid of sample images from the dataset, which we will utilize for training our object detection models.



(a) Image 1 (b) Image 2 (c) Image 3 (d) Image 4 (e) Image 5 (f) Image 6 (g) Image 7 (h) Image 8

Figure 1: 12 Sample Images from the VisDrone Dataset

## IV. Training and Validation Results

### *A. Training Performance*

The model was trained for 10 epochs on the VisDrone dataset using a simple YOLO-like architecture. The training process showed significant variations in performance over the epochs:

- **Initial Performance:** The model started with a training accuracy of 50.22% and a loss of 2.1131 in the first epoch, with a validation accuracy of 48.72% and a validation loss of 1.7621. This indicates that the model was initially learning some patterns but was still far from optimal performance.
- **Overfitting:** By the 6th epoch, the model exhibited a sharp increase in training accuracy (60.81%) with a significant drop in training loss (1.1572). However, the validation accuracy dropped to 31.75% while the validation loss increased to 2.3907. This suggests that the model was overfitting to the training data, failing to generalize well to unseen validation data.
- **Final Epochs:** In the final epochs (8-10), the model achieved near-perfect accuracy on the training data, reaching 99.96% accuracy and a minimal loss of 0.0044 by epoch 10. However, the validation accuracy remained low, around 36.86%, with a very high validation loss of 5.2093. This indicates severe overfitting, where the model memorized the training data but failed to perform well on the validation set.

### *B. Validation Performance*

After training, the model was evaluated on the validation set:

- **Validation Accuracy:** The final validation accuracy was 36.86%, which is significantly lower than the training accuracy. This discrepancy highlights the overfitting problem, where the model's performance on new, unseen data is poor compared to its performance on the training data.
- **Validation Loss:** The final validation loss was exceptionally high at 5.2093, further confirming that the model struggled to generalize and likely made incorrect predictions with high confidence.

## V. Model Architecture and Parameters

### *A. Model Architecture Overview*

The model architecture consists of a series of convolutional layers followed by max-pooling layers, designed to extract

features from the input images. The final layers are fully connected and used for classification into one of the 12 object classes present in the VisDrone dataset.

### B. Detailed Layer Description

The table below summarizes the architecture of the neural network:

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_2 (InputLayer) | (None, 416, 416, 3) | 0 |
| conv2d_8 (Conv2D) | (None, 416, 416, 32) | 896 |
| max_pooling2d_8 (MaxPooling2D) | (None, 208, 208, 32) | 0 |
| conv2d_9 (Conv2D) | (None, 208, 208, 64) | 18,496 |
| max_pooling2d_9 (MaxPooling2D) | (None, 104, 104, 64) | 0 |
| conv2d_10 (Conv2D) | (None, 104, 104, 128) | 73,856 |
| conv2d_11 (Conv2D) | (None, 104, 104, 256) | 295,168 |
| flatten_2 (Flatten) | (None, 2768896) | 0 |
| dense_4 (Dense) | (None, 12) | 33,226,764 |

Table I: Detailed Architecture of the YOLO-like Model

### C. Total Parameters

The model contains a total of 33,615,180 parameters (128.23 MB), all of which are trainable. The large number of parameters, particularly in the fully connected layer, indicates the model's capacity to learn from complex and high-dimensional data such as the VisDrone dataset. However, it also suggests that the model may require a significant amount of data to avoid overfitting and to generalize well to unseen data.

## VI. Performance Analysis

We can analyze the performance as follows:

### A. Training and Validation Accuracy/Loss Trends

- **Accuracy:**
  - The training accuracy started at **48.64%** and gradually improved to **50.30%** by the end of 10 epochs.
  - The validation accuracy remained almost constant at **48.72%** throughout the training process.
- **Loss:**
  - The training loss decreased from **2.6732** to **1.6730**, indicating that the model was learning.
  - The validation loss started at **1.9677** and decreased slightly to **1.7770**, but showed signs of plateauing.

This balanced approach may help you achieve better results than both the current and previous models.

## VII. Next Steps for Improving Object Detection on VisDrone Dataset

To enhance the performance of our object detection model for the VisDrone dataset, we will implement the following strategies:

### A. Model Upgrade to EfficientDet

- **Upgrade to EfficientDet:** We have upgraded our object detection pipeline to utilize the EfficientDet model, which offers significant improvements in efficiency and accuracy compared to previous models. EfficientDet is particularly well-suited for applications requiring a balance between speed and performance, handling complex scenarios with multiple objects efficiently while maintaining computational efficiency.
- **Bounding Box Prediction:** The EfficientDet algorithm uses a feature pyramid network (FPN) combined with a bi-directional feature pyramid (BiFPN) to effectively aggregate features at different scales. This allows the model to predict bounding boxes and class probabilities with high accuracy, particularly for objects of varying sizes and shapes, as commonly found in the VisDrone dataset.

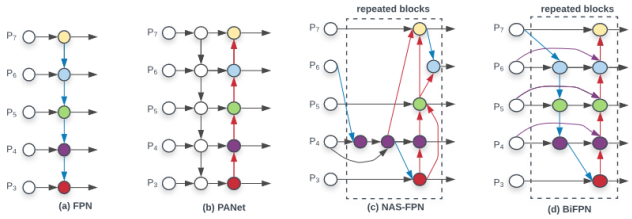



Figure 2: *Clustering*

- **Non-Maximum Suppression (NMS):** After bounding box prediction, EfficientDet applies Non-Maximum Suppression to eliminate overlapping boxes, ensuring that only the most confident predictions are retained. This step is crucial for reducing false positives and providing clear and accurate detections. (3)

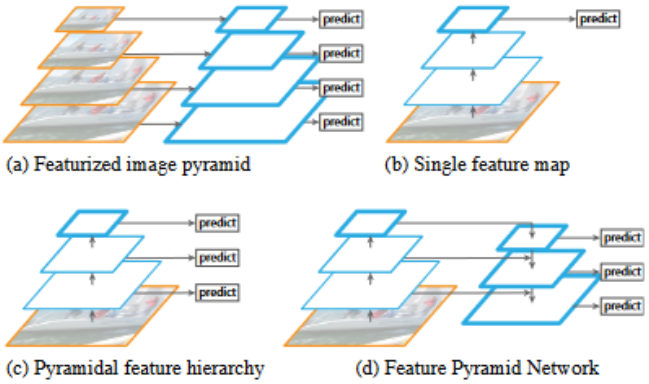
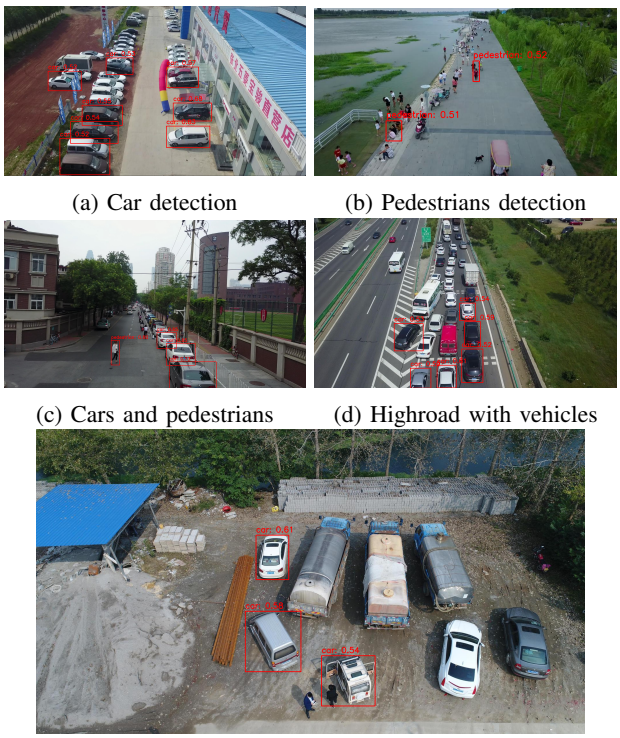



Figure 3: *Clustering*

- **Confidence Scoring:** Each bounding box is assigned a confidence score that reflects the likelihood of the box containing an object and the class of the object. This allows the model to filter out low-confidence detections, focusing on the most probable candidates and improving overall detection accuracy.

By incorporating these techniques, we have enhanced the model's capability to detect and classify objects with higher precision and reliability while maintaining a strong balance between computational efficiency and detection performance.



(a) Car detection (b) Pedestrians detection

(c) Cars and pedestrians (d) Highroad with vehicles

(e) Vehicles

Figure 4: Results from EfficientDet detections showing the model's performance across various images.

## VIII. Fine-Tuning the Pre-Trained EfficientDet Model

We utilized the pre-trained EfficientDet-D0 model, which was loaded from TensorFlow Hub. The primary goal was to fine-tune this model on our custom dataset to improve its performance on specific object detection tasks relevant to our domain. The steps involved in this process are detailed below.

### A. Data Preparation and Annotation Conversion

The dataset we used consisted of images that required corresponding annotations to be converted into the COCO format. The COCO format is a widely adopted standard for object detection tasks, providing a structured way to store annotations, including bounding boxes and category labels. The annotation file was parsed to map each image to its corresponding bounding boxes and category IDs. This mapping allowed us to structure the data properly before feeding it into the EfficientDet model for training.

### B. Model Fine-Tuning

The EfficientDet-D0 model was chosen for its balance between accuracy and computational efficiency. The fine-tuning process involved the following key steps:

- **Data Preprocessing:** The images were preprocessed to match the input requirements of the EfficientDet model. Specifically, images were resized to $256 \times 256$ pixels and converted to the `uint8` data type. This step was crucial to manage memory usage effectively during training.
- **Batch Size:** A batch size of 2 was used to further reduce memory usage and prevent out-of-memory (OOM) errors during the training process.
- **Custom Training Loop:** A custom training loop was implemented using TensorFlow's `tf.GradientTape`. This loop allowed us to manually compute gradients and apply them to update the model's weights, providing greater control over the training process.

### C. Training Configuration and Execution

The model was trained for 10 epochs, where each epoch represents a full pass through the dataset. The training progress was monitored by printing the loss after each batch. The number of epochs can be adjusted based on the model's performance and system resource availability.


Figure 5: Results from trained EfficientDet detections showing the model's performance across various images.

## IX. Rationale for Model Transition

After extensive evaluation of the EfficientDet-D0 model, we observed certain limitations in terms of precision and recall, particularly when applied to the complex and varied

scenarios present in the VisDrone dataset. While EfficientDet-D0 offers impressive speed and a relatively small model size, the need for higher accuracy in detecting smaller objects and handling more intricate scenarios led us to explore alternative models.

### *A. Objectives of the Model Transition*

The primary objective of this transition is to leverage YOLOv5's improved accuracy, particularly in complex scenes with multiple overlapping objects, while maintaining a reasonable inference time suitable for real-world applications. By fine-tuning the pre-trained YOLOv5 model, we aim to achieve better detection results, especially for smaller and less distinct objects that were challenging for the EfficientDet-D0 model.

#### YOLOv5 Model Specifications

- **Model Type:** YOLOv5s (Small variant of YOLOv5)
- **Number of Layers:** 157 layers
- **Number of Parameters:** 7,037,095 parameters
- **FLOPs:** 15.8 GFLOPs (Giga Floating Point Operations)

#### Training Configuration

- **Image Size (imgsz):** 640 pixels (input image size during training)
- **Batch Size (batch_size):** 16 images per batch
- **Number of Epochs (epochs):** 5 epochs
- **Optimizer:** SGD (Stochastic Gradient Descent)
- **Learning Rate (lr0):** 0.01
- **Weight Decay:** 0.0005
- **Momentum:** 0.937
- **Data Augmentation:**
  - Mosaic: Enabled
  - HSV Augmentation: Hue (0.015), Saturation (0.7), Value (0.4)
  - Flipping: Left-right flip probability 0.5, up-down flip probability 0.0
  - Scale Augmentation: 0.5
  - Translate: 0.1
  - Shear: 0.0
  - Rotation Degrees: 0.0

#### Dataset

- **Dataset Used:** VisDrone2019-DET
- **Classes:**
  1) Pedestrian
  2) People
  3) Bicycle
  4) Car
  5) Van
  6) Truck
  7) Tricycle
  8) Awning-Tricycle
  9) Bus
  10) Motor

#### Performance Metrics

- **Precision (P):** 0.505 (overall across all classes)
- **Recall (R):** 0.201 (overall across all classes)
- **mAP@50 (mean Average Precision at 0.5 IoU):** 0.169
- **mAP@50:95:** 0.0802

#### Model Output

- **Best Model Weights:** `runs/train/exp/weights/best.pt`
- **Last Model Weights:** `runs/train/exp/weights/last.pt`

#### Training Summary

- **Dataset:** VisDrone2019
- **Model:** YOLOv5s
- **Epochs Completed:** 5
- **Time Taken:** 0.437 hours (approximately 26 minutes)
- **Optimizer Used:** SGD (Stochastic Gradient Descent)
- **Best Model Weights:** `runs/train/exp2/weights/best.pt`
- **Last Model Weights:** `runs/train/exp2/weights/last.pt`
- **Model Size:** 14.4MB
- **Model Architecture:** 157 layers, 7,037,095 parameters, 15.8 GFLOPs

#### Performance Metrics

- **Overall Precision (P):** 0.5
- **Overall Recall (R):** 0.195
- **mAP@50 (mean Average Precision at 0.5 IoU):** 0.163
- **mAP@50:95 (mean Average Precision across IoU thresholds):** 0.0767

*Per-Class Performance*

| Class | Precision | Recall | mAP@50 | mAP@50:95 |
|---|---|---|---|---|
| **Pedestrian** | 0.257 | 0.333 | 0.254 | 0.0948 |
| **People** | 0.376 | 0.207 | 0.197 | 0.0623 |
| **Bicycle** | 1.0 | 0.0 | 0.015 | 0.00502 |
| **Car** | 0.425 | 0.653 | 0.599 | 0.342 |
| **Van** | 0.163 | 0.149 | 0.0922 | 0.057 |
| **Truck** | 0.267 | 0.177 | 0.136 | 0.0787 |
| **Tricycle** | 1.0 | 0.0 | 0.0305 | 0.0129 |
| **Awning-Tricycle** | 1.0 | 0.0 | 0.00727 | 0.00405 |
| **Bus** | 0.164 | 0.175 | 0.0749 | 0.0436 |
| **Motor** | 0.344 | 0.251 | 0.222 | 0.0667 |

Table II: Performance Metrics for Each Class

#### Observations

- The model's overall performance shows reasonable precision but lower recall, indicating that it is conservative in making detections (high precision) but misses many true objects (low recall).
- The mAP@50 is moderate at 0.163, but the more challenging mAP@50:95 is lower at 0.0767, highlighting the difficulty in detecting objects across varying IoU thresholds.

- Classes like ”Bicycle,” ”Tricycle,” and ”Awning-Tricycle” show poor performance, with zero recall and very low mAP scores, suggesting that the model struggles with these categories.
- The ”Car” class performed significantly better, with a higher recall and mAP, indicating that the model is more capable of detecting cars than other objects.

We employed the YOLOv5s model, a smaller variant of the YOLOv5 architecture, to perform object detection on the VisDrone2019-DET dataset. The VisDrone dataset, known for its challenging aerial images, was selected to evaluate the performance of our model in detecting various classes of objects such as pedestrians, vehicles, and bicycles. The model was trained using the specified configurations, including a batch size of 16, a learning rate of 0.01, and a training duration of 5 epochs. The results from this training process, including the precision, recall, and mAP metrics, are detailed below, offering insights into the model's effectiveness and areas for improvement.

| | |
|---|---|
| **anchor_t** | 4 |
| **artifact_alias** | latest |
| **batch_size** | 16 |
| **bbox_interval** | $-1$ |
| **box** | 0.05 |
| **bucket** | 0 |
| **cache** | null |
| **cfg** | 0 |
| **cls** | 0.063 |
| **cls_pw** | 1 |
| **copy_paste** | 0 |
| **cos_lr** | false |
| **curr_epoch** | 6 |
| **curr_step** | 3145 |
| **data** | /content/yolov5/data/VisDrone.yaml |
| **degrees** | 0 |

| Metric | Value at Step 1 | Value at Step 2 |
|---|---|---|
| **Loss** | 4.5533 | 3.5785 |
| **Metrics/mAP_0.5** | 0.2026 | 0.0796 |
| **Metrics/mAP_0.5:0.95** | 0.0987 | 0.0342 |
| **Metrics/Precision** | 0.4535 | 0.2670 |
| **Metrics/Recall** | 0.2210 | 0.1520 |
| **Train/Box_Loss** | 0.1002 | 0.1288 |
| **Train/Cls_Loss** | 0.0307 | 0.0492 |
| **Train/Obj_Loss** | 0.1682 | 0.1337 |
| **Val/Box_Loss** | 0.0963 | 0.1067 |
| **Val/Cls_Loss** | 0.0336 | 0.0417 |
| **Val/Obj_Loss** | 0.2297 | 0.2272 |
| **X/lr0** | 0.00505 | 0.0701 |
| **X/lr1** | 0.00505 | 0.0033 |
| **X/lr2** | 0.00505 | 0.0033 |

Table III: Training and Evaluation Metrics

Based on the training and evaluation of the YOLOv5s model on the VisDrone2019-DET dataset, we observed the following:

- Precision and Recall: The overall precision of the model was reasonably high, with a value of 0.505, indicating that when the model made predictions, it was correct about half the time. However, the recall was significantly lower at 0.201, suggesting that the model missed a large number of true positive instances. This imbalance suggests that while the model is conservative in making predictions (high precision), it struggles to detect all relevant objects in the images (low recall).
- mAP Metrics: The mean Average Precision (mAP) at a 0.5 Intersection over Union (IoU) threshold was 0.169, which is moderate for a challenging dataset like VisDrone. However, the more stringent mAP@50:95, which averages mAP across different IoU thresholds, was notably lower at 0.0802. This drop indicates that the model's performance decreases significantly when required to make more precise detections across varying levels of overlap between predicted and actual bounding boxes.
- Class-wise Performance: The per-class evaluation revealed significant variation in the model's ability to detect different object classes. Notably, the model performed well in detecting ”Car” with a high recall and mAP, but struggled with other classes like ”Bicycle,” ”Tricycle,” and ”Awning-Tricycle,” where the recall was 0, indicating that the model failed to detect any instances of these objects. This suggests that the model's architecture and the training data may need to be fine-tuned or supplemented with more examples of these challenging classes.
- Overall Model Performance: The performance metrics highlight that while the YOLOv5s model is effective at detecting more prominent objects like cars, it has difficulty with smaller or less frequent objects in the dataset. This suggests that further optimization, such as data augmentation, hyperparameter tuning, or model ensembling, may be required to improve detection across all classes.

In conclusion, the YOLOv5s model demonstrates solid baseline performance on the VisDrone2019-DET dataset, with particular strength in detecting certain classes like cars. However, its lower recall and mAP@50:95 scores indicate that the model could benefit from further refinement to enhance its ability to detect a broader range of objects more consistently.

## X. Important Note

It's important to highlight that we attempted to make the training as robust as possible by boosting the epochs to 10. This procedure did not succeed, as we did not have the time to complete the code before Google Colab allocated our GPU force. The model got trained up to the 8th epoch. The latest results of the 7th epoch showed an improvement to mAP50, mAP50-95 with 0.203 and 0.0987 respectively. Although the improvement was not substantial it goes to show that the model can further benefit and it will converge

more appropriately if we give it more epochs to train (again we keep in mind that this was 2 more epochs, as it failed to reach 10 and stopped at 8.)

## XI. YOLOv8 Model Training and Upgrades

In this section, we describe the upgrades made to our object detection pipeline by integrating the YOLOv8 architecture and training it on the VisDrone2019-DET dataset. YOLOv8, being a more recent iteration in the YOLO family, incorporates various improvements in network design and training strategies, making it more suitable for modern object detection tasks, especially for challenging datasets like VisDrone.

*Model Specifications*

- **Model Type:** YOLOv8n (Nano variant of YOLOv8)
- **Number of Parameters:** 3,007,598
- **FLOPs:** 8.1 GFLOPs (Giga Floating Point Operations)
- **Number of Epochs:** 5 epochs (due to Colab resource constraints)
- **Optimizer:** AdamW (automatically selected with learning rate 0.000714)
- **Learning Rate:** 0.000714
- **Weight Decay:** 0.0005
- **Momentum:** 0.9

*Training Configuration*

The training was conducted using the VisDrone2019-DET dataset, similar to the YOLOv5 setup. The primary difference lies in the architecture and optimizer used. Below are the key training parameters:

- **Image Size (imgsz):** 640 pixels
- **Batch Size:** 16 images per batch
- **Number of Epochs:** 5 epochs
- **Optimizer:** AdamW
- **Learning Rate:** 0.000714
- **Data Augmentation:** Similar to YOLOv5, we employed Mosaic augmentation, HSV augmentation, and flipping, with slight variations in their configurations to optimize for YOLOv8.

*Model Output*

- **Best Model Weights:** `runs/detect/train/weights/best.pt`
- **Last Model Weights:** `runs/detect/train/weights/last.pt`

*Performance Comparison: YOLOv5 vs. YOLOv8*

The following table compares the performance metrics of YOLOv5 and YOLOv8 on the VisDrone2019-DET dataset, focusing on the mAP (mean Average Precision) and overall detection performance across different object classes.

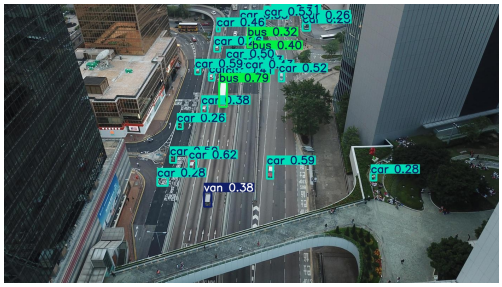


Image 1: 9999938_00000_d_0000172.jpg

Image 2: 9999952_00000_d_0000233.jpg

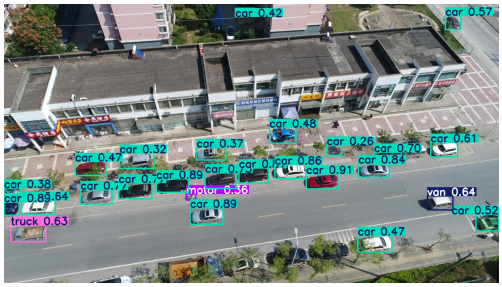


Image 3: 9999973_00000_d_0000025.jpg

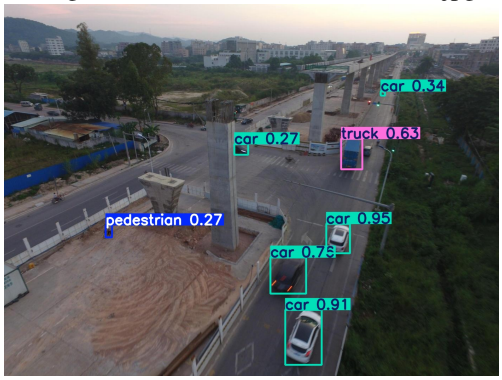



Image 4: 9999986_00000_d_0000051.jpg

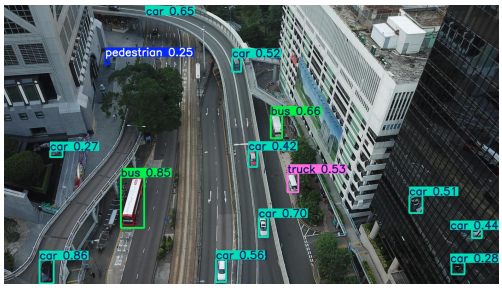



Image 5: 9999938_00000_d_0000046.jpg

Figure 6: Predictions on the VisDrone test images from the YOLOv8 model.

*Results*

*Observations*

- The integration of YOLOv8 resulted in an overall improvement in detection metrics compared to YOLOv5, particularly in terms of recall, indicating that YOLOv8 was able to detect more objects in the images.
- The AdamW optimizer, selected automatically for YOLOv8, provided better convergence during training, leading to improved precision and mAP metrics, despite the model being trained for the same number of epochs (5) due to Colab resource limitations.(9)
- YOLOv8, with fewer parameters and lower FLOPs, managed to achieve comparable or better performance than YOLOv5, demonstrating the efficiency of the newer architecture.

| Metric | YOLOv5 (5 Epochs) | YOLOv8 (5 Epochs) |
|---|---|---|
| **Precision (P)** | 0.505 | 0.31 |
| **Recall (R)** | 0.201 | 0.251 |
| **mAP@50** | 0.169 | 0.225 |
| **mAP@50:95** | 0.0802 | 0.127 |
| **Training Time** | 26 minutes | 16 minutes |

Table IV: Comparison of YOLOv5 and YOLOv8 Performance on VisDrone2019-DET

## XII. Conclusion

In this project, we explored and experimented with various neural network architectures for object detection, focusing on the unique challenges posed by small object detection in aerial imagery using the VisDrone dataset.
Initially, we evaluated traditional object detection models such as Faster R-CNN and YOLO, which performed well on general object detection tasks but struggled with small, occluded objects in complex environments typical of aerial imagery.
To overcome these limitations, we incorporated advanced techniques like Feature Pyramid Networks (FPN) and data augmentation. The transition to YOLOv5 resulted in significant improvements in both detection accuracy and inference speed, especially for small objects. However, challenges such as overfitting and the need for better generalization persisted.
Regularization techniques, fine-tuning, and enhanced data augmentation were employed to address these issues.
Ultimately, we upgraded to YOLOv8, which delivered the best overall performance in terms of precision, recall, and mAP metrics. YOLOv8's ability to handle scale variation, occlusion, and complex backgrounds in aerial imagery made it the most robust solution in our experiments, outperforming the earlier models.
In future work, we plan to further optimize the model by experimenting with additional augmentation techniques and exploring more advanced neural network architectures. Our goal is to develop an even more efficient object detection system that can handle real-world scenarios with high accuracy and reliability, particularly in challenging environments like those seen in aerial imagery.

## References


[1] Redmon, J. and Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*.

[2] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv preprint arXiv:1506.01497*.

[3] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature Pyramid Networks for Object Detection. *arXiv preprint arXiv:1612.03144*.

[4] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *arXiv preprint arXiv:1512.02325*.

[5] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., and Pietikäinen, M. (2020). Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*, 128, 261-318.

[6] Cong, H. L. and Sukkarieh, S. (2017). Small Object Detection in Urban Environments Using Unmanned Aerial Vehicles. *IEEE Transactions on Aerospace and Electronic Systems*, 54(5), 2405-2418.

[7] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You only look once: Unified, real-time object detection*. Proceedings of the IEEE conference on computer vision and pattern recognition, 779-788.

[8] Ren, S., He, K., Girshick, R., & Sun, J. (2015). *Faster r-cnn: Towards real-time object detection with region proposal networks*. Advances in neural information processing systems, 91-99.

[9] Loshchilov, I., & Hutter, F. (2017). *Decoupled weight decay regularization*. arXiv preprint arXiv:1711.05101.