Drone Detection Project

Authors:
Denis Dagaev
Evgeniy Zaborschikov

Problem statement

- A collection of 14 black-white videos with annotations of drones presented on them

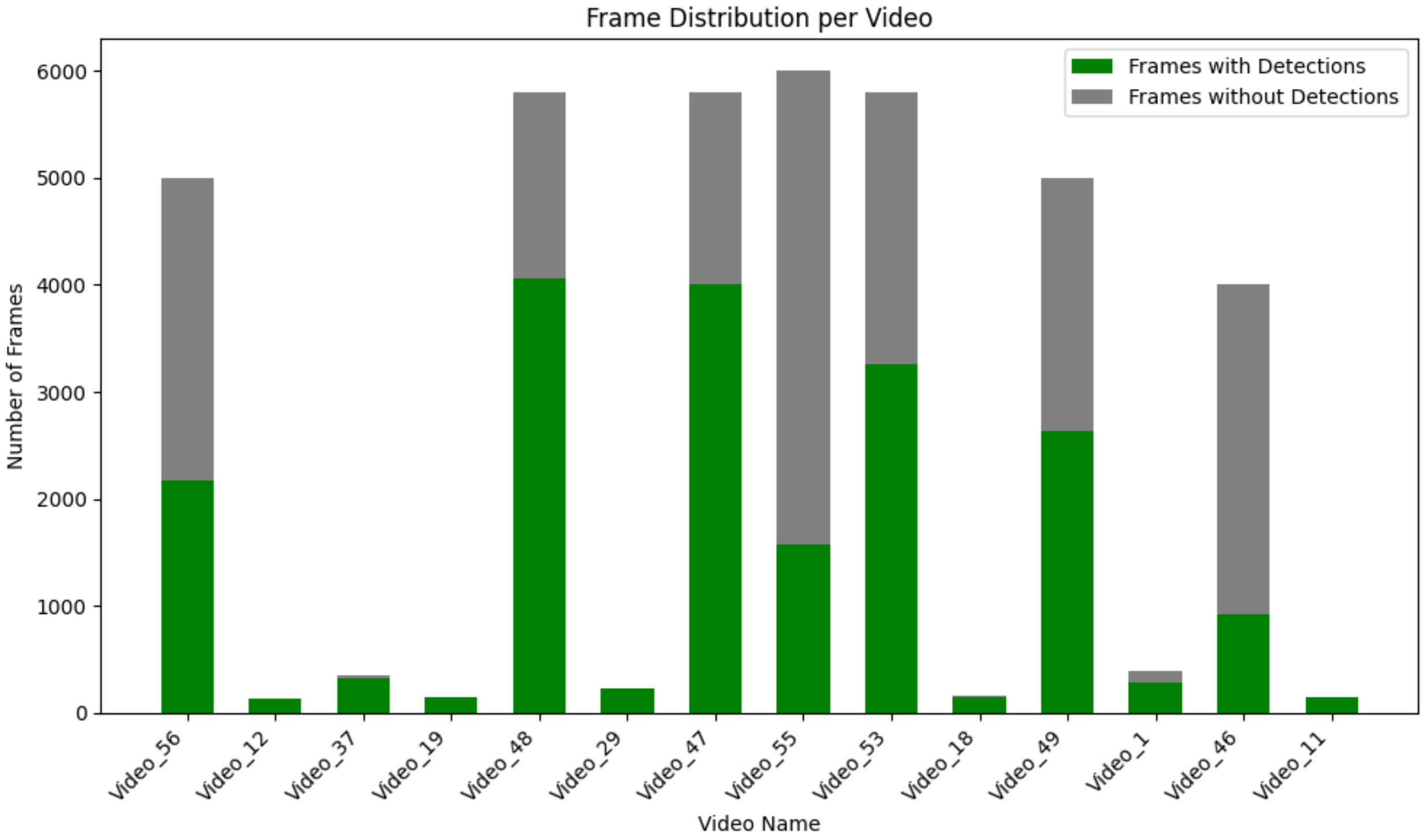
- **Task:** develop and train model to detect drones in real-time on these videos

- **Criteria:** evaluate model performance on the given annotations and calculate metrics

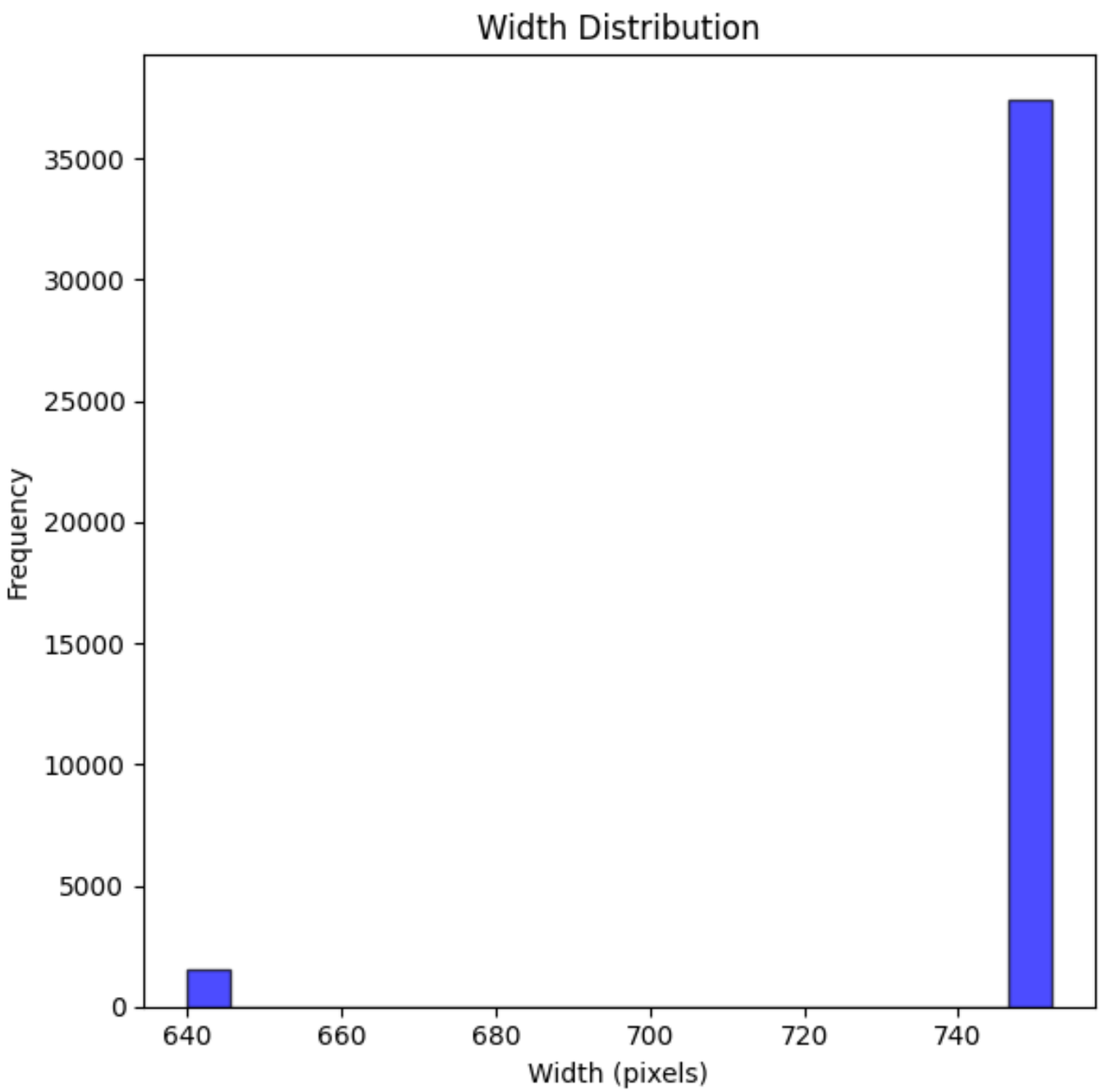
Dataset exploration

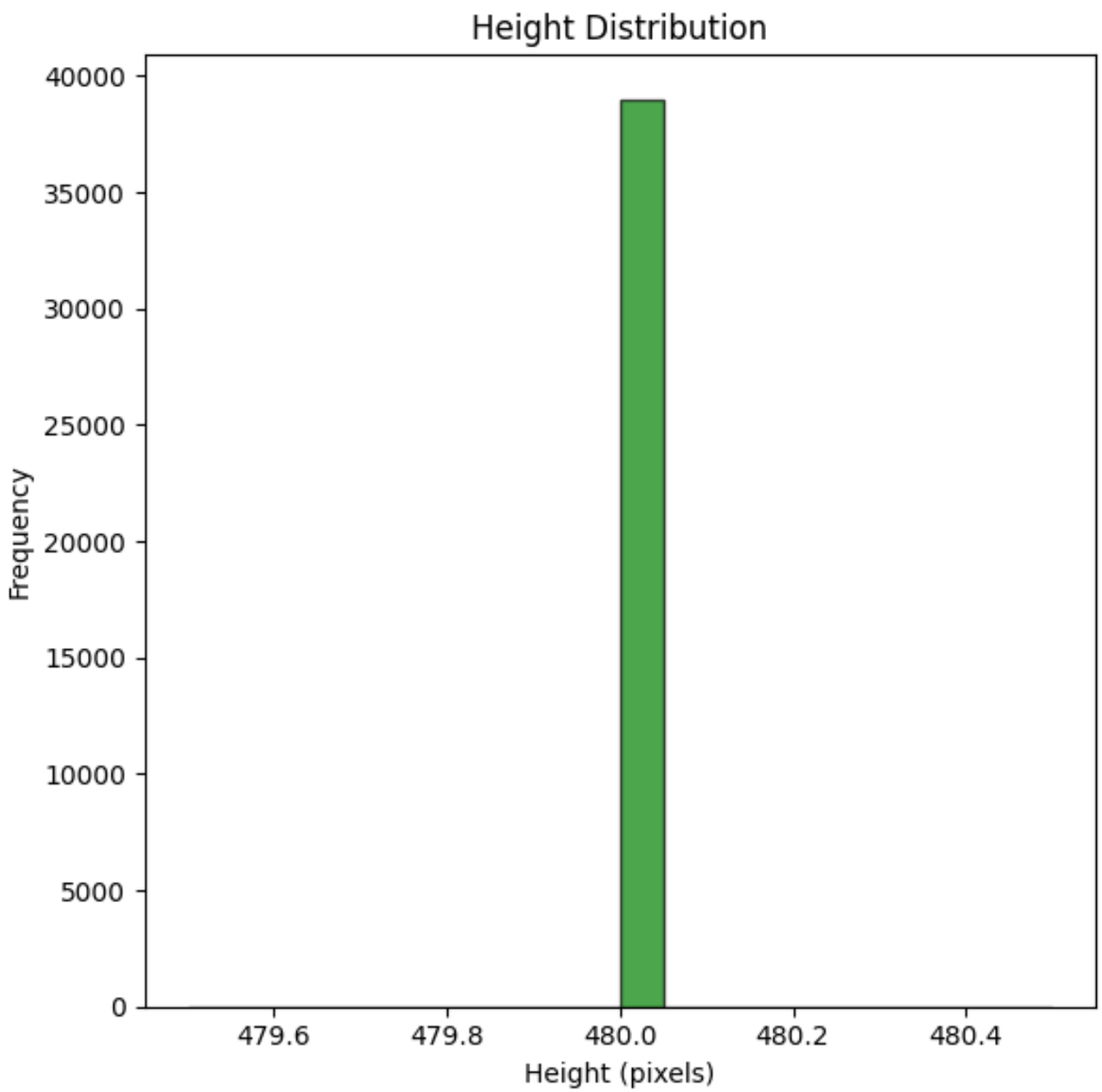
- Dataset contains 14 videos
- Total 38948 frames
- 48.6% of images with no drones detection

Data exploration

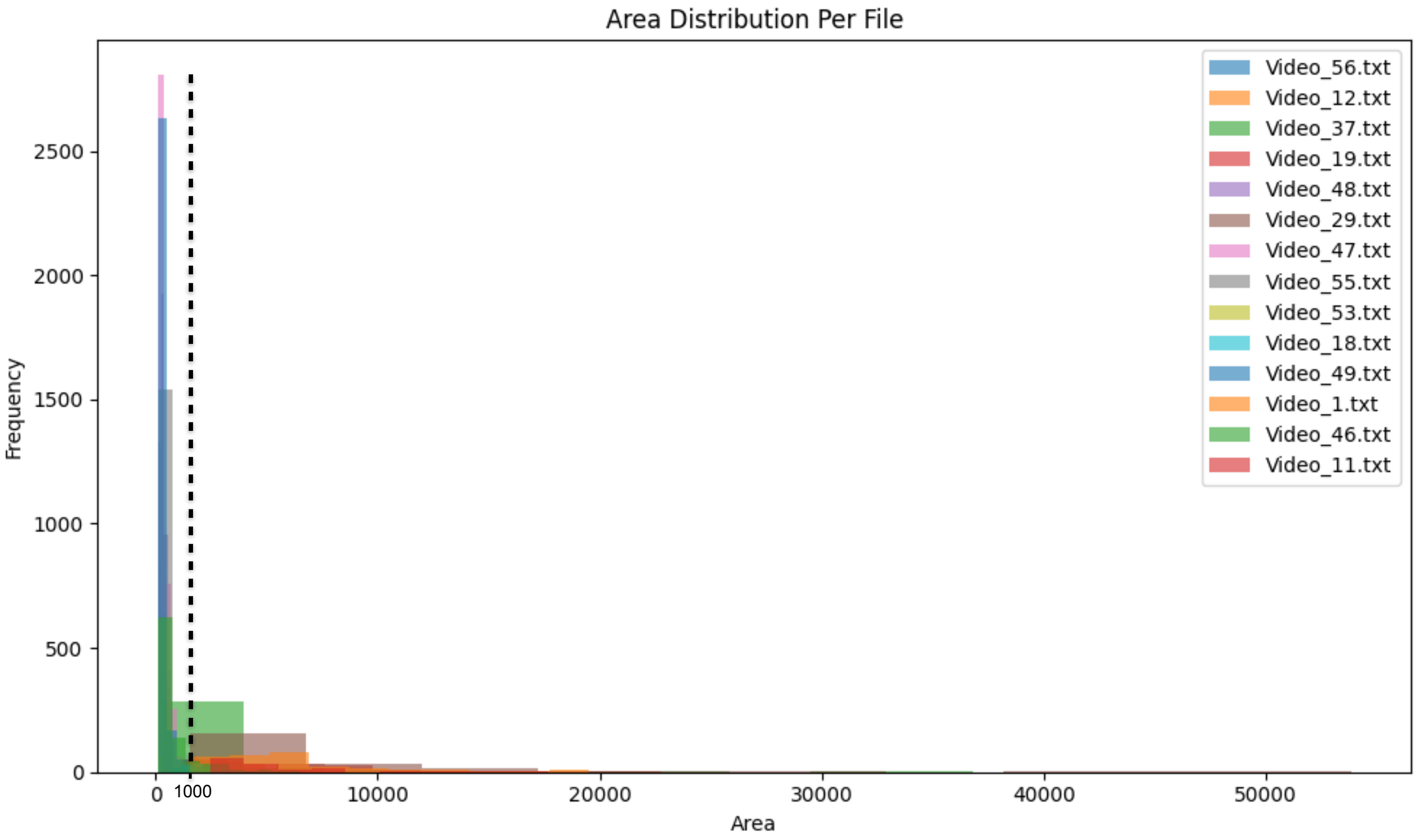


Images resolution





Drone’s area


Area Distribution Per File
Frequency
Area
0
500
1000
1500
2000
2500
0
1000
10000
20000
30000
40000
50000
Video_56.txt
Video_12.txt
Video_37.txt
Video_19.txt
Video_48.txt
Video_29.txt
Video_47.txt
Video_55.txt
Video_53.txt
Video_18.txt
Video_49.txt
Video_1.txt
Video_46.txt
Video_11.txt

Data preprocessing

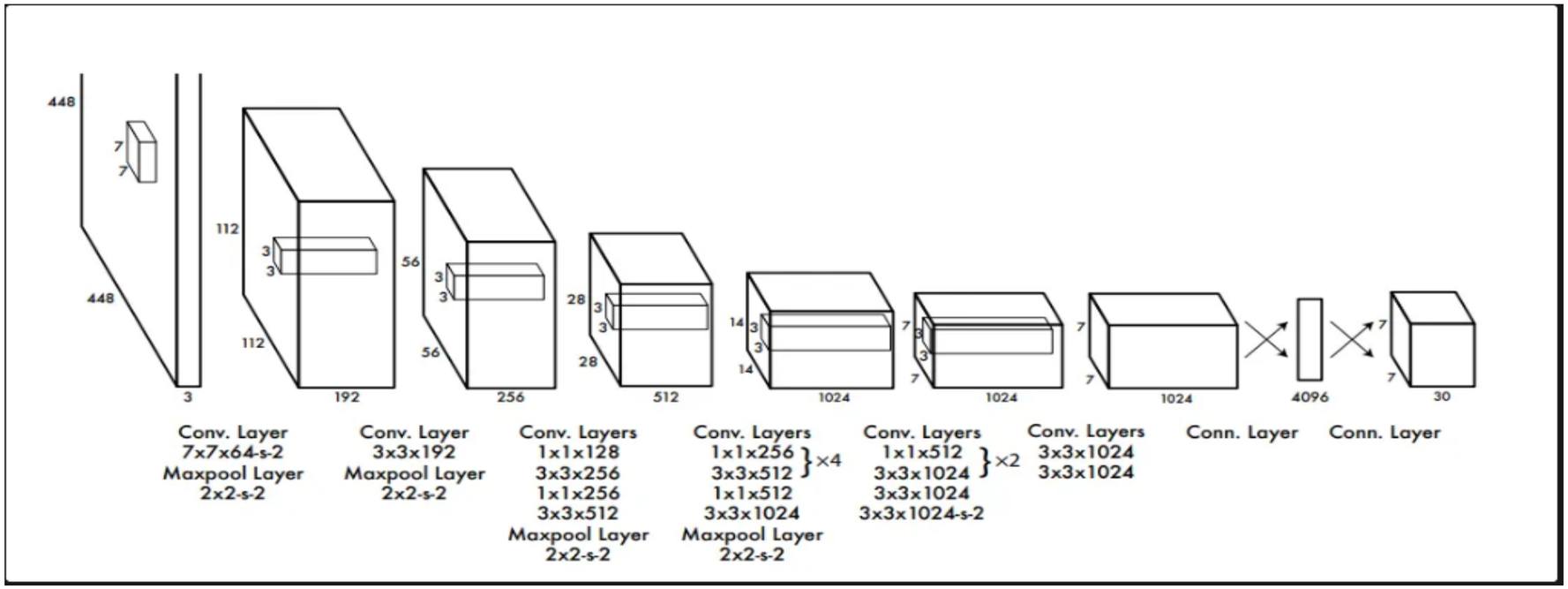
- Since all images has the same height and 2 different widths, 2 approaches have been proposed to save object's ratio: letterboxing and cropping

- Crop to 480x480 was chosen in order to save size of objects **(most of them too small)**

Architecture selection

- YOLOv11 M size
- RT-DETR

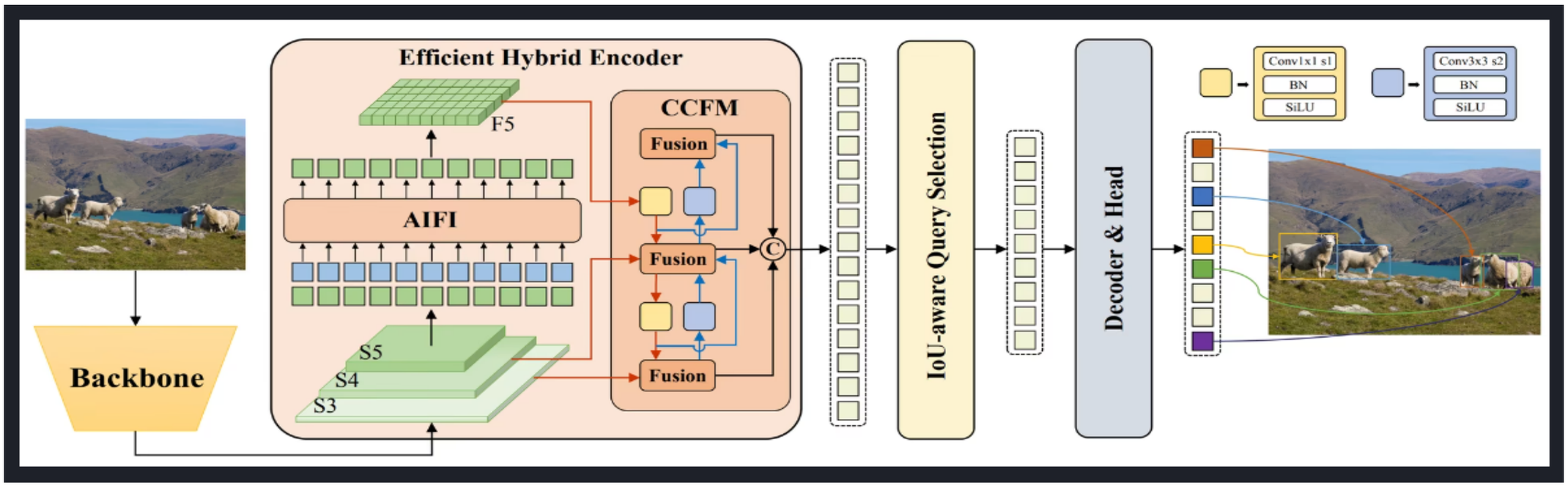
Yolo network architecture



- The core of YOLO's architecture is a single convolutional neural network (CNN) that simultaneously predicts multiple bounding boxes and class probabilities for each bounding box. The key components of YOLO's architecture are:

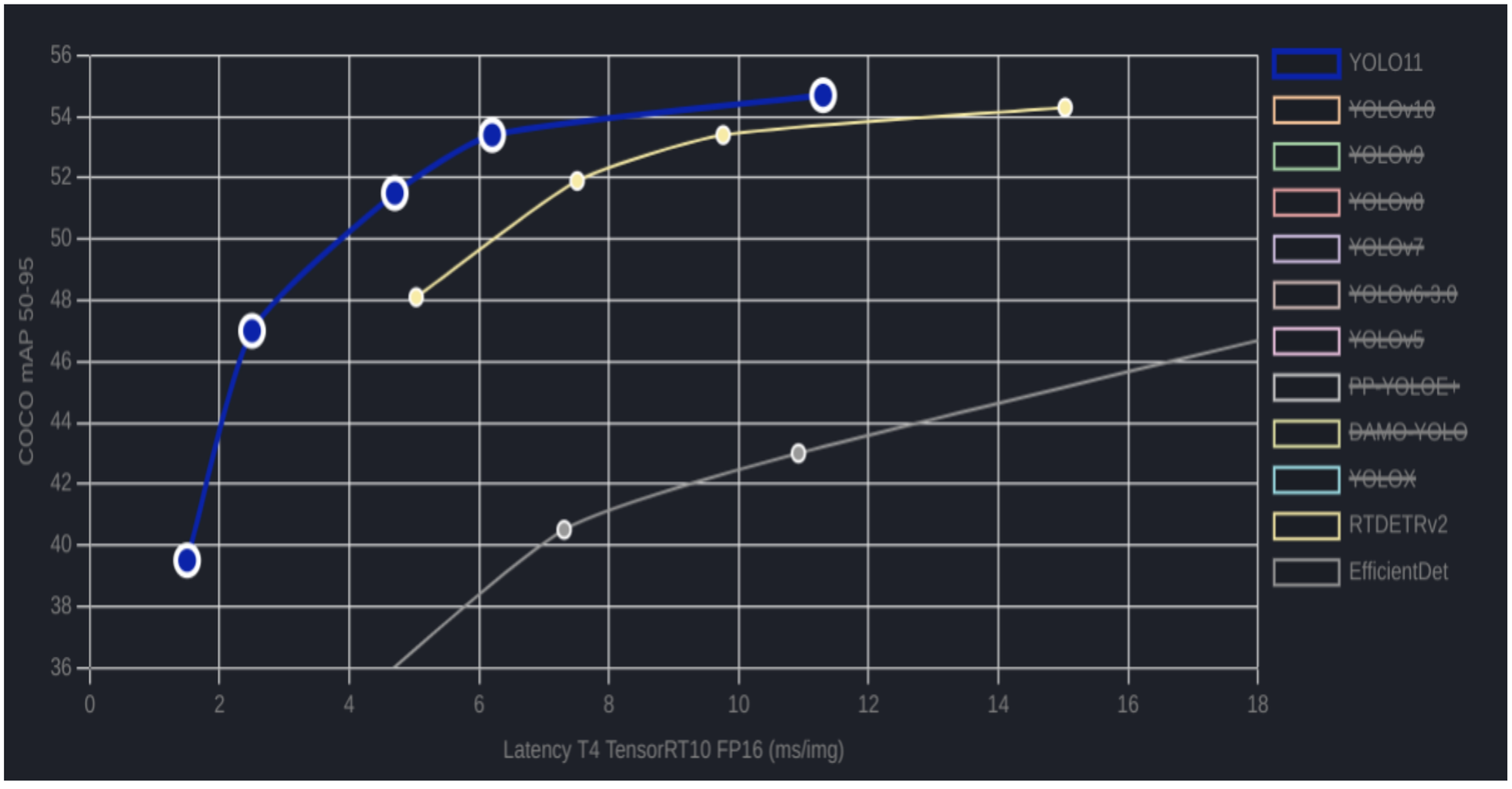
- Grid Division: The input image is divided into an S x S grid. Each grid cell is responsible for detecting objects whose center falls within it.
- Bounding Box Prediction: Each grid cell predicts B bounding boxes, each with a confidence score. The confidence score reflects the probability that a bounding box contains an object and the accuracy of the bounding box.
- Class Prediction: Each grid cell also predicts the class probabilities C for the object, given that an object exists in that grid cell.
- Final Predictions: YOLO combines these predictions to generate the final detected objects, applying non-max suppression to eliminate redundant bounding boxes.

RT-DETR network architecture



- The RT-DETR model architecture diagram shows the last three stages of the backbone {S3, S4, S5} as the input to the encoder. The efficient hybrid encoder transforms multiscale features into a sequence of image features through intrascale feature interaction (AIFI) and cross-scale feature-fusion module (CCFM). The IoU-aware query selection is employed to select a fixed number of image features to serve as initial object queries for the decoder. Finally, the decoder with auxiliary prediction heads iteratively optimizes object queries to generate boxes and confidence scores (source).
-
- Key Features
- Efficient Hybrid Encoder: Baidu's RT-DETR uses an efficient hybrid encoder that processes multiscale features by decoupling intra-scale interaction and cross-scale fusion. This unique Vision Transformers-based design reduces computational costs and allows for real-time object detection.
- IoU-aware Query Selection: Baidu's RT-DETR improves object query initialization by utilizing IoU-aware query selection. This allows the model to focus on the most relevant objects in the scene, enhancing the detection accuracy.
- Adaptable Inference Speed: Baidu's RT-DETR supports flexible adjustments of inference speed by using different decoder layers without the need for retraining. This adaptability facilitates practical application in various real-time object detection scenarios.

Performance Metrics



Data preparation

- Conversion to YOLOv8 format with the following structure:
  - dataset
    - images
      - train
      - val
      - test
    - labels
      - train
      - val
      - Test
    - data.yaml

Data preparation

- Converted origin annotations to YOLO format with convert_to_yolo(x1, y1, x2, y2, img_width, img_height) -> (center_x, center_y, width, height) in YOLO format.

- Took all images with annotations and include 20% of no-detection images to make the dataset more diverse

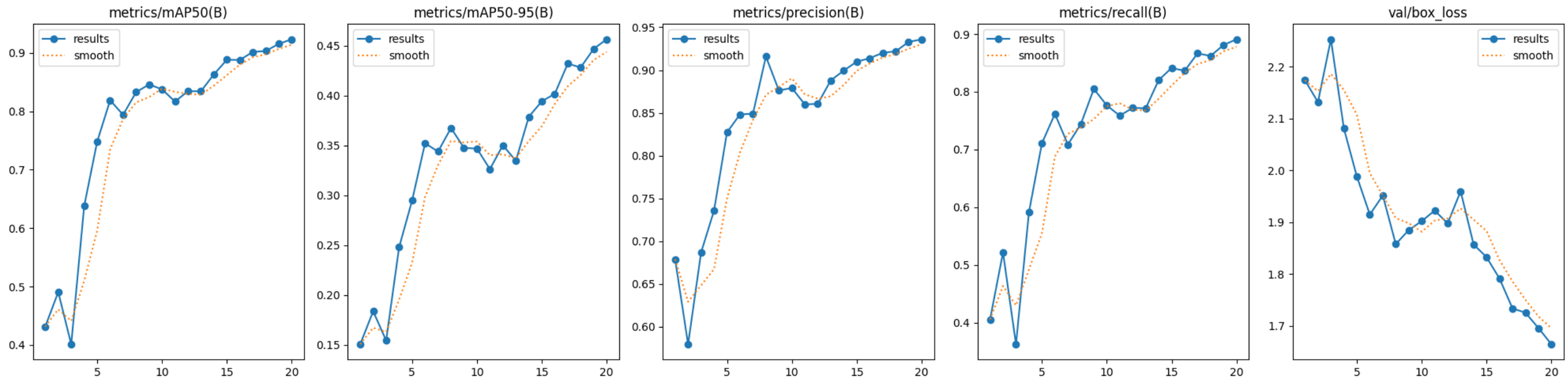
Dataset prepared

- Total number of images in dataset: 20364
- Number of detections in dataset: 18704
- Percentage of images with detection is 88.94%
- Train/test/val splits with 70/15/15 ratio

Case 1: YOLOv11m

- Ultralytics YOLOv11 M size model as SOTA object detection model
- Train for 20 epochs on our dataset
- Best score mAP50 is 0.9233 at epoch 20
- Best score mAP50-95 is 0.45646 at epoch 20

Case 1: YOLOv11m



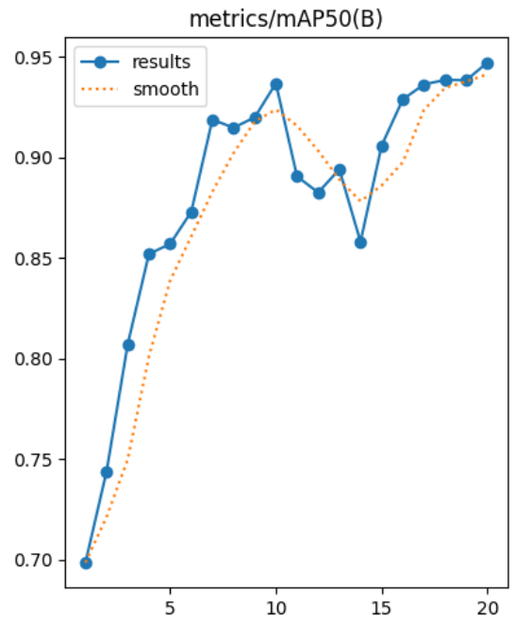
Case 2: VisDrone pertained

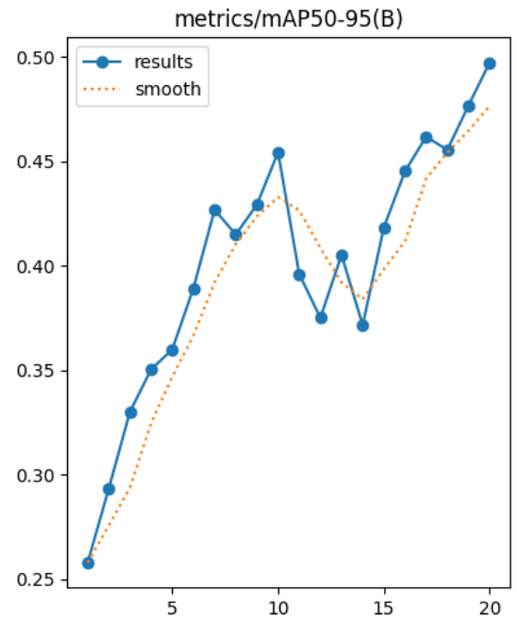
- Our dataset contains many small drones in a dynamically moving scene, similar to drone-mounted footage. To enhance detection of small, fast objects, we pre-train on VisDrone, a diverse dataset of 288 video clips (261,908 frames) and 10,209 images captured by drone-mounted cameras across 14 cities in various environments, lighting, and weather conditions.

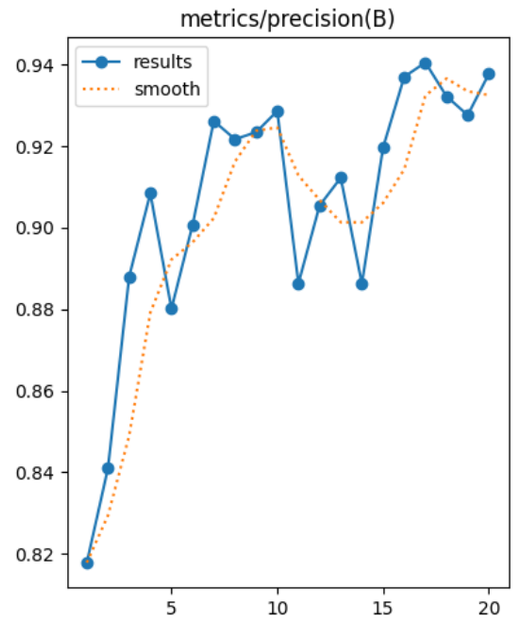
- Best score mAP50 is 0.94717 at epoch 20

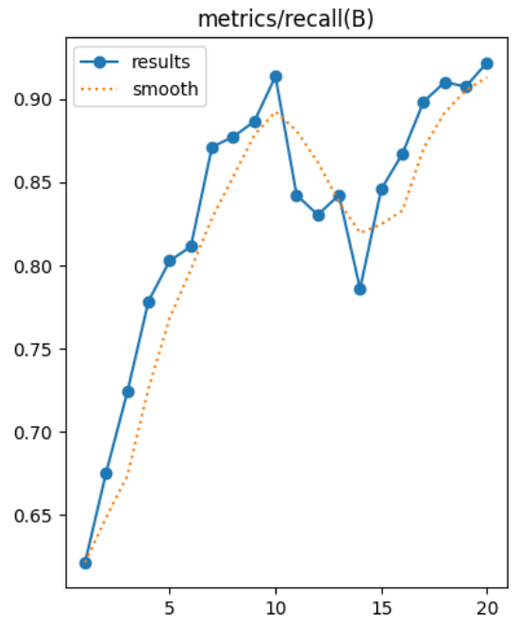
- Best score mAP50-95 is 0.49744 at epoch 20

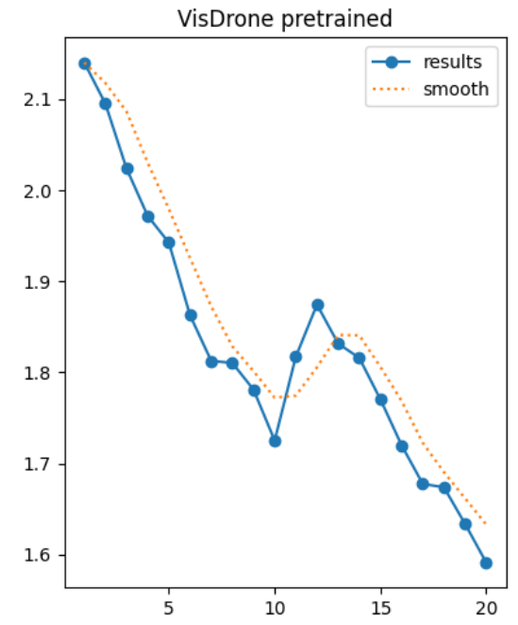
Case 2: VisDrone pertained



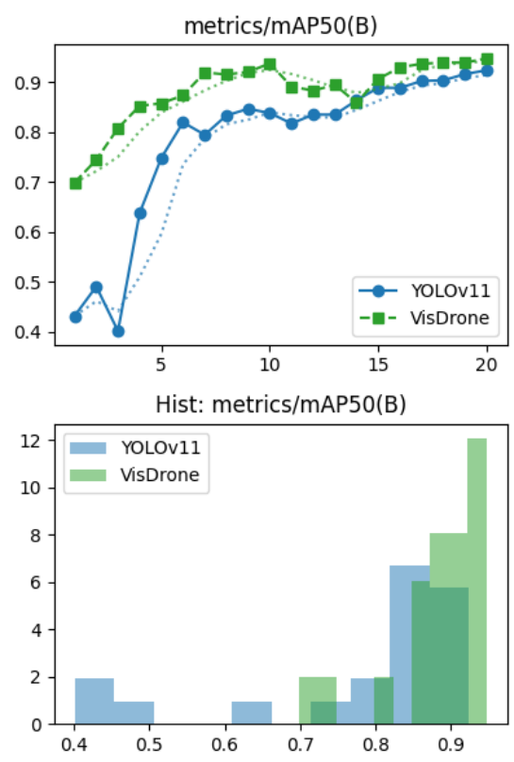


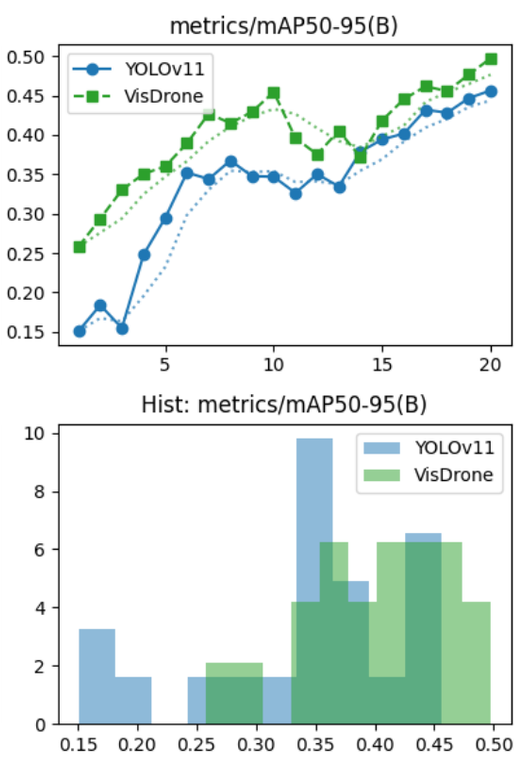


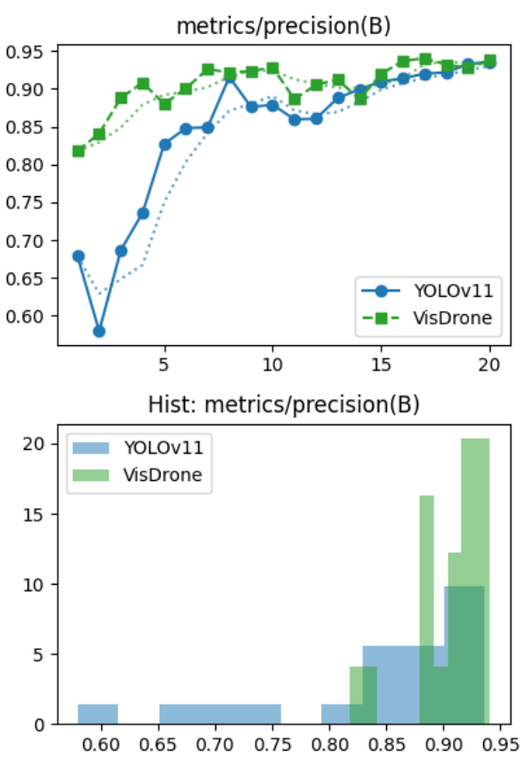


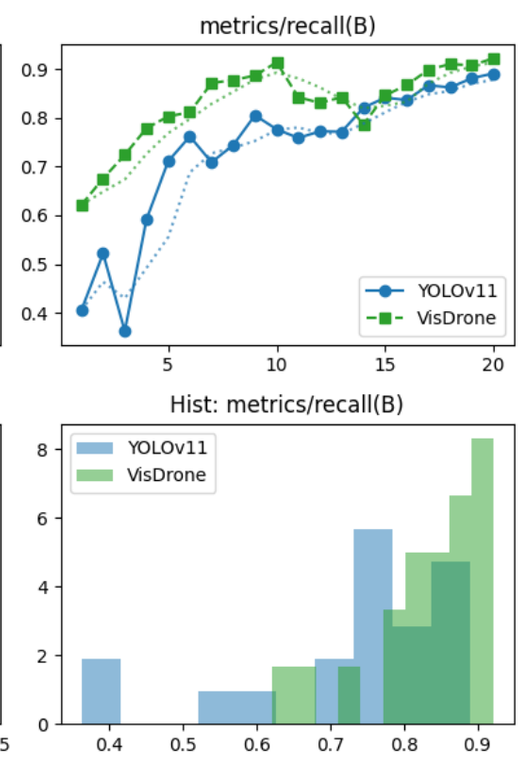


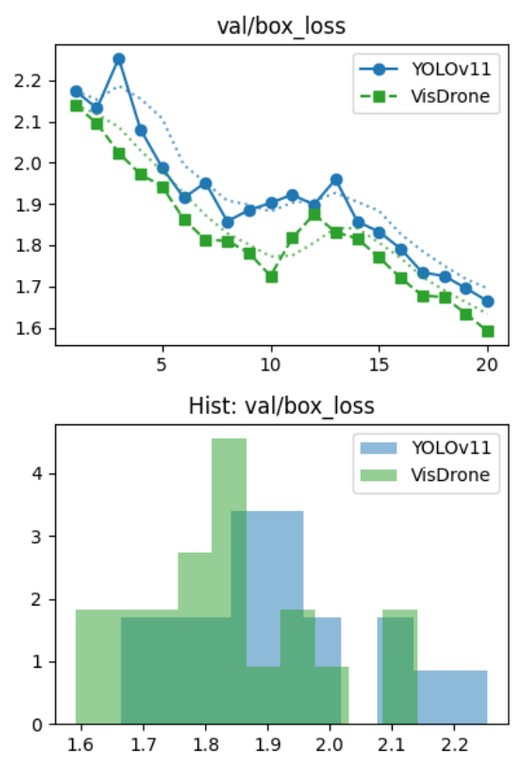
YOLOv11 vs YOLOv11 VisDrone pre-trained



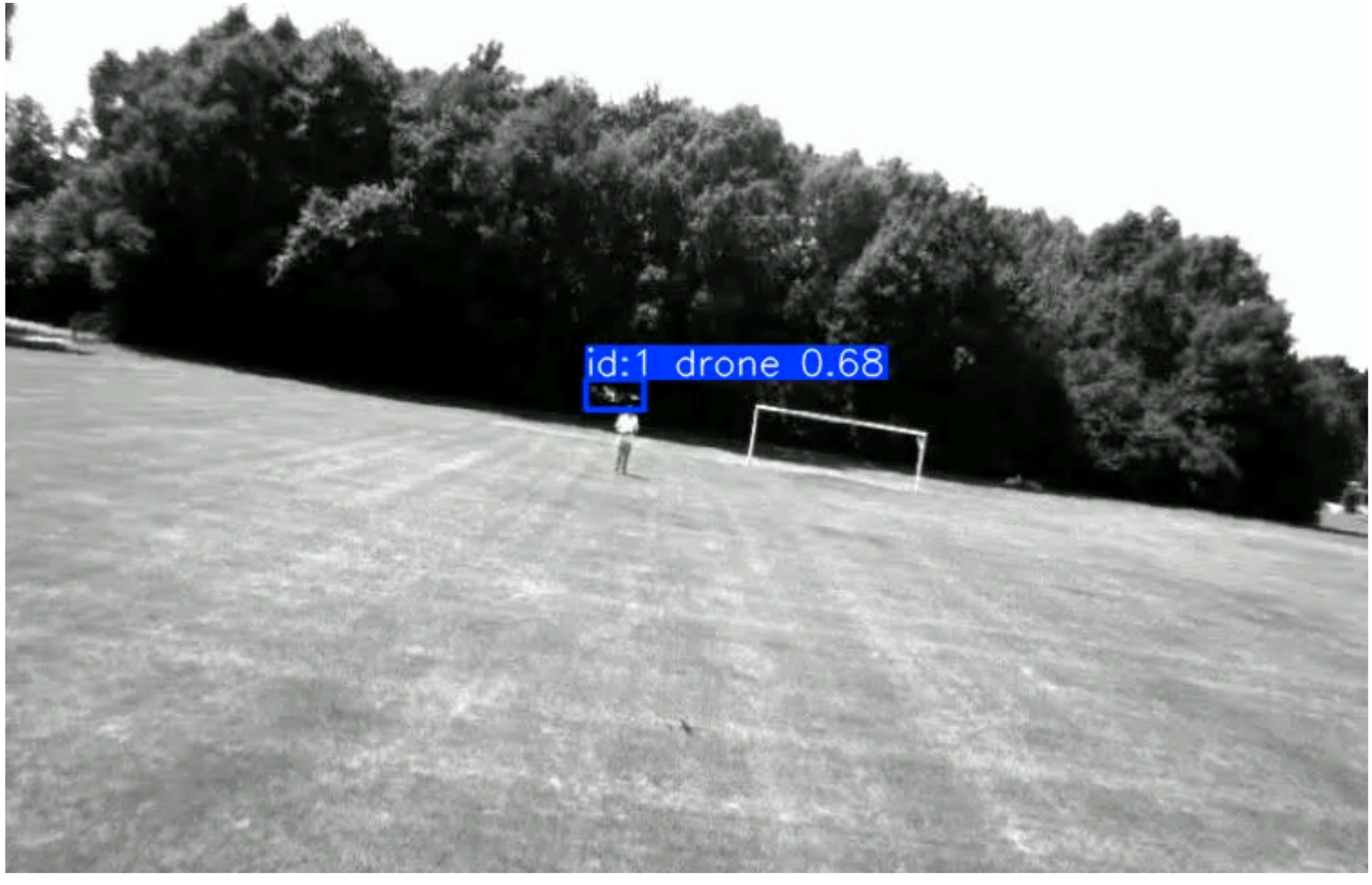




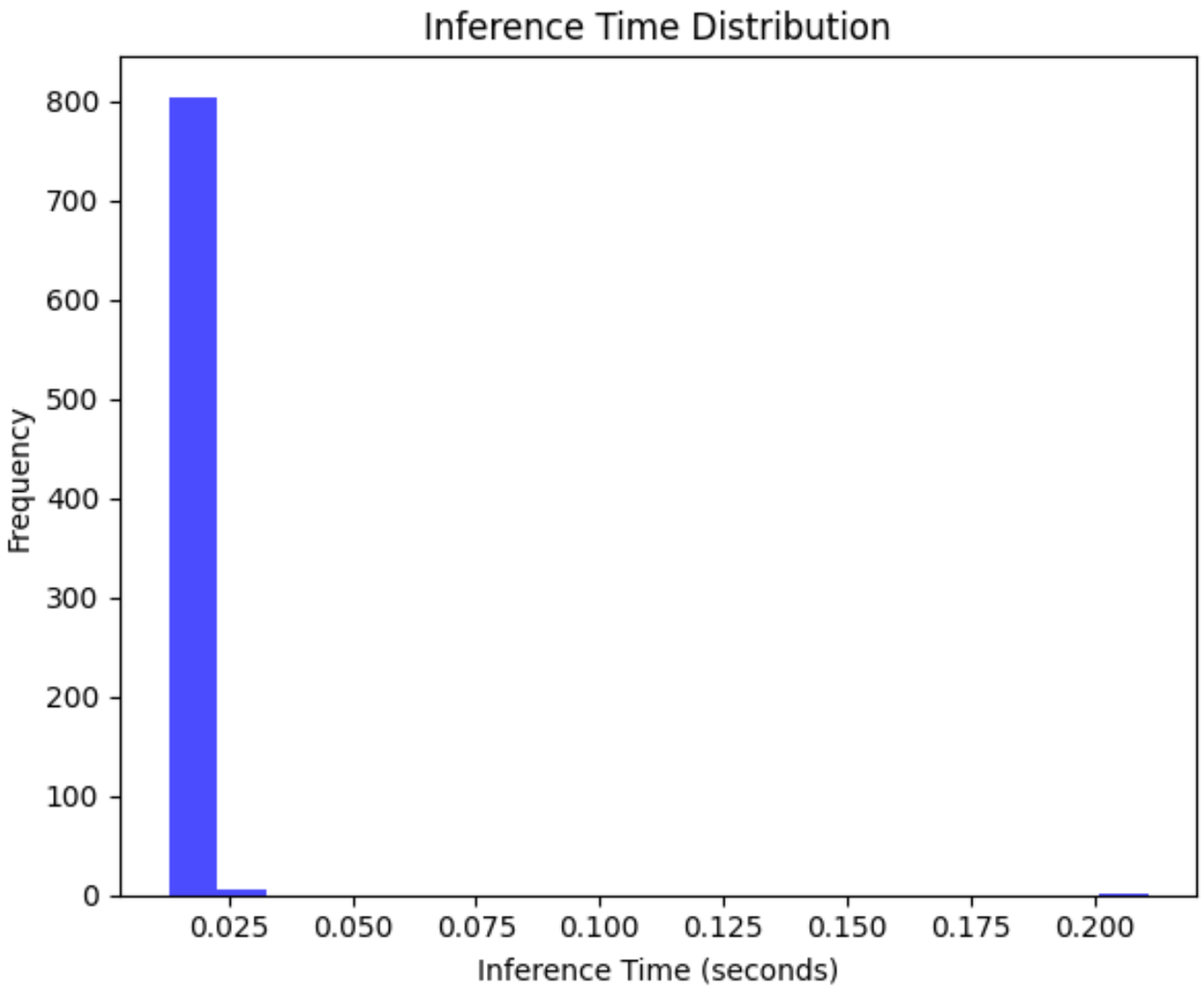


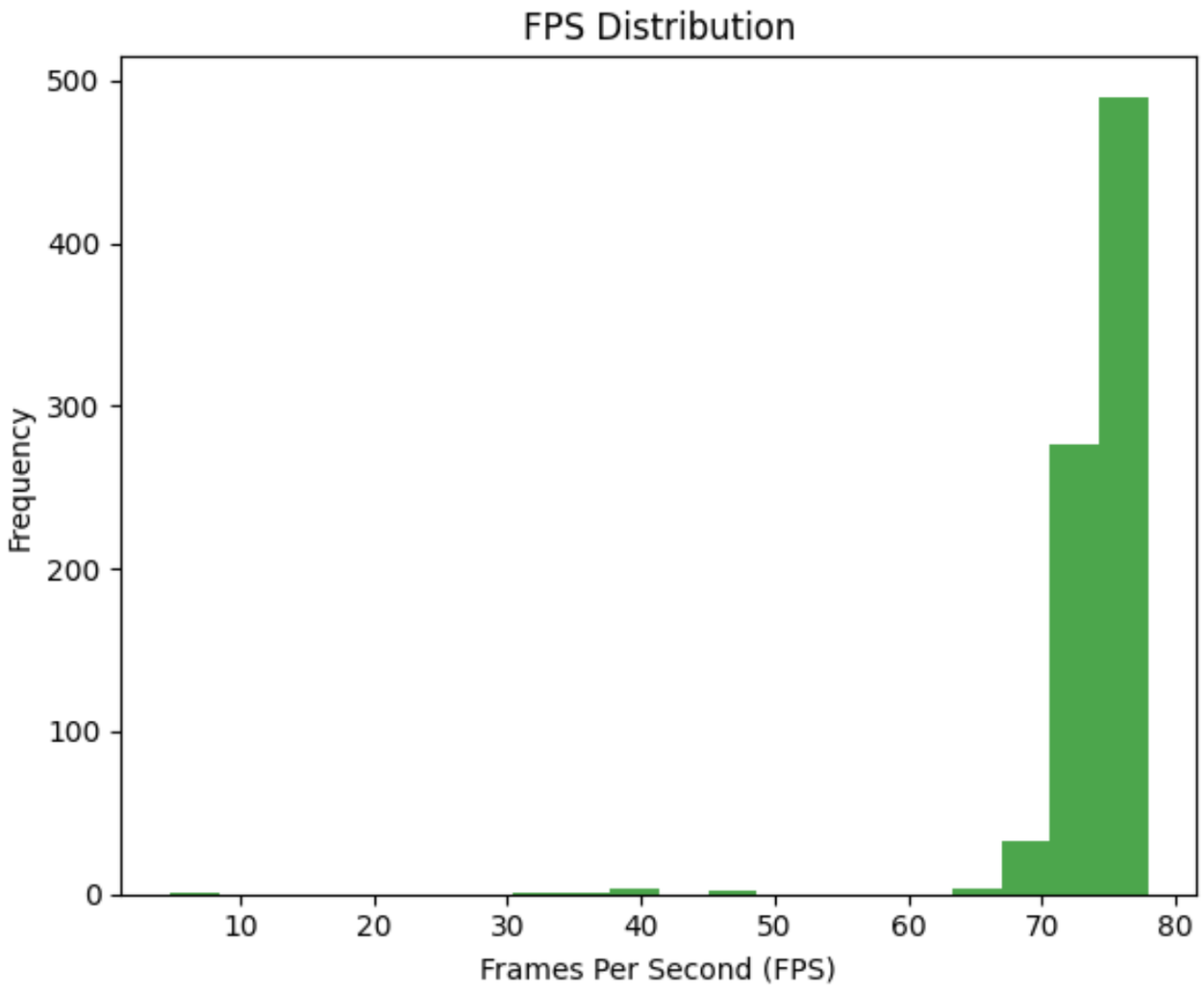


YOLOv11 inference video

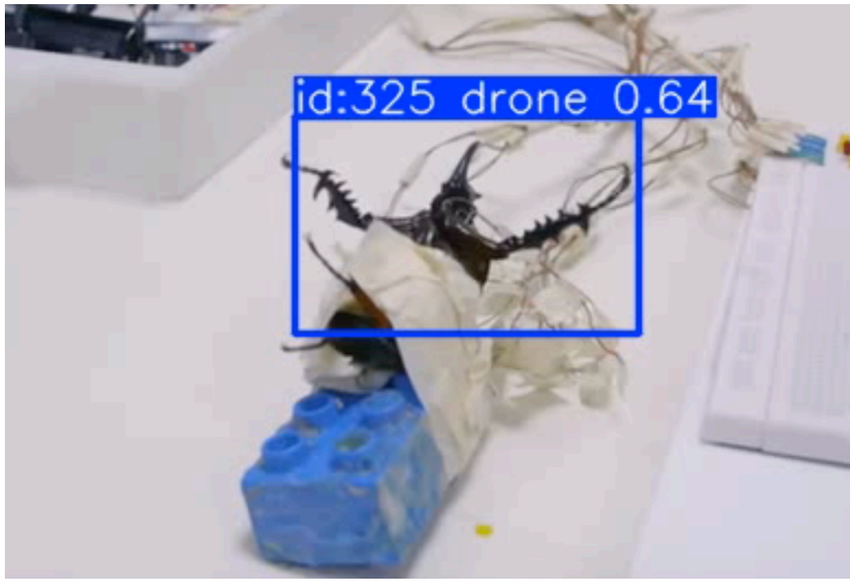


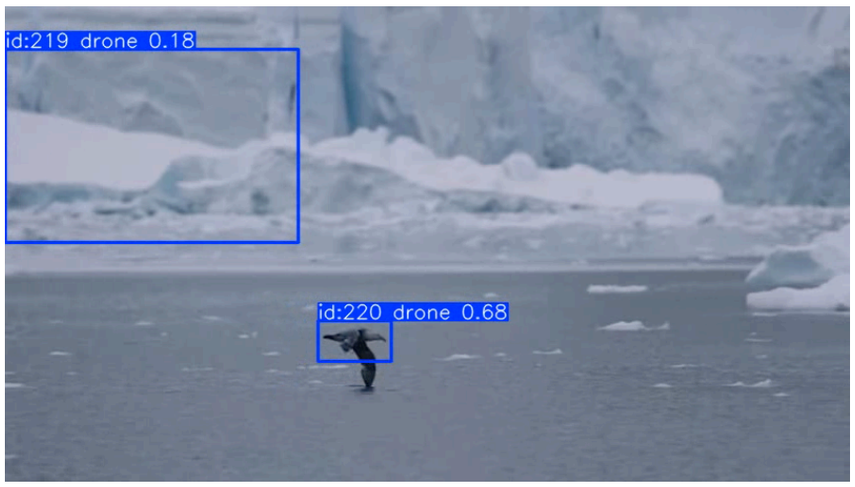
YOLOv11 inference time





YOLOv11 mismatch





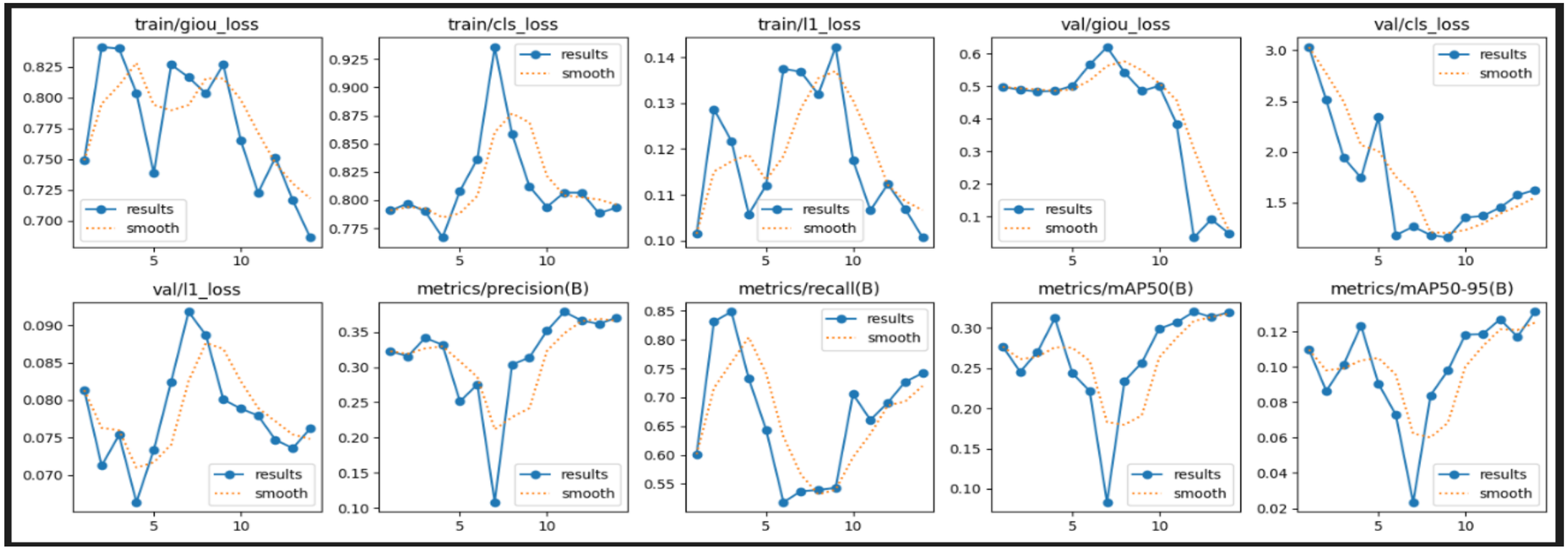
YOLOv11 mismatch

YOLOv11 Conclusion

- Pre-training on VisDrone improves detection performance across all key metrics
- Accuracy has continued to increase throughout the 20 epochs
- From epoch 10 to 15 there is a pullback, after epoch 15 the trend reversed and broke the best value upwards
- Potential improving in mAP50-50 and mAP50-95
- Populate the training dataset to reduce false detection level
- Retrain with higher images resolution

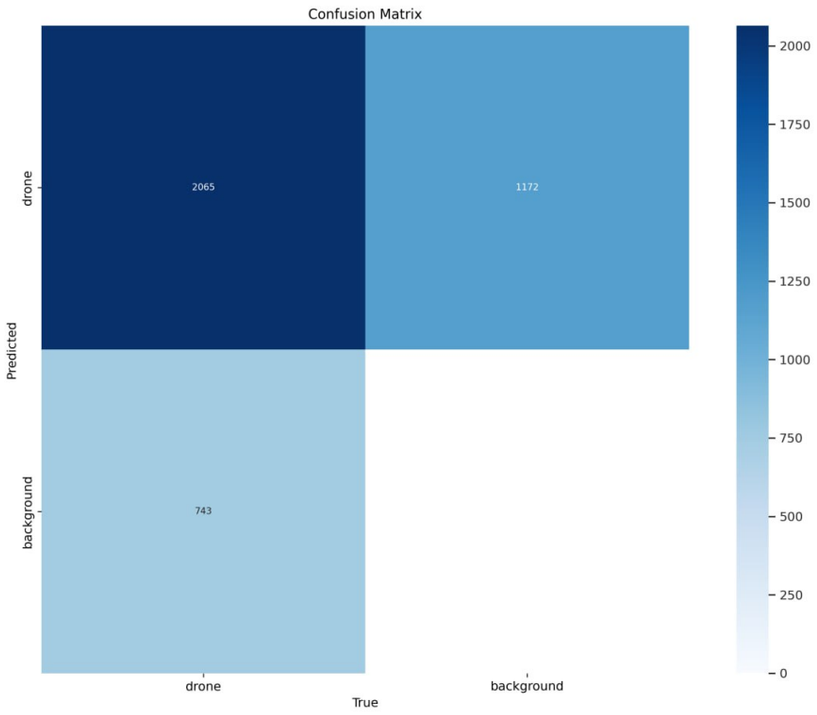
CASE 3: RT-Detr (‘rtdetr-l.pt’)

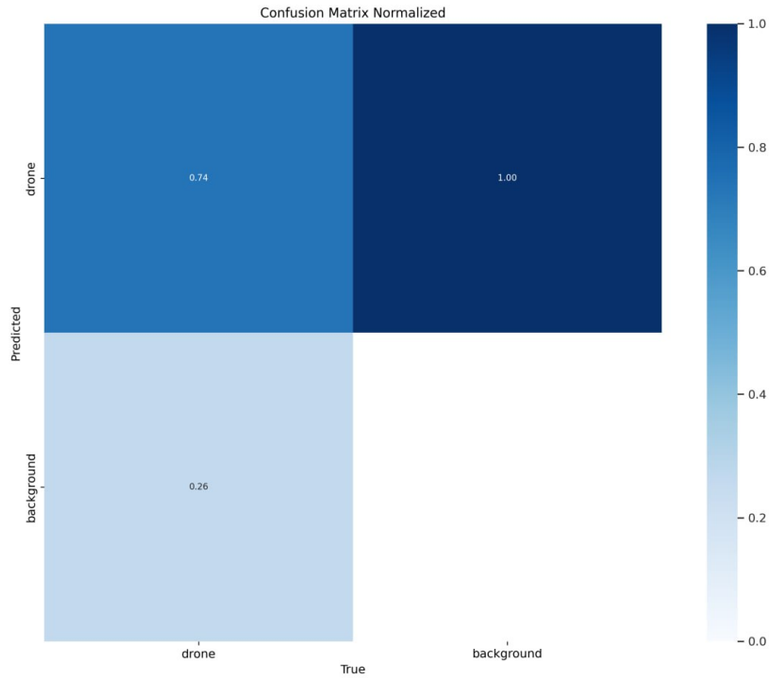
- Ultralytics RT-DETR L size model
- Train for 14 epochs on our dataset
- Best result achieved at 14 epoch



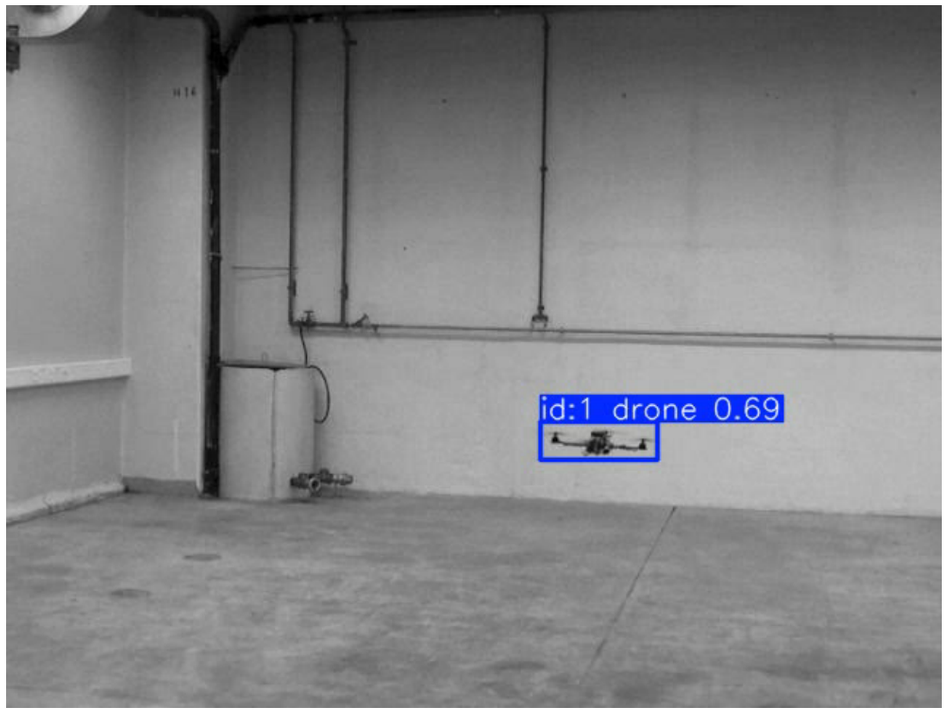
RT-DETR validation

```
Ultralytics 8.3.78 🚀 Python-3.8.10 torch-2.4.1+cu118 CUDA:0 (NVIDIA GeForce GTX 1650, 4096MiB)
rt-detr-l summary: 305 layers, 31,985,795 parameters, 0 gradients, 103.4 GFLOPs
val: Scanning /home/jacksonrr3/hse/hse_dl_project/dataset/yolo/labels/val.cache... 3076 images, 368 backgrounds, 0 corrupt: 100%|██████████| 3076/3076
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 193/193 [03:23<00:00,  1.05s/it]
                   all       3076       2808      0.744       0.64      0.695       0.27
Speed: 0.4ms preprocess, 63.0ms inference, 0.0ms loss, 0.2ms postprocess per image
Results saved to /home/jacksonrr3/hse/hse_dl_project/runs/detect/val3
```



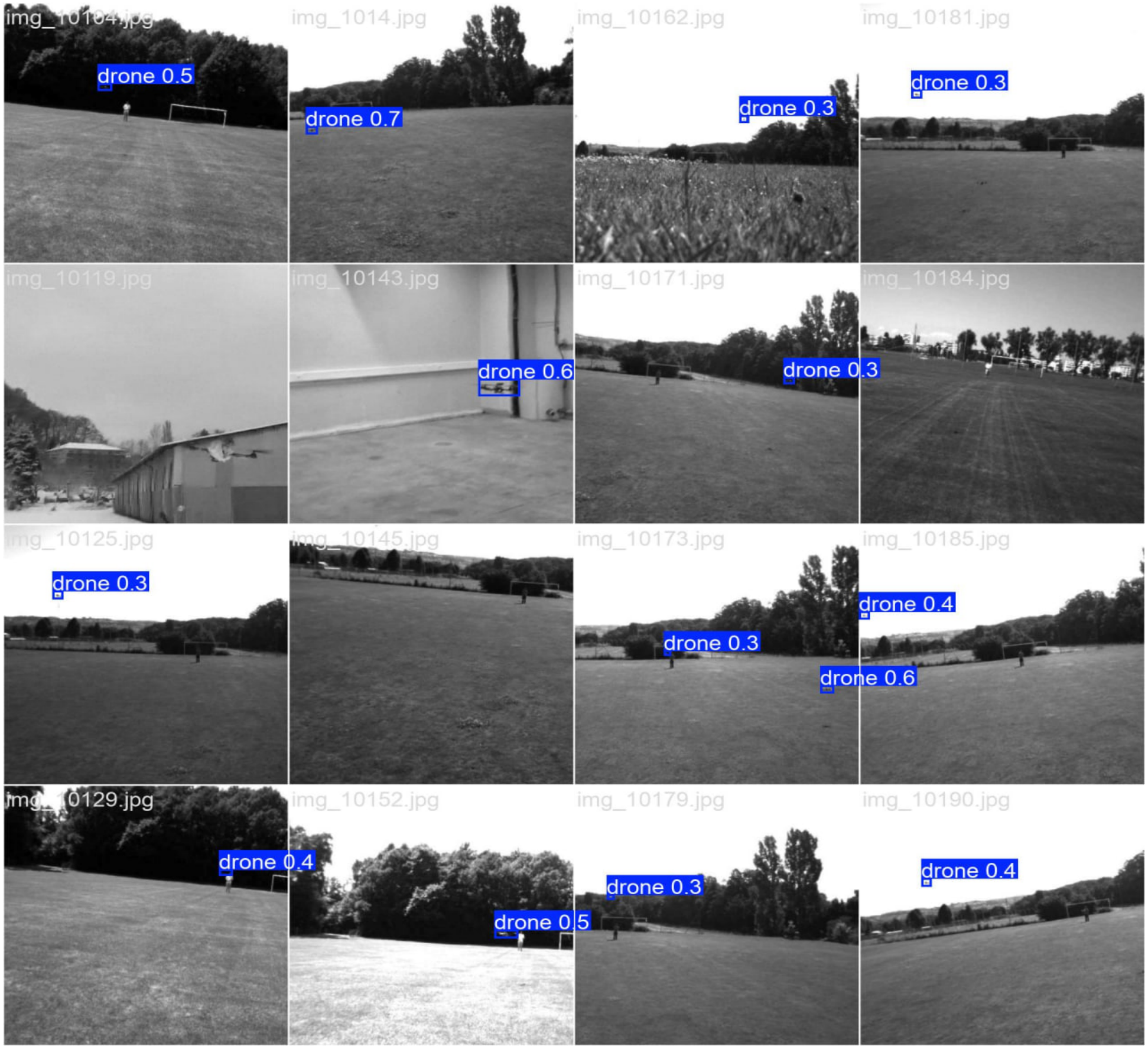


Video detections RT-Detr



[Video_1]

[Video_12]


drone 0.5
drone 0.7
drone 0.3
drone 0.3
drone 0.6
drone 0.3
drone 0.3
drone 0.4
drone 0.3
drone 0.6
drone 0.4
drone 0.5
drone 0.3
drone 0.4