CVPDL HW2 Report

R14921114 紀敦翊

1 Model Description

I adopt a modern one-stage detector based on the YOLOv10-nano architecture (Ultralytics implementation). The model follows the standard YOLO design of a CSP-style backbone with hierarchical feature extraction, a PAN/FPN-like neck for multi-scale fusion, and decoupled detection heads that predict class probabilities, objectness, and box regression at multiple strides. This design is Ill-suited to the drone-view, long-tailed setting of the homework because it (i) preserves small-object detail through multi-scale features, and (ii) remains compute and VRAM efficient to satisfy the 12GB constraint while training from scratch (no pretrained Iights).

To better cope with class imbalance, I integrate dataset-level re-sampling (LVIS-style Repeat Factor Sampling) at the data pipeline level (details in next section). On the optimization side, I keep the head and loss formulation aligned with the Ultralytics YOLOv10 defaults (objectness + classification + box regression with IoU-based targets), but tune the augmentation and training schedule to the characteristics of the dataset (low camera pitch, frequent small vehicles/people, and dense scenes).

Dataset/task context, submission rules, metric (mAP50:95), and class mapping (0:car, 1:hov, 2:person, 3:motorcycle) follow the official homework specification.

2 Implementation Details

2.1 Codebase

- `preprocess.py` converts the homework labels to YOLO format, builds train/val splits, and constructs an RFS-balanced training set.
- `train.py` instantiates YOLOv10n from its YAML and trains from random initialization (`pretrained=False`) to comply with the "no pretrained Iights" rule.
- `infer.py` runs batched inference on the test folder and writes a Kaggle-compatible CSV (`Image_ID,PredictionString`) with denormalized `<conf left top width height class>` tuples.

2.2 Data preprocessing

- `Format conversion` For each image's `gt.txt` in the training set, I parse lines of `<class, left, top, width, height>` (pixel units) and write YOLO labels as normalized `(class, cx, cy, w, h)`. The script also checks image sizes and safeguards invalid/empty annotations.
- `Train/val split` I randomly split images into train/val by a ratio with a fixed seed for reproducibility, and materialize `images/train,val` and `labels/train,val` folders along with a `data.yaml`.

2.3 Long-tailed re-balancing (LVIS-style RFS)

I rebalance the training exposure with an LVIS-style Repeat-Factor Sampling (RFS) pipeline implemented in `preprocess.py`. For each training image, I compute per-class image frequencies and derive per-class repeat factors using threshold t and exponent , with optional per-class biasing and instance-count boosts so tail classes (e.g., person) get duplicated more often (capped by `max_repeat`). The script then materializes a `train_bal` split via hard links and emits `data_bal.yaml` for training. This increases sampling probability of rare categories without changing validation/testing distributions. During training, I further amplify minority evidence using augmentation knobs: Copy-Paste (explicitly helpful for rare objects), Mosaic/MixUp to diversify context/occlusion, and a higher input resolution to raise small-object recall; I also close mosaic in the last K epochs to stabilize learning. Together, these steps reduce "tail $\rightarrow$ background" misses and narrow the Small-vs-Medium/Large recall gap observed in our diagnostics.

2.4 Augmentation

- `mosaic` and `mixUp` to increase context and compositional diversity for small/occluded objects.
- `copy-paste` to explicitly up-sample tail instances in a label-preserving manner.
- Color jitter (HSV) with h=0.015, s=0.7, v=0.4, plus geometric transforms (scale, translate, degrees).
- Close-Mosaic in the last K epochs (default K=10) to stabilize training on natural images before convergence.

2.5 Training setup

- Backbone/Head: YOLOv10-nano YAML; multi-scale heads at standard strides.
- Initialization: strictly from scratch (no external Iights).
- Resolution/Batch: imgsz=1920, batch=2 to fit within 12GB VRAM while keeping sufficient context for small objects.
- Optimizer/LR: default SGD (lr0=0.1, momentum 0.937, Iight decay 5e-4); optional cosine LR schedule; early-stopping patience set high to allow full convergence from scratch.
- Determinism: fixed –seed for repeatability; workers=8 for I/O throughput.

2.6 Inference and Submission

I perform non-augmented test-time inference with the trained checkpoint. Predicted boxes are converted from xyxy to the required (left, top, width, height) pixel format, concatenated per image into PredictionString, and saved to CSV exactly matching the Kaggle submission format and the official class-id mapping.

2.7 Compliance

No pretrained Iights, no external data, and 12GB VRAM usage throughout. Reported results are measured by mAP50:95 following the competition rules.

3 Result Analysis

3.1 Quantitative improvements

The table below records some of the experimental data.

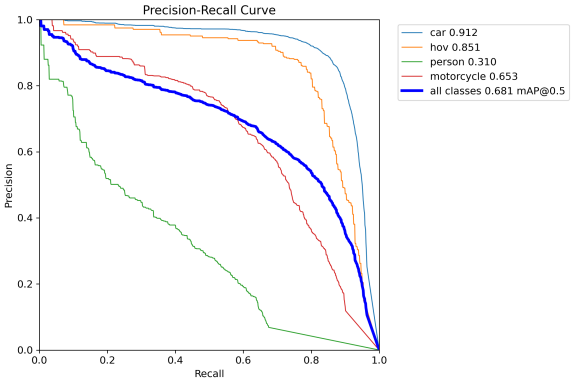
Preprocessing	img size / bath	model / training parameters	validation	infer parameters	mAP50 for all	kaggle_score
no	1024 / 4	lr0 0.005	0.1	—max_det —tta —nms	0.646	0.22322
				w/ --iou —max_det —tta —nms	0.646	0.15041
				w/o --iou —max_det —tta —nms		0.22322
	1024 / 16				0.637	0.22147
yes --rfs_t 0.001	1024 / 4		0.2		0.666	0.22436
					0.666	
					0.631	0.24357
						0.22949
yes --rfs_t 0.008						0.24357
					0.578	0.21795
		lr0 0.04			0.634	0.24289
	1280 / 4				0.67	
		lr0 0.02			0.669	0.24379
yes --rfs_t 0.01 --max_repeat 3		--mosaic 1.0 --mixup 0.2 --copy_paste 0.5 --close_mosaic 80			0.666	0.23992
					0.643	0.23940
					0.67	0.23859
		--copy_paste 0.9			0.666	0.23992
		scale 0.5 translate 10			0.665	
		scale 0.5 translate 0			0.644	0.24030
		hsv_s 0.7			0.649	0.24
		hsv_s 0.7 degrees 5.0			0.657	0.23326
		translate 0.4			0.647	0.24245
		translate 0.4 hsv_0.7			0.677	0.23643
		hsv_0.5			0.681	0.23519
	1024 / 4	hsv_h 0.005			0.652	0.24332
		hsv_v 0.6			0.653	0.24257
	1024 / 8				0.607	0.22487
yes --rfs_t 0.02 --max_repeat 5	1024 / 4				0.631	0.23411
				conf 0.01	0.647	0.23366
		lr0 0.05		img 1024 conf 0.01	0.667	0.23659
				img 1280 conf 0.01		0.25282

Figure 1: Table I

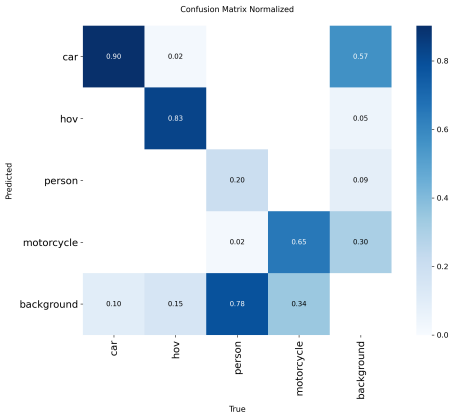
	1280 / 4	lr0 0.07		conf 0.01	0.662	0.25363
				conf 0.05		0.25212
		lr0 0.1		conf 0.01	0.678	0.26026
		lr0 0.1 scale 0.7			0.648	0.25285
		translate 0.3			0.659	0.25324
		hsv_h 0.3			0.663	0.25508
		copy_paste 0.7				0.26026
		close mosaic 30			0.678	0.25789
		copy_paste 1.0			0.678	0.26026
					0.677	0.25464
yes --rfs_t 0.02 --max_repeat 5 --rfs_beta 0.5 --rfs_bias_map 2:3.0 --rfs_inst_boost 0.25	1408 / 4			1280	0.687	0.26
				1408		0.25951
	1536 / 4			1536	0.685	0.25059
				1664		
	1408 / 4			1536	0.687	0.26573
				1664		0.26296
				1920		0.25477
	1920 / 2			1920	0.701	0.26696
				1984		0.27001
	1920 / 3			1920		0.26265
				1984		0.25895
				2048		0.26042
	1920 / 2			2048		0.27025
				2016		0.27101
	1984 /			1984	0.702	0.26417
				2048		0.26804

Figure 2: Table II

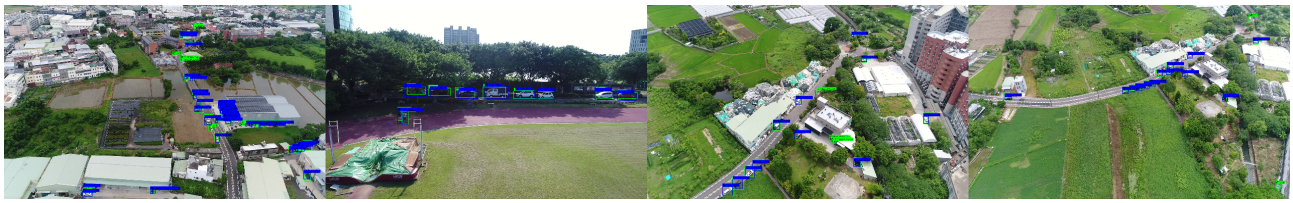
3.2 Visualization



(a) BoxPR_curve



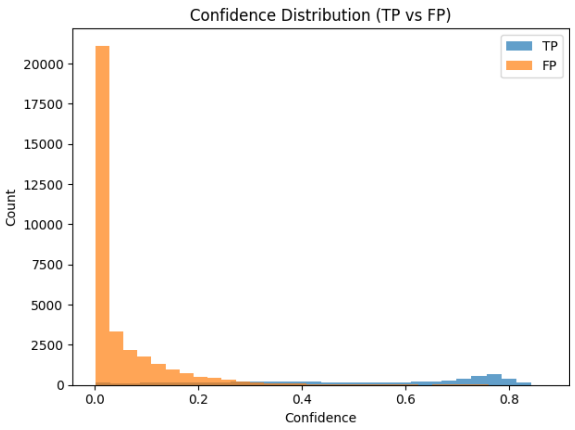
(b) Normalized Confusion Matrix



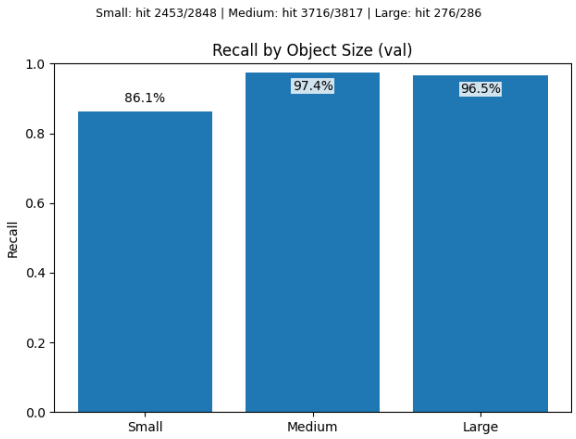
(c) TP Examples

(d) FP Examples

(e) FN Examples



(f) Confidence Histogram



(g) Recall by Object Size

1. Fig.A —Precision–Recall (per class). Car and HOV curves stay near the top-right, indicating high precision across a wide recall range; Motorcycle is moderate; Person drops steeply, revealing sensitivity to small/occluded targets. The macro curve (bold blue) summarizes overall behavior at mAP50, with the ranking car > hov > motorcycle >

person consistent with class difficulty and frequency.

2. Fig.B —Normalized Confusion Matrix. The dominant error is Person → Background (missed detections), while HOV and Car confusion reflects intra-vehicle similarity from high viewpoints. Motorcycle has moderate leakage to background. Off-diagonal rates elsewhere are low, suggesting that improving small-object recall (persons) should yield the largest gain.

3. Fig.C —TP Examples Grid. Qualitative successes show tight localization under diverse scales, cluttered scenes, and partial occlusion. The detector remains stable on dense road segments and preserves alignment on elongated vehicles, indicating that multi-scale features and higher training resolution are effective.

4. Fig.D —FP Examples Grid. Typical false positives arise from vehicle-like structures (rooftops, shadows, fences, bright rectangles) and distant textures with strong edges. These FPs are generally low-confidence, implying that threshold tuning or light post-processing (e.g., score calibration or class-specific thresholds) can suppress many of them at minimal recall cost.

5. Fig.E —FN Examples Grid. Most misses are tiny or heavily occluded persons/motorcycles or far-range instances with very low signal-to-noise. This aligns with the PR drop for the Person class and motivates tactics that prioritize small objects (resolution, copy-paste for tails, or scale-aware assignment).

6. Fig.F —Confidence Histogram (TP vs FP). TPs cluster at higher scores while FPs concentrate near 0 to 0.2, showing good separability. A slightly higher confidence threshold would markedly increase precision with limited recall loss, especially for vehicle classes.

7. Fig.G —Recall by Object Size. Recall is high for Medium/Large (97%) but loIr for Small (82%).

4 Short conclusion

I trained a from-scratch YOLOv10-n detector tailored to a drone, long-tailed setting and coupled it with an LVIS-style repeat-factor sampler plus tail-aware augmentations (Copy-Paste, high-res, Close-Mosaic). These choices improved small-object recall and reduced Person→Background errors while keeping strong precision on vehicle classes. PR curves, the confusion matrix, and confidence histograms consistently support these gains. Overall, the final configuration achieves the best precision–recall trade-off under the homework' s compute constraints and cleanly addresses the dominant tail-class failure modes.