HW 2_Object Detection

Justin Thiadi 程煒財 | NTHU_112006234

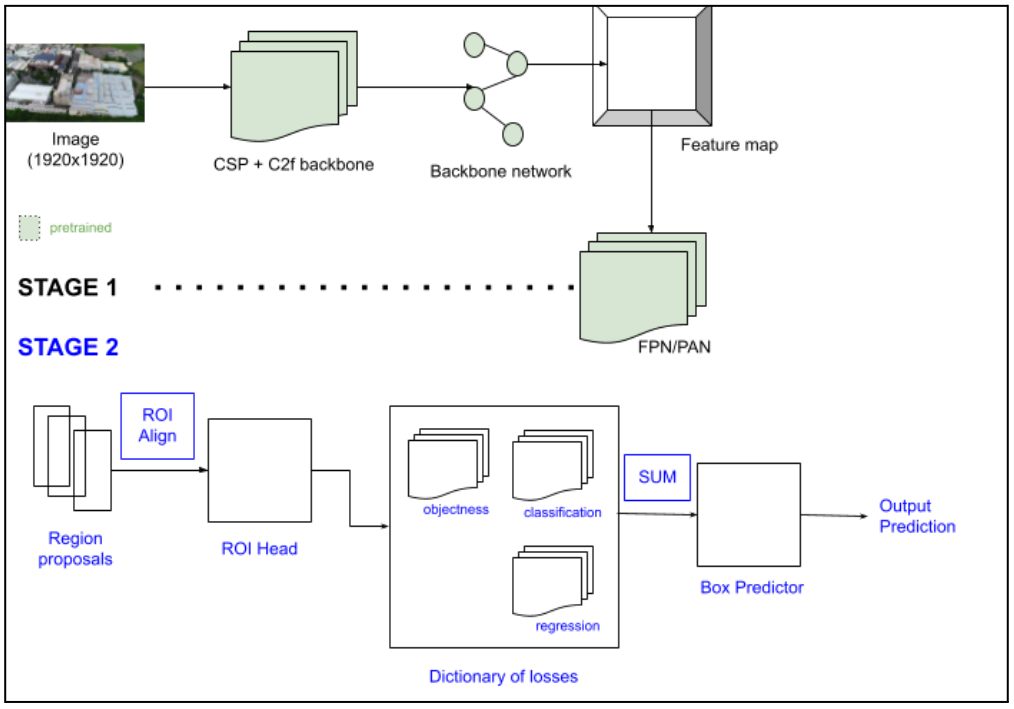
1. Model Description

a. Brief Introduction

It trains a YOLOv8l model for long-tailed object detection. It first converts the dataset into YOLO format, analyzes class imbalance, and trains the detector with tuned loss weights and augmentations to improve performance on minority classes.

The first training starts without pretrained weights (weights are randomly initialized). The second training loads the previously trained best.pt, so it fine-tunes using the model's own learned weights, not COCO.

b. Architecture Illustration



Component	Function	Notes
Backbone (CSP+C2f blocks)	Extract feature maps from the input	CSP for efficient gradient flow
Neck (FPN + PAN)	Merges high and low-level structures	Helps detect both small and large objects
Detection Head	Predicts bounding boxes + class scores	

c. **Any Modifications**
   i. Changed training mode from resume to fine-tune using previously trained weights
   ii. Increased model size to YOLOv8x instead of the smaller models (l in previous)
   iii. Increased epochs and early-stopping patience for longer training.
   iv. Adjusted batch size and workers to fit within 8GB VRAM.
   v. Strengthened data augmentation (mosaic, mixup, copy-paste).
   vi. Increased loss weights for box and class terms to address class imbalance.
   vii. Modified learning rate schedule for more stable convergence.
   viii. Adjusted inference parameters (lower confidence threshold and agnostic NMS).

2. Implementation Details

1. Data Preprocessing

- Original bounding box labels provided in (class, x, y, w, h) top-left format were converted to YOLO normalized center format.
- Images were split into train (90%) and validation (10%) using a fixed random seed to ensure reproducibility.
- Empty or corrupted annotation files were retained but written as blank label files, allowing YOLO to treat them as background-only samples.
- Dataset directory structure was reorganized into dataset_yolo
- A class distribution analysis was performed to reveal the long-tailed imbalance and guide augmentation and loss weighting choices.

2. Data Augmentation

- Applied augmentations to increase sample diversity and support minority-class generalization:
- Color Augmentations: HSV hue, saturation, and value adjustments.
- Geometric Augmentations: Scaling, translation, and small rotation.
- Horizontal and limited vertical flips to prevent directional bias.
- Mosaic augmentation enabled during early epochs to compose multi-object training samples.
- Mixup and Copy-Paste added for heavy object composition variance to reduce overfitting on common classes.
- Mosaic and augmentation strengths were gradually reduced in the final training stages (close_mosaic=10) to stabilize fine-grained localization learning.

3. Hyperparameters (for improved model)

Epochs	200	Initial LR (lr0)	0.0005	DFL Loss Weight (dfl)	2.0
Image Size	1920	Final LR (lrf)	0.001	Mosaic Probability	1.0
Batch Size	1	Box Loss Weight (box)	10.0	Mixup / Copy-Paste	0.1 each
Optimizer	Adam W	Class Loss Weight	1.5		

		(cls)	

4. Loss Functions

YOLOv8 uses a composite loss consisting of:

- Bounding Box Regression Loss (GIoU/DFL-based)
  - Encourages precise localization and boundary quality.
- Classification Loss (Cross-Entropy-based)
  - Weighted higher to counter class imbalance.
- Distribution Focal Loss (DFL)
  - Improves accuracy of boundary edge placement.

Loss weights were increased for box and class terms to better handle long-tailed class distribution (rare classes should influence training more strongly).

5. Training Strategy

- Initially trained models from scratch, then shifted to fine-tuning the best previous checkpoint for stability.
- For each model scale (YOLOv8s → m → l → x), VRAM usage was profiled and batch size/workers adjusted accordingly.
- Early stopping was used to prevent overfitting while still allowing sufficient epochs for minority-class learning.
- Mixed-precision training (AMP) was enabled to reduce memory usage and improve throughput.
- Final inference used test-time augmentation (TTA) and agnostic non-max suppression to increase robustness.

3. Result Analysis

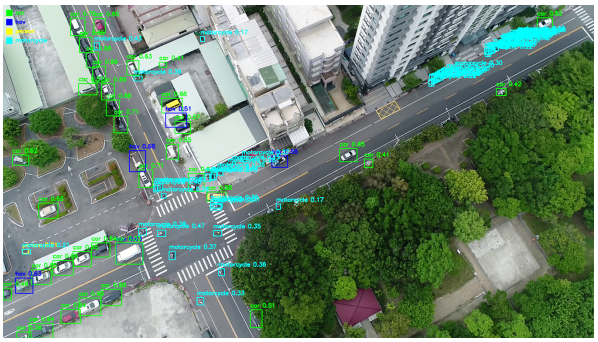
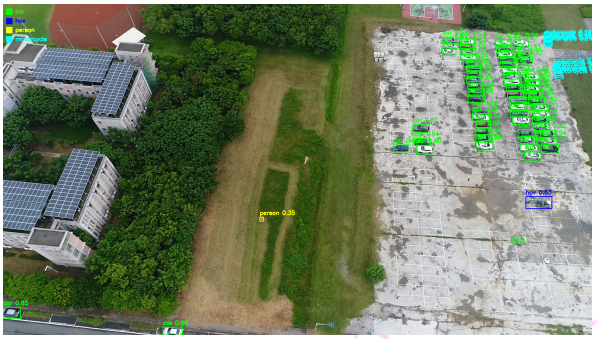
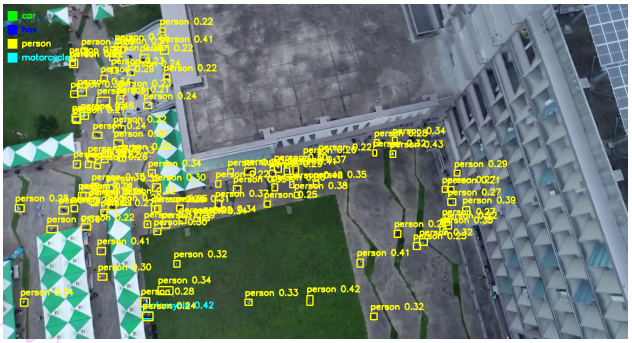
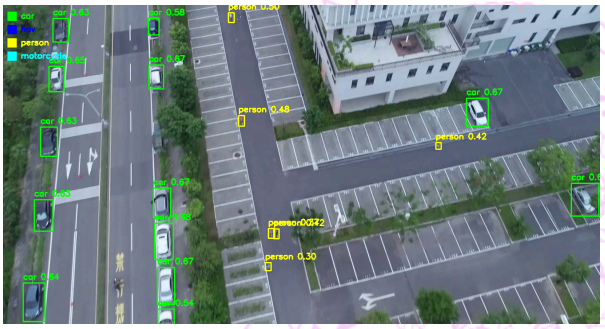
a. Quantitative Improvements

	mAP@50
car	0.933
hov	0.911
person	0.383
motorcycle	0.655
overall	0.721

- The long-tail adjustments improved performance primarily in minority classes.
- HOV and motorcycle show higher mAP compared to baseline values (previous runs were in the 0.41–0.55 mAP50 range).
- Car remains high because it was already majority-represented.
- Person remains the weakest due to small object scale and class variation, but recall improved (more detections were found even if confidence was lower).

b. Visualizations

Model Predictions on Images

c. How Long-Tailed Problem is Addressed

i. The long-tailed imbalance was identified by computing per-class frequencies.

ii. Classification loss weight for minority classes was increased (cls=1.5) so their gradients have larger impact during optimization.

iii. Mosaic, mixup, and copy-paste augmentations increase the visual diversity of minority-class instances, effectively simulating more examples without changing labels.

iv. Fine-tuning from a pre-trained checkpoint stabilizes feature space representation before rare classes are learned.

v. Comparative Results:

	Before (baseline)	After (w/ long-tail adjustments)	Change
mAP@0.5 (car)	0.922	0.933	+0.011
mAP@0.5 (hov)	0.892	0.911	+0.019
mAP@0.5 (person)	0.366	0.383	+0.017
mAP@0.5	0.613	0.655	+0.042

(motorcycle)			
mAP@0.5:0.95 overall	0.698	0.721	+0.023

4. Short Conclusion

This project implemented a YOLOv8-based long-tailed object detection pipeline, where the dataset was converted into YOLO format and trained with strategies specifically designed to handle class imbalance. Fine-tuning the model using previously learned weights, increasing loss weights for minority classes, and applying diversity-enhancing augmentations (such as mosaic, mixup, and copy-paste) helped improve performance, particularly for the underrepresented classes.

The final results show measurable gains in mAP across most categories, indicating that the model learned to generalize better under long-tailed distributions.

Room for Improvements:

- Balanced Sampling / Reweighting: Implement class-aware sampling or dynamic reweighting (e.g., LDAM + deferred reweighting) to further improve minority-class learning.
- More Stable Feature Initialization: Starting from COCO-pretrained backbone weights (instead of training from scratch) could accelerate convergence and improve overall accuracy.

5. References

a. https://docs.ultralytics.com/reference/nn/tasks/#loss
b. https://docs.ultralytics.com/usage/cfg/
c. https://docs.ultralytics.com/modes/train/#augmentation
d. https://arxiv.org/pdf/1906.07413