Crackathon - Road Damage Detection Challenge 2026

Team name: Inception.JS
Model filename: best.pt
Team Members: Tanmay Bhatkar, Sachi Jadhav, Durva Kadam, Anish Kalbhor

REPORT

Abstract

This report summarizes the methodology, training strategy, augmentation experiments, hyperparameter tuning, and post-processing trials used to improve a YOLOv9-compact object detection baseline. Two compact models were trained at different input resolutions (648 and 768 px) for 80 epochs using the same base configuration. An ensemble of the two models was attempted but produced diminishing returns and was abandoned. To improve the single-model baseline we experimented with a focused set of augmentation and regularization techniques applied during the final 20 epochs together with a targeted learning-rate refinement. The report presents the rationale, implementation details, results (training curves attached), and recommendations for further work.

1. Model architecture: brief overview

We used the YOLOv9 compact architecture as the backbone object detector. The compact variant is designed to trade off some representational capacity for inference efficiency, while retaining detector heads and anchor-free prediction structure typical of the YOLO family.

2. Training strategy and baseline run

Both models (trained at 648×648 and 768×768 nominal sizes) were trained from the same base configuration for 80 epochs. Key points of the baseline training:

- Training was performed on a dataset with 768×768 pixel images using batch size 16 over 80 epochs with a patience of 30 epochs for early stopping.

Optimizer & Learning Rate Strategy:

SGD optimizer was selected with momentum 0.937 and weight decay 0.0005 for stable convergence. The learning rate schedule started at lr0=0.01 and decayed to lrf=0.0001 (100× reduction) using cosine annealing (cos_lr=true). A 3-epoch warmup phase gradually increased the learning rate from warmup_bias_lr=0.1 using warmup_momentum=0.8 to prevent early training instability.

Loss Function Weights

The multi-task loss function balanced three components: box loss (7.5) for bounding box regression, classification loss (0.5) for class prediction, and distribution focal loss (1.5) for anchor-free detection. These weights prioritized accurate localization while maintaining classification performance

Data Augmentation Strategy: Domain-Aware Design

Geometric Augmentations (Orientation-Preserving)

Critical design decision: No rotation (degrees=0.0) and no shearing (shear=0.0) were applied because crack orientation is semantically meaningful—rotating a longitudinal crack 90° would incorrectly transform it into a transverse crack. Vertical flipping was disabled (flipud=0.0) as roads maintain consistent gravity orientation, while horizontal flipping (fliplr=0.5) was enabled as left-right symmetry is valid. Moderate translation (0.1) and scaling (0.5) augmentations provided spatial variability without altering crack semantics.

Mosaic & Mixing Strategies

Mosaic augmentation (1.0) was always enabled until 10 epochs before completion (close_mosaic=10), creating multi-scale training samples. Notably, mixup, cutmix, and copy_paste were disabled (0.0) to preserve spatial context—these techniques can create physically unrealistic crack placements that violate how damage actually manifests on road surfaces following stress patterns.

Regularization & Stability

Deterministic training (seed=42, deterministic=true) ensured reproducibility. Mixed precision training (amp=true) accelerated computation while maintaining numerical stability. No dropout was applied initially (dropout=0.0), relying instead on data augmentation and weight decay for regularization.

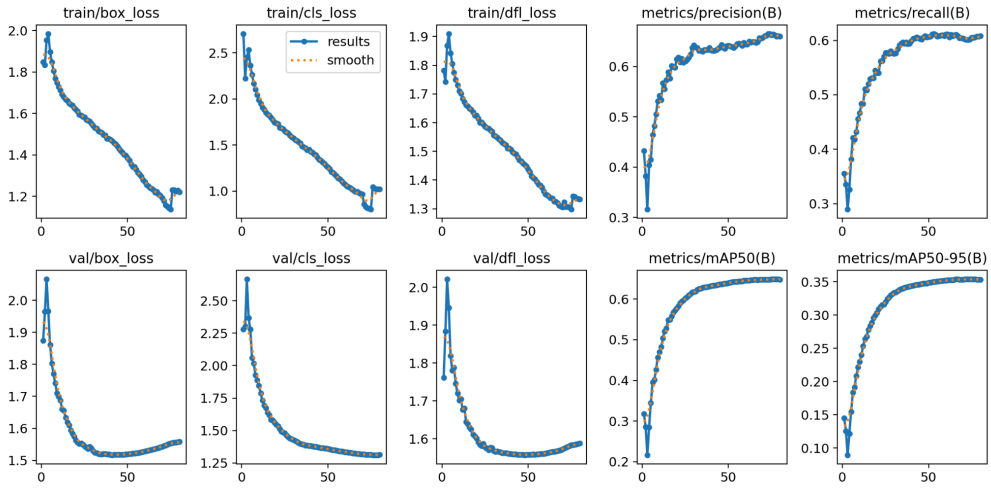
Validation & Checkpointing

Models were validated every epoch with IoU threshold 0.7 for Non-Maximum Suppression and evaluated on the validation split. Checkpoints were saved every 2 epochs with the best model selected based on mAP@0.5:0.95. Training used 6 worker threads for data loading on GPU device 0.

```
Validating C:\Users\Cerelab\Desktop\GroupIJS2\outputs\runs\yolov9c_ep_20260103_112944\weights\best.pt...
Ultralytics 8.3.243  Python-3.12.1 torch-2.5.1+cu121 CUDA:0 (NVIDIA GeForce RTX 4070 Ti SUPER, 16376MiB)
YOLOv9c summary (fused): 156 layers, 25,323,103 parameters, 0 gradients, 102.3 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 188/188 3.5it/s 53.9s0.3ss
                   all       6000      10443      0.662      0.605      0.649      0.354
    Longitudinal_Crack       2171       4093      0.665      0.576      0.621      0.352
       Transverse_Crack       1209       1830      0.645      0.591      0.634      0.314
        Alligator_Crack       1353       1698      0.668      0.625      0.693      0.388
      Other_Corruption       1170       1737      0.689      0.747       0.76      0.483
               Pothole        577       1085      0.646      0.487      0.538      0.231
Speed: 0.1ms preprocess, 5.4ms inference, 0.0ms loss, 1.0ms postprocess per image
Results saved to C:\Users\Cerelab\Desktop\GroupIJS2\outputs\runs\yolov9c_ep_20260103_112944

============================================================
✅ TRAINING COMPLETE
============================================================
```

BASELINE RESULTS



BASELINE RESULTS GRAPH

Later we resumed for improvement on baseline from checkpoint saved 20 epoch earlier, with certain changes to arrive at different conclusions which are mentioned in section 3 and 4 of this report.

3. Focused augmentation & refinement applied to improve the baseline

To increase model generalization and final detection performance, we adopted a *refinement stage* during the last 20 epochs of training. The idea was to stabilize early training and then apply stronger regularization / targeted augmentations at the end to push generalization without destabilizing convergence.

The key interventions applied during this 20-epoch finetune were:

- Learning-rate refinement: initial LR was set high for earlier training, then during the final 20 epochs we operated with a refined learning-rate schedule starting at LR_0 = 0.005 and annealing toward a small final factor LRF = 0.00005. This slow, low final-rate regime encourages fine-grained weight adjustments and reduces over-shooting near minima.
- MixUp augmentation (soft image mixing): a mild mixup probability/effect of ~5% to encourage robustness to partial occlusion and varied backgrounds.
- Random erasing: applied with a strength of ~10%, forcing the model to rely less on single-object context cues and improving robustness to missing parts.
- HSV jittering: photometric augmentation with modest hue/saturation/value perturbations (hue ≈ 0.015, saturation scale ≈ 0.5, value scale ≈ 0.2) to increase color invariance.
- Label smoothing: small smoothing (≈ 0.05) on class labels to reduce overconfidence and improve calibration.
- Close-mosaic augmentation: an aggressive mosaic-like augmentation with a parameter (close_mosaic = 4) to present denser, multi-object configurations and encourage detection in crowded or overlapping settings.

These interventions were applied together as a short, targeted finetune rather than throughout the whole 80-epoch run. The rationale was to let the model learn stable feature representations first, then use stronger augmentation and a low learning rate to fine-tune decision boundaries and improve generalization.

FINAL METRICS:

mAP50: 0.668

mAP50-95: 0.388

Precision:0.653

Recall:0.619

```
Ultralytics 8.3.250 🚀 Python-3.12.12 torch-2.9.0+cu126 CUDA:0 (Tesla T4, 15095MiB)
YOLOv9c summary (fused): 156 layers, 25,323,103 parameters, 0 gradients, 102.3 GFLOPs
val: Fast image access ✅ (ping: 0.0±0.0 ms, read: 1209.7±501.0 MB/s, size: 198.7 KB)
val: Scanning /root/.cache/kagglehub/datasets/anulayakhare/crackathon-data/versions/1/rand
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all       6000      10443      0.653      0.619      0.668      0.388
    Longitudinal_Crack       2171       4093       0.65      0.588      0.644      0.392
      Transverse_Crack       1209       1830      0.636      0.605      0.652      0.346
       Alligator_Crack       1353       1698      0.664       0.64      0.704      0.418
      Other_Corruption       1170       1737      0.683      0.763      0.767      0.515
               Pothole        577       1085      0.631      0.501       0.57      0.266
Speed: 1.3ms preprocess, 39.5ms inference, 0.0ms loss, 0.7ms postprocess per image
```

4. Discussion(techniques that improved the baseline)

We have uploaded the code which was used to infer on images (Inference.py) More on the results:

1. Final-stage LR refinement was the most effective single change. Reducing the learning rate and annealing to a tiny final factor during the last 20 epochs allowed the network to settle into better minima and improved metric stability.
2. Combined augmentations in the finetune (mixup + erasing + close-mosaic) added complementary robustness: mixup regularized classification boundaries; random erasing reduced over-reliance on specific object parts; close-mosaic increased the model's ability to cope with crowded scenes and multiple nearby objects.
3. Photometric jittering and label smoothing further contributed to small but consistent gains in mAP and reduced overconfidence in predictions.
4. Ensemble attempt - while conceptually attractive, failed to yield net gains likely due to model similarity. This highlights that ensembling reductions in error require diverse error modes or significantly different training regimes.