NIGHT-VISION OBJECT DETECTION

Using Synthetic Data

Generation and Domain Adaptation

BTech Final Year Project

Team Members:

Aditya Jaiswal (B22CS025)
Gaurav Manish (B22CS079)
Vedant Funde (B22AI041)

Supervisor
Prof. Hardik Jain

Department of Computer Science and Engineering

November 2025

Contents

1 Executive Summary

Object detection tasks are essential for 24/7 operations in search and rescue, surveillance, and autonomous navigation. However, standard object detectors fail catastrophically on night-vision footage due to fundamental differences in visual characteristics. This project addresses the critical data bottleneck by synthesizing an entire labeled night-vision dataset using Pix2Pix GAN, then fine-tuning YOLOv8 and Faster R-CNN on this synthetic data.

Key Results:

- YOLOv8-Large: mAP@50 = 12.2%, mAP@50-95 = 6.0%
- Faster R-CNN: mAP@50 = 3.6%, successful convergence
- Dataset: 6,471 training and 548 validation images
- Real-time capability: 11.4 ms/image (87.7 FPS) with YOLOv8

2 Introduction

2.1 Problem Statement

Drones require object detection for round-the-clock operations. When deployed at night, conventional models fail due to:

- **Domain Gap:** Night-vision images have different visual characteristics (low contrast, grayscale, noise, thermal effects)
- **Data Scarcity:** Large-scale, annotated night-vision drone datasets do not exist
- **Cost Barrier:** Collecting night-vision data requires expensive equipment
- **Small Objects:** Aerial imagery contains extremely small objects (10–30 pixels)

2.2 Proposed Solution

Instead of collecting expensive real night-vision data, we:

1. Generate synthetic night-vision images using Pix2Pix GAN
2. Scale and convert bounding box annotations to match resized images
3. Fine-tune YOLOv8 and Faster R-CNN on synthetic data
4. Achieve domain adaptation without real night-vision datasets

2.3 Dataset Overview

We used the VisDrone2019 dataset:

- 6,471 training images + 548 validation images
- 10 object classes: pedestrian, person, car, van, bus, truck, bicycle, motor, tricycle, awning-tricycle
- Original resolution: variable (up to 1920×1080)
- Processed resolution: 256×256 pixels
- Total annotations: 50,000+

3 Methodology

3.1 Phase 1: Day-to-Night Image Translation

3.1.1 Synthetic Night-Vision Generation

Before training Pix2Pix, we created pseudo-night-vision images using image processing:

1. Convert RGB to grayscale
2. Reduce brightness by 30–50%
3. Apply histogram equalization
4. Add Gaussian noise ($\sigma = 15$–25)
5. Apply vignette effect
6. Resize to 256×256 pixels

3.1.2 Pix2Pix Architecture

Pix2Pix is a conditional GAN for paired image-to-image translation:

- **Generator:** U-Net with skip connections (8 encoder + 8 decoder layers)
- **Discriminator:** PatchGAN (classifies 70×70 patches as real/fake)
- **Loss Function:** Adversarial loss + L1 reconstruction loss

3.1.3 Training Configuration

Parameter	**Value**
Optimizer	Adam
Learning Rate	0.0002
Beta 1, Beta 2	0.5, 0.999
Batch Size	1
Epochs	200
Lambda (L1 weight)	100
Image Size	256×256
GPU	Tesla T4 (16GB)

Table 1: Pix2Pix Training Configuration

Output: 6,471 synthetic night-vision training images with preserved bounding boxes.

3.2 Phase 2: Annotation Scaling and Conversion

3.2.1 Coordinate Scaling Algorithm

The key engineering challenge was scaling bounding box coordinates from original resolution to 256×256:

Scaling factors:

$$\text{scale}_x = \frac{256}{w_{\text{orig}}} \tag{1}$$

$$\text{scale}_y = \frac{256}{h_{\text{orig}}} \tag{2}$$

For each bounding box (x, y, w, h):

$$x_{\text{new}} = x \times \text{scale}_x \tag{3}$$

$$y_{\text{new}} = y \times \text{scale}_y \tag{4}$$

$$w_{\text{new}} = w \times \text{scale}_x \tag{5}$$

$$h_{\text{new}} = h \times \text{scale}_y \tag{6}$$

3.2.2 Format Conversion

We converted annotations to two formats:

- **YOLO Format:** Normalized coordinates in .txt files (one per image)
- **COCO Format:** Absolute coordinates in JSON (single file with metadata)

Processing statistics: 7,019 images processed, 50,000+ annotations scaled, 300+ duplicates removed.

3.3 Phase 3: Model Training

3.3.1 YOLOv8-Large

Started from best.pt (pre-trained on daytime VisDrone), fine-tuned for 50 epochs.

Architecture: CSPDarknet backbone, PANet neck, anchor-free detection head (43.6M parameters)

Hyperparameters:

Parameter	Value
Epochs	50
Batch Size	16
Image Size	256×256
Optimizer	AdamW
Learning Rate	0.000714
Weight Decay	0.0005
Augmentation	Mosaic, Flip, Translate, Scale

Table 2: YOLOv8 Training Hyperparameters

3.3.2 Faster R-CNN with MobileNetV3

Started from COCO pre-trained weights, trained for 20 epochs.

Architecture: MobileNetV3-Large backbone with FPN, RPN, two-stage detection head ($\sim$15M parameters)

Hyperparameters:

Parameter	Value
Epochs	20
Batch Size	16
Optimizer	SGD
Learning Rate	0.005
Momentum	0.9
Weight Decay	0.0005
LR Scheduler	StepLR (step=10, gamma=0.1)
Num Classes	11 (10 objects + background)

Table 3: Faster R-CNN Training Hyperparameters

Custom COCO Dataset Implementation:

```python
class CustomCocoDataset(CocoDetection):
    def __getitem__(self, idx):
        img, target_list = super().__getitem__(idx)
        target = {'image_id': torch.tensor(self.ids[idx])}

        if not target_list:
            target['boxes'] = torch.zeros((0, 4))
            target['labels'] = torch.zeros(0, dtype=torch.int64)
        else:
            boxes = torch.tensor([obj['bbox']
                                  for obj in target_list])
            # Convert (x,y,w,h) to (x1,y1,x2,y2)
            boxes[:, 2] += boxes[:, 0]
            boxes[:, 3] += boxes[:, 1]
            target['boxes'] = boxes
            target['labels'] = torch.tensor(
                [obj['category_id'] for obj in target_list],
                dtype=torch.int64
            )
        return img, target
```

4 Results and Evaluation

4.1 YOLOv8 Performance

4.1.1 Overall Metrics

Validation on 548 images:

Metric	**Value**
mAP@50	12.2%
mAP@50-95	6.0%
Precision	0.254
Recall	0.132
Inference Time	11.4 ms/image
FPS	87.7

Table 4: YOLOv8 Performance Metrics

4.1.2 Per-Class Performance

Class	**mAP@50**	**Precision**	**Recall**
Van	46.3%	0.478	0.474
Awning-tricycle	22.5%	0.457	0.219
Bus	15.8%	0.274	0.193
Truck	9.2%	0.215	0.120
Tricycle	7.3%	0.275	0.074
Pedestrian	6.4%	0.208	0.073
Person	6.0%	0.317	0.044
Motor	5.0%	0.172	0.080
Bicycle	2.6%	0.093	0.034
Car	1.1%	0.055	0.011

Table 5: Per-Class Performance

Best performance on van class (46.3% mAP@50) due to large instance count (14,058) and larger object size. Small objects struggle due to low contrast and tiny pixel representation in synthetic night-vision.

4.2 Faster R-CNN Performance

Metric	**Value**
mAP@50	3.6%
Final Training Loss	0.0694

Table 6: Faster R-CNN Performance

Model achieved stable convergence (loss: 0.0876 → 0.0694), validating synthetic data viability.

4.3 Model Comparison

Aspect	**YOLOv8**	**Faster R-CNN**
Pre-training	VisDrone (day)	COCO (general)
Parameters	43.6M	∼15M
mAP@50	12.2%	3.6%
Inference Time	11.4 ms	18.8 ms
FPS	87.7	∼53
Real-time	Yes	Marginal

Table 7: Model Comparison

YOLOv8 outperforms due to VisDrone pre-training (same aerial viewpoint), enabling faster domain adaptation compared to COCO's ground-level perspective.

5 Challenges and Solutions

5.0.1 Challenge 1: Data Format Heterogeneity

Problem: VisDrone (CSV), YOLO (normalized), COCO (JSON) formats require custom conversion. Original coordinates don't match 256×256 resized images.
Solution: Implemented coordinate scaling with per-image scale factors. Validated through visual inspection and duplicate removal.

5.0.2 Challenge 2: GPU Memory Constraints

Problem: T4 GPU (16GB) insufficient for standard configurations.
Solution: Used 256×256 images (4× memory reduction), enabled Automatic Mixed Precision (AMP), batch size = 16.

5.0.3 Challenge 3: Small Object Detection

Problem: Objects 10–30 pixels at 256×256 resolution with low contrast.
Solution: Used YOLOv8-Large (not Nano), enabled mosaic augmentation.

5.0.4 Challenge 4: GAN Training Stability

Problem: GANs prone to mode collapse and instability.
Solution: Used Pix2Pix (more stable than unconditional GAN), added L1 loss (lambda = 100) anchoring generator to ground truth.

6 Understanding the Results

The 6.0% mAP@50-95 represents a **successful proof-of-concept**:

- **Task Difficulty:** Training on low-contrast synthetic data, detecting 10–30 pixel objects
- **Domain Gap:** Synthetic night-vision $\neq$ real night-vision (GAN limitation)
- **Learning Validation:** Consistent loss reduction and plausible predictions
- **Achievement:** Viable domain adaptation without expensive real night-vision data

7 Technical Implementation

7.1 Hardware and Software

Component	Specification
GPU	Tesla T4 (16GB VRAM)
Framework	PyTorch 2.8.0+cu126
YOLO Library	Ultralytics 8.3.228
CUDA	12.6
Python	3.12.12

Table 8: Hardware and Software Stack

7.2 YOLOv8 Training

```
from ultralytics import YOLO

model = YOLO('content/best.pt')
```

```
results = model.train(
    data='visdrone_night.yaml',
    epochs=50,
    batch=16,
    imgsz=256,
    device=0,
    project='runs/detect',
    name='finetune_visdrone_nightscaled'
)
```

7.3 Data Pipeline

1. Input: VisDrone daytime images + annotations
2. Pix2Pix: Synthetic night-vision generation
3. Scaling: Coordinate adjustment to 256×256
4. Conversion: YOLO and COCO format generation
5. Training: Fine-tune YOLOv8 and Faster R-CNN
6. Inference: Predictions on validation set

8 Future Work

8.1 Real-World Validation

Test on actual night-vision drone footage to measure true domain transfer performance.

8.2 Advanced Architectures

CycleGAN: Unpaired image-to-image translation, trains directly on real night-vision images.

8.3 Ensemble Methods

Weighted Boxes Fusion (WBF): Combine YOLOv8 and Faster R-CNN predictions to leverage complementary strengths.

8.4 Multi-Resolution Training

Train at 512×512 or 1024×1024 for improved small object detection.

9 Conclusion

This project successfully demonstrates a complete pipeline for night-vision object detection using synthetic data generation and domain adaptation. Key contributions:

- Trained Pix2Pix GAN generating 6,471 synthetic night-vision images
- Developed robust annotation scaling preserving bounding box labels
- Fine-tuned YOLOv8 (mAP@50 = 12.2%) and Faster R-CNN (convergence achieved)
- Proved synthetic data viability for domain adaptation

While 6.0% mAP@50-95 is modest, it represents successful learning on an extremely challenging task with:

- Consistent loss reduction across epochs
- Plausible bounding box predictions
- Real-time inference (11.4 ms/image)
- Strong foundation for future improvements

This framework enables cost-effective development of 24/7 autonomous UAV capabilities without expensive night-vision data collection. Real-world validation on actual night-vision footage remains the critical next step.

10 References

References

[1] VisDrone Dataset. GitHub Repository. `https://github.com/VisDrone/VisDrone-Dataset`

[2] PyTorch CycleGAN and Pix2Pix. GitHub Repository. `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`

[3] ImageNet Night Vision Dataset and augmentation pipeline. Kaggle. `https://www.kaggle.com/datasets/udoysaha103/imagenet-night-vision/discussion?sort=hotness`