Motorcycle Detection System

Technical Documentation and Implementation Report

Abdullah Kamal

Table of Contents

---

1. Executive Summary

This report documents a comprehensive motorcycle detection system that processes hierarchically organized image datasets using deep learning object detection. The system employs Faster R-CNN with a ResNet-50 Feature Pyramid Network (FPN) backbone to identify and annotate motorcycles in images and produces both visual outputs and structured data suitable for analytics and reporting.

Key Capabilities - Automated detection of motorcycles across large-scale image datasets - Preservation of a strict recording → person → image folder hierarchy - Generation of annotated images with bounding boxes and confidence scores - Production of machine-readable CSV reports for downstream analysis - Creation of statistical summaries of detection results for quick assessment

---

2. System Objectives

2.1 Primary Objectives

Automated Object Detection - Identify and localize motorcycles in digital images with high accuracy - Process large volumes of images without manual intervention - Support multiple image formats (JPEG, PNG, BMP, TIFF, WebP)

Data Organization and Traceability - Maintain strict hierarchical organization (recording/person/image) - Preserve metadata relationships throughout the pipeline - Enable granular analysis at recording, person, and image levels

Visual Documentation - Generate annotated copies of images with bounding boxes - Display detection confidence scores - Create publication-ready visualizations for review

Structured Reporting - Produce machine-readable CSV reports - Generate summary statistics on detection outcomes - Enable integration with analytics workflows

2.2 Use Case Scenarios

- Traffic monitoring and surveillance across recording sessions
- Urban planning research on motorcycle density by location and time
- Safety and compliance monitoring in restricted zones
- Transportation studies of motorcycle traffic patterns
- Dataset annotation for machine learning training

---

3. System Architecture

3.1 Technology Stack

Deep Learning Framework - PyTorch (GPU-accelerated when available) - TorchVision (pre-trained detection models) - Model: Faster R-CNN with ResNet-50 FPN - Training Dataset for weights: COCO

Image Processing - Pillow (PIL) for loading, conversion, and annotation - Supported formats: JPEG, PNG, BMP, TIFF, WebP

Data Management - CSV for structured export - Pathlib for cross-platform path handling - TQDM for progress tracking

3.2 Model Specifications

Architecture: Faster R-CNN ResNet-50 FPN - Backbone: ResNet-50 - Feature Extractor: FPN for multi-scale detection - Region Proposal Network: candidate region generation - Detection Head: classification and bounding box regression

Model Characteristics - Pre-trained on COCO (80 categories) - Motorcycle class extracted from COCO labels - Inference optimized with `torch.inference_mode()` - Default confidence threshold: 0.5 (configurable)

Performance Considerations (Model Level) - GPU acceleration utilized when available; CPU fallback supported - Optional image resizing for resource-constrained environments

4. Input Specifications

4.1 Required Directory Structure

The system expects a three-level hierarchy:

```
input_directory/
├── recording_001/
│   ├── person_A/
│   │   ├── image_001.jpg
│   │   ├── image_002.jpg
│   │   └── image_003.png
│   └── person_B/
│       ├── image_001.jpg
│       └── image_002.jpg
├── recording_002/
│   └── person_C/
│       ├── photo_001.jpg
│       └── photo_002.jpg
└── recording_003/
    ├── person_D/
```

```
│   └── ...
└── person_E/
    └── ...
```

Hierarchy Levels - Root: base input folder containing all recordings - Recording Level: sessions/dates/locations - Person Level: subjects or tracked entities per recording - Image Level: individual image files

4.2 Input Requirements and Constraints

Supported Formats: JPG/JPEG, PNG, BMP, TIFF, WebP
File Naming: no restrictions; case-insensitive extensions; Unicode supported
Image Specs: RGB conversion automatic; `--max_size` controls optional downscaling; any aspect ratio; higher resolution improves accuracy

4.3 Data Exclusion Rules

Excluded at Recording Level: files at root, hidden folders, symbolic links
Excluded at Person Level: files (non-directories), hidden folders
Excluded at Image Level: non-images, subdirectories below person level, hidden or corrupted files
Empty Folders: silently skipped; no output rows; counts summarized in console

---

5. Processing Pipeline

5.1 Workflow Overview

1) Input validation
2) Device selection (GPU/CPU)
3) Model loading and initialization
4) Image discovery and metadata collection
5) Iterative processing per image: load → preprocess → inference → detection & annotation → save
6) CSV report generation
7) Statistical summary

5.2 Detailed Processing Steps

Step 1: Input Validation - Verify input directory exists and is accessible - Check structure compliance

Step 2: Device Selection - `torch.device("cuda" if torch.cuda.is_available() else "cpu")` - Auto GPU selection with CPU fallback; device logged

Step 3: Model Initialization - Load Faster R-CNN with pre-trained weights - Extract COCO category labels - Identify motorcycle class ID dynamically (case-insensitive), with fallback index 4 - Set model to evaluation mode

Step 4: Image Discovery - Recursive traversal of the directory structure - Collect `recording_id`, `person_id`, `image_id` - Validate paths and count total images

Step 5: Image Processing Loop - **Load**: open file, convert to RGB, optional resize by `--max_size` - **Preprocess**: model-specific transforms; convert to tensor; move to device - **Inference**: forward pass; obtain boxes, scores, labels - **Filter**: apply confidence threshold; retain motorcycle class only - **Annotate**: draw boxes; add labels with confidence; adaptive styling by image size - **Save**: preserve original subfolders; create directories as needed; save annotated image - **Record**: append detection count and metadata to results

Step 6: Report Generation - Write per-image summary CSV with UTF-8 encoding

Step 7: Statistics - Compute aggregate metrics and print summary

5.3 Performance Optimizations (Implementation-Level)

- Optional downscaling via `--max_size` (aspect ratio preserved; no upscaling)
- Use of `torch.inference_mode()` to reduce overhead
- Single-image inference for simplicity (batching can be added later)

---

6. Output Specifications

6.1 Output Directory Structure

```
output_directory/
├── motorcycle_detections.csv
├── recording_001/
│   ├── person_A/
│   │   ├── image_001.jpg
│   │   ├── image_002.jpg
│   │   └── image_003.png
│   └── person_B/
│       ├── image_001.jpg
│       └── image_002.jpg
├── recording_002/
```

```
│   └── person_C/
│       ├── photo_001.jpg
│       └── photo_002.jpg
└── recording_003/
    └── ...
```

Structure Preservation: output mirrors input hierarchy; filenames unchanged

6.2 CSV Report Format

Filename: `motorcycle_detections.csv` (overridable via `--csv_name`)

Columns - `recording_id` (string): recording folder name - `person_id` (string): person folder name - `image_id` (string): image filename without extension - `num_motorcycles` (int): count of detected motorcycles

Characteristics - UTF-8 encoding; comma delimiter; header row included - Exactly one row per processed image, including zero-detection images

Example

```
recording_id,person_id,image_id,num_motorcycles
recording_001,person_A,image_001,2
recording_001,person_A,image_002,0
recording_001,person_A,image_003,1
```

6.3 CSV Data Logic and Coverage

Inclusion Criteria - Every successfully processed image yields one row - Images with zero detections are included - All recording/person combinations represented

Rationale for Zero-Detection Rows - Complete inventory of processed images - Accurate calculation of detection rates - Auditability of pipeline coverage

6.4 Annotated Images

Bounding Boxes - Color: green; adaptive stroke width (≈2–5 px) - Rectangle per detected motorcycle

Text Labels - Format: `motorcycle {confidence}` (e.g., `motorcycle 0.87`) - Positioned above top-left of the box; solid background for legibility; black text

Image Quality - Original resolution preserved unless `--max_size` is used - Same file format and filename as input

6.5 Console Output

Progress Example

```
Using device: cuda
Found 1250 images across recordings and persons
Processing images: 100%|████████████| 1250/1250 [05:23<00:00, 3.87it/s]
```

Summary Statistics Example

```
=== Detection Summary ===
Total images processed:         1250
Images with motorcycles:        342 (27.36%)
Total motorcycles detected:     487
Average motorcycles per image:  0.390
Avg per image with motorcycles: 1.424
========================
```

Metrics Definitions - **Total images processed**: total CSV rows - **Images with motorcycles**: rows with `num_motorcycles > 0` - **Detection rate**: (images with motorcycles / total) × 100 - **Total motorcycles**: sum of `num_motorcycles` - **Average per image**: total motorcycles / total images - **Average per detected image**: total motorcycles / images with motorcycles

---

7. System Logic and Algorithm

7.1 Detection Algorithm

Phase 1: Feature Extraction (ResNet-50) - Multi-scale features produced via FPN

Phase 2: Region Proposals (RPN) - Anchor boxes at multiple scales/aspect ratios; thousands of proposals per image

Phase 3: Classification and Refinement - Proposal classification; bounding box regression; confidence scoring

Phase 4: Filtering - Non-Maximum Suppression removes overlaps; confidence threshold applied; retain motorcycle class only

7.2 Confidence Scoring

Range: 0.0–1.0
Default Threshold: 0.5

Guidelines - 0.9–1.0: very high confidence - 0.7–0.9: high confidence - 0.5–0.7: moderate confidence - <0.5: low confidence (more false positives expected)

Threshold Selection - Higher threshold reduces false positives but may miss objects - Lower threshold increases recall with more false positives

7.3 Motorcycle Identification Logic

Dynamic Class ID Detection - Case-insensitive string match on COCO labels to find "motorcycle" - Fallback to index 4 when labels vary across weight versions

Multi-Label Handling - Multiple motorcycles per image are supported; counts aggregated per image

7.4 Error Handling and Edge Cases

- Corrupted images: detected on load; processing can skip or terminate depending on configuration
- Invalid folder structure: non-directories are skipped without error
- Empty folders: no CSV rows produced; output dirs may be created empty
- GPU memory overflow: mitigated by `--max_size`; possible CPU fallback
- Invalid predictions: boundary checks prevent index errors; malformed predictions skipped

---

8. Command-Line Interface

8.1 Basic Usage

```
python detect_motorcycles_nested.py \
    --input_dir /path/to/input \
    --output_dir /path/to/output
```

8.2 Complete Parameter Reference

Parameter	Type	Required	Default	Description
`--input_dir`	String	Yes	—	Root folder containing the recording/person/image hierarchy
`--output_dir`	String	Yes	—	Destination folder for outputs (created if missing)
`--conf`	Float	No	0.5	Confidence threshold (0.0–1.0) for motorcycle detection
`--max_size`	Integer	No	1536	Maximum image dimension for inference (0 = no resizing)
`--csv_name`	String	No	`motorcycle_det`	Custom CSV filename

Parameter	Type	Required	Default	Description
			ections.csv	

8.3 Usage Examples

Example 1: Basic Execution

```
python detect_motorcycles_nested.py \
    --input_dir "D:\\data\\recordings\\root" \
    --output_dir "D:\\data\\results"
```

Example 2: High-Confidence Detection

```
python detect_motorcycles_nested.py \
    --input_dir "D:\\data\\recordings\\root" \
    --output_dir "D:\\data\\results" \
    --conf 0.75
```

Example 3: Faster Processing (Lower Resolution)

```
python detect_motorcycles_nested.py \
    --input_dir "D:\\data\\recordings\\root" \
    --output_dir "D:\\data\\results" \
    --max_size 1024 \
    --conf 0.6
```

Example 4: Maximum Quality (No Resizing)

```
python detect_motorcycles_nested.py \
    --input_dir "D:\\data\\recordings\\root" \
    --output_dir "D:\\data\\results" \
    --max_size 0 \
    --conf 0.5
```

Example 5: Custom Report Name

```
python detect_motorcycles_nested.py \
    --input_dir "D:\\data\\recordings\\root" \
    --output_dir "D:\\data\\results" \
    --csv_name "traffic_analysis_2025_10_08.csv"
```

8.4 Windows-Specific Path Handling

Recommended invocation patterns:

```
# Double quotes (recommended)
python detect_motorcycles_nested.py --input_dir "D:\\data\\root"

# Forward slashes
python detect_motorcycles_nested.py --input_dir "D:/data/root"

# Raw strings (PowerShell)
python detect_motorcycles_nested.py --input_dir 'D:\\data\\root'
```

---

9. Limitations and Considerations

9.1 Technical Limitations

- COCO pre-training may not generalize to all motorcycle types
- Reduced performance on very small objects (< 50 px)
- Sensitivity to severe occlusion and unusual viewpoints
- Potential confusion among visually similar classes (e.g., bicycles, scooters)
- Single class detection; no temporal tracking; fixed batch size of 1

9.2 Environmental Factors Affecting Accuracy

- **Lighting:** daylight is optimal; low light/backlighting can degrade results
- **Weather:** rain, fog, or snow may obscure objects
- **Camera Angle:** side or 45-degree views are preferable to top-down
- **Image Quality:** blur, low resolution, or heavy compression can reduce accuracy
- **Occlusion:** partial occlusions are often handled; severe occlusions may lead to misses

---

10. Conclusion

10.1 Summary

The system provides an automated, traceable solution for detecting motorcycles in hierarchically organized image datasets using a state-of-the-art detector (Faster R-CNN ResNet-50 FPN). Outputs include annotated images and a compact CSV per image, enabling straightforward analysis and reporting.

10.2 System Impact

- **Efficiency:** reduces manual review by more than an order of magnitude
- **Scalability:** verified on datasets up to tens of thousands of images with linear scaling
- **Accuracy Tuning:** threshold configuration balances precision and recall to match operational needs

10.3 Recommendations

- Begin with a representative pilot subset to verify results against expectations
- Adjust `--conf` and `--max_size` based on the desired balance of speed and accuracy
- Maintain clear input folder conventions to ensure smooth processing and analysis