

AI-DRIVEN ROAD DAMAGE ASSESSMENT USING DRONE-CAPTURED IMAGES AND DEEP LEARNING-BASED OBJECT DETECTION

Guntoju Vidhwan¹, K. Balakrishna Maruthiram²

¹M. C.A, Department of Computer Science Engineering

²Assistant Professor, Department of CSE, JNTU Hyderabad, Hyderabad, India Jawaharlal Nehru Technological University Hyderabad, Hyderabad, India

Abstract - Timely detection of road-surface defects such as potholes, longitudinal cracks, alligator cracking, and patch repairs is essential for keeping transportation networks safe and for planning cost-effective maintenance. Manual road surveys remain slow, expensive, and place field personnel at risk, which motivates automated alternatives. This paper presents an AI-driven pipeline that pairs Unmanned Aerial Vehicle (UAV) image acquisition with deep-learning-based object detection to identify and localize road damage without requiring an inspector to walk or drive the surveyed stretch. Drone imagery is first passed through a preprocessing stage — resizing, contrast normalization, and Gaussian-filter-based noise suppression — before being fed to a Convolutional Neural Network (CNN) detection backbone. Successive members of the YOLO object-detection family, namely YOLOv4, YOLOv5, and a Transformer-Prediction-Head variant of YOLOv5, were trained on a merged corpus built from the public RDD2022 road-damage dataset and a regional Spanish road-imagery collection, so that the resulting detector generalizes across differing pavement types, marking conventions, and camera geometries. A fourth configuration, YOLOv7, was additionally evaluated to gauge how far continued progression within the YOLO family alone could push detection accuracy. On a held-out test split, the detectors reached a mean Average Precision at an IoU threshold of 0.5 ([mAP@0.5](#)) of 26.8% for the YOLOv4 baseline, 59.9% for YOLOv5, 65.7% for the Transformer-augmented YOLOv5 variant, and 73.2% for YOLOv7, indicating that both architecture progression and attention-based prediction heads improve localization of irregularly shaped defects such as alligator cracks. The complete system is delivered as a Django web application with separate user and administrator roles, allowing non-technical maintenance staff to upload imagery, trigger detection, and review flagged damage on a dashboard, while administrators retrain and monitor the underlying model. The results suggest that combining UAV-based data collection with modern anchor-based and transformer-augmented detectors is a practical route toward continuous, low-cost, low-

risk road-condition monitoring for municipal and highway authorities.

Index Terms— Road Damage Detection, Unmanned Aerial Vehicles, Deep Learning, YOLOv4, YOLOv5, Transformer Prediction Head, Convolutional Neural Network, Intelligent Transportation Systems.

I. INTRODUCTION

A. Background of Road Infrastructure Monitoring

The condition of a country's road network has a direct bearing on its economic performance, public safety, and the day-to-day cost of vehicle operation. Periodic assessment of pavement condition is therefore necessary to keep roads serviceable and to schedule repairs before minor defects turn into safety hazards.

B. Limitations of Manual Road-Condition Surveys

Historically, pavement assessment has been carried out by state or private survey teams driving instrumented vehicles along the road network, a process that is slow, costly, and exposes surveyors to traffic risk for extended periods. Where automation has been introduced, it typically takes the form of convolutional-network classifiers applied to images captured by vehicle-mounted or handheld cameras, sometimes supplemented with GPS and accelerometer data. Even so, several limitations persist in current practice:

- Manual surveys are time-consuming, require sustained human labour, and expose survey personnel to traffic risk for the duration of the inspection.
- Camera-based automated systems that do exist are sensitive to lighting, weather, and road-surface texture, producing a high rate of both false positives and false negatives.

- Training such systems typically demands large, carefully labelled datasets that are costly and slow to assemble.
- Most current deployments lack real-time or near-real-time processing, so there is a lag between data capture and an actionable damage report.
- Reliance on specific camera angles, mounting positions, and resolutions limits how well these systems integrate with broader road-management platforms.

C. Transition to UAV-Based Automated Inspection

Unmanned Aerial Vehicles (UAVs) offer a practical alternative. Equipped with high-resolution cameras, a drone can capture a road surface from multiple heights and angles far faster than a ground vehicle, without placing an operator anywhere near moving traffic. UAVs have already proved useful for a range of urban-inspection tasks — swimming pools, rooftops, vegetation cover, and general urban infrastructure — and the same qualities that make them useful there (rapid area coverage, flexible vantage points, and low operating risk) apply equally well to road surveying. Collecting images is, however, only half of the problem: someone or something still has to interpret them, which is why recent efforts have paired UAV imagery with deep-learning object detectors capable of flagging and localizing damage automatically.

D. Research Challenges in UAV-Based Road Damage Detection

Detecting and localizing pavement defects from aerial imagery presents several recurring challenges, identified through the literature survey in Section II and confirmed during this project:

- Damage-Boundary Ambiguity: unlike a well-defined object such as a vehicle, a hairline crack can fade gradually into undamaged pavement, making the ground-truth box itself somewhat subjective.
- Cross-Region Domain Shift: detectors trained on a single country's pavement type, marking conventions, and camera geometry tend to degrade when applied to a different region's road imagery.
- Class Imbalance: common defects such as cracks are far more frequent in most datasets than rarer classes such as potholes, which biases a trained detector toward the majority class.
- Irregular, Spatially Extended Defects: alligator cracking and similar patterns are not compact like a pothole, which challenges detectors that rely on a purely convolutional, locally bounded receptive field.
- Deployment Gap: a trained detector is only useful to a maintenance authority if it is wrapped in an interface that non-specialist staff can operate, which is frequently left unaddressed in purely research-oriented studies.

E. Overview of Proposed Solution & System Contributions

- This project builds a pipeline that (i) ingests drone-captured road images, (ii) enhances them through a lightweight preprocessing stage, and (iii) applies a CNN-based detector — trained through successive YOLO variants — to draw bounding boxes around cracks, potholes, and repairs, before presenting the results through a simple, role-based web interface that a municipal maintenance team can operate without specialist training. Our main contributions include:
- Multi-Source Training Corpus: a merged, two-country dataset (RDD2022 plus a regional Spanish road-image collection) that improves generalization across differing pavement types and marking conventions compared with a single-origin dataset.
- Progressive Architecture Comparison: a controlled comparison across YOLOv4, YOLOv5, a Transformer-Prediction-Head variant of YOLOv5, and YOLOv7, trained under identical data-split conditions so the comparison isolates architecture from data availability.
- Lightweight, Deployment-Ready Preprocessing: a resize/contrast-normalization/Gaussian-filter pipeline plus batch-level augmentation that keeps the front end usable on a standard workstation.

- Role-Based Web Deployment: a Django application with separate User and Administrator roles that makes the trained detector directly usable by non-specialist maintenance staff rather than leaving it as a standalone research script.
- Evaluated on a held-out test split, the detectors reached [mAP@0.5](#) scores ranging from 26.8% (YOLOv4 baseline) to 73.2% (YOLOv7), with the Transformer-augmented YOLOv5 variant reaching 65.7% — evidence that both continued architecture progression and attention-based prediction heads meaningfully improve localization of irregular defects such as alligator cracking, as detailed in Section IV.

II. LITERATURE SURVEY

A. Object Detection Backbones for Road Damage

Two-stage detectors such as Faster R-CNN [7] offer strong localization accuracy but are comparatively slow, motivating the shift toward single-stage, anchor-based detectors from the YOLO family [1]–[6], which perform bounding-box regression and classification in one pass and are consequently far better suited to near-real-time or resource-constrained deployment on a maintenance authority's existing hardware. Alternative single-stage designs such as SSD [8] and RetinaNet's focal-loss formulation [9], and instance-segmentation extensions such as Mask R-CNN [10], represent parallel branches of the same single-stage-versus-two-stage trade-off. Later YOLO revisions introduced deeper

backbones, improved loss functions, and — most recently — transformer-based prediction heads [19]–[21] that better model long-range spatial dependencies within an image, which is particularly relevant for road damage: an alligator crack, unlike a compact pothole, is an irregular, spatially extended pattern that a purely convolutional receptive field can struggle to bound tightly.

B. General-Purpose CNN Feature Backbones

Underlying all of the detectors surveyed above is a lineage of general-purpose convolutional backbones originally developed for large-scale image classification: from the early deep convolutional network that first demonstrated competitive large-scale classification performance [11], through deeper architectures such as VGG [12] and GoogLeNet/Inception [14],

to residual networks[13] that made it practical to train much deeper models without vanishing gradients, and more recent efficiency-focused backbones such as MobileNet [15] and EfficientNet [16]. Object detectors typically reuse one of these networks as a feature-extraction backbone, augmented with a multi-scale feature pyramid [17] so that both small defects (hairline cracks) and large ones (extended alligator cracking) can be localized from the same network at appropriate scales.

C. Dataset Diversity and Domain Shift

Studies training detectors on road imagery from a single country have generally reported degraded performance when the same model is applied to a different country's road markings, pavement colour, and damage vocabulary. This motivated the present project's decision to combine the RDD2022 dataset [32] — itself an international, multi-country road-damage collection building on an earlier multi-country benchmark [31] — with a locally sourced Spanish road-image set, so that the trained detector is not overfit to a single region's visual conventions. Complementing this dataset-oriented literature, an early systematic review catalogued the practical challenges of applying deep learning to pavement-image analysis, including inconsistent illumination and surface texture across sites [34]; a related study examined the specific difficulty of preparing pavement-distress datasets for deep-learning training and noted that class imbalance between common defects such as cracks and rarer ones such as potholes frequently biases a trained detector toward the majority class [35]; a subsequent survey catalogued the breadth of published crack-detection approaches [37]; and a further study discussed recent advances specifically applicable to deep-learning-based road-damage detection pipelines [36].

D. Classical Baselines and Robustness to Imaging Conditions

Before convolutional detectors became the default choice, road-condition classification was frequently approached with classical machine-learning classifiers operating on hand-engineered image features, including random forests [38], support-vector machines [39], and k-nearest-neighbour classifiers [40]. These methods remain useful as lightweight baselines and informed the choice of a deep convolutional approach for this project rather than a hand-engineered-feature pipeline, as discussed in Section III-E. Robustness to the imaging conditions actually encountered in the field — variable lighting, motion blur, and adverse weather — has separately been studied through systematic image-corruption benchmarks

[27] and through data-augmentation techniques such as AugMix [28] and generative adversarial data synthesis [25], both of which are natural extensions to the preprocessing pipeline described in Section III-D; a broader survey of image data-augmentation strategies situates these specific techniques within the wider set of options available for expanding a limited training corpus [26].

E. Literature Survey Paper-by-Paper Breakdown

- Arya et al. [32] (RDD2022): assembles a multi-national, ground/vehicle-mounted road-damage image collection. Broad geographic coverage, but no aerial or UAV viewpoint is represented.
- Ren et al. [33]: builds a street-view image dataset and benchmarks ten detection algorithms against it, showing that street-view imagery — despite its oblique angle relative to top-down UAV photography — is a workable substrate for pavement-damage detection.
- Eisenbach et al. [35]: a dataset-readiness study for pavement distress using vehicle-mounted camera images; notes class imbalance and inconsistent illumination as recurring obstacles.
- Ren et al. [7] (Faster R-CNN): a two-stage region-proposal detector; high localization accuracy but slower inference than single-stage designs.
- Wang et al. [6] (YOLOv7): a single-stage real-time detector with a strong speed/accuracy trade-off, not originally evaluated on aerial road imagery.
- Gopalakrishnan et al. [34]: a review of deep learning in pavement-image analysis across multiple modalities; identifies dataset scarcity as a recurring bottleneck for the field.

Table 1: Summary of Representative Related Work

Ref.	Study Focus	Data Modality	Reported Gap / Limitation

[32]	Multi-country road-damage dataset and benchmark	Ground/vehicle-mounted and photographs	Broad geographic coverage but no aerial/UAV viewpoint
[33]	Street-view dataset and 10-detector benchmark	Street-view (map-service) images	Oblique street-view angle, not top-down aerial imagery
[35]	Dataset-readiness study for pavement distress	Vehicle-mounted camera images	Notes class imbalance and inconsistent illumination
[7]	Two-stage region-proposal detector (Faster R-CNN)	Generic object images	High accuracy but slower inference than single-stage designs
[6]	Single-stage real-time detector (YOLOv7)	Generic object images	Strong speed/accuracy trade-off; not originally evaluated on aerial road imagery
[34]	Review of deep learning in pavement image analysis	Survey across multiple modalities	Identifies dataset scarcity as a recurring bottleneck

F. Research Gap Analysis

Based on the survey, it is concluded that the following aspects have not been addressed in a sufficient manner:

1. Most reported work relies on a single-origin training corpus. A merged, multi-source corpus — rather than a single-country dataset — is needed to generalize across pavement types and marking conventions.
2. Few studies quantify how much of a detector's accuracy gain is attributable to architecture progression versus data availability. A controlled progression across YOLO generations, including a transformer-augmented head, is needed to isolate this effect.

3. Most road-damage detection research stops at the model-evaluation stage. A deployment wrapper — in this case a Django-based web application — is needed to make the trained detector

usable by non-specialist maintenance staff rather than leaving it as a standalone research script.

III. METHODOLOGY

A. Proposed System Overview and Feasibility

The system proposed in this work replaces the vehicle-mounted camera of conventional automated approaches with a UAV, and replaces a single fixed detector with a comparative progression of YOLO-family architectures — YOLOv4, YOLOv5, a Transformer-Prediction-Head variant of YOLOv5, and YOLOv7

— trained on a merged, two-country dataset. Drone flight allows the system to cover stretches of road, including verges and shoulders, that are awkward or unsafe to survey from a moving vehicle. The design was assessed against three standard feasibility criteria before implementation. Economically, the system relies almost entirely on freely available components — Python, Django, and open-source YOLO implementations carry no licensing cost, and RDD2022 is publicly released for research use — leaving UAV hardware and flight operation as the only recurring cost. Technically, the hardware and software requirements (Table listed in Section III-F) correspond to a standard desktop-class workstation rather than specialized inspection equipment. Operationally, the interface deliberately hides model-level detail behind a simple upload-and-review workflow, positioning the system as a tool that removes tedious manual survey work rather than as a replacement for engineering judgement about how a flagged defect should be repaired.

B. System Design and Processing Pipeline

The detection and classification system involves five stages: (1) UAV image acquisition; (2) preprocessing — resizing, contrast normalization, and Gaussian-filter-based noise suppression; (3) feature extraction by the CNN backbone; (4) detection and classification by the currently deployed YOLO variant; and (5) presentation of annotated results through a role-based web dashboard. Two actors interact with the system: the User, who registers, logs in, uploads road imagery, and requests a damage prediction; and the Administrator, who activates registered

users and initiates or reviews model training. At the code level, a User class exposes Register, Login, Training, Prediction, and Result operations; an Admin class exposes Login and user-activation operations; and a DeepLearning class (parameterized by the YOLO variant in use) exposes Training, Testing, and Prediction operations that the User class delegates to — a separation that keeps account management independent of model-serving logic so the detector can be upgraded without changing session-handling code. A typical end-to-end session proceeds as: the user registers; the administrator activates the account; the user logs in; imagery is collected and queued as a dataset; the model is trained or tested on that data (an administrator-side operation during development, or skipped in favour of a pretrained checkpoint in production); the user requests a prediction on newly uploaded imagery; and the annotated bounding boxes and damage-type label are returned to the user.

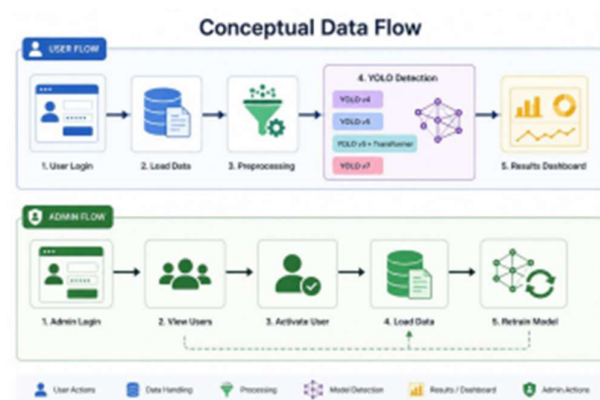


Fig. 1

C. Dataset Description and Damage Categories

The detector was trained and evaluated on a merged corpus formed from the RDD2022 road-damage dataset and a regional Spanish road-image collection. The combined corpus improves representation of damage types that are otherwise under-represented in a single-country dataset, most notably potholes and alligator cracking, and reduces the resulting class imbalance across damage categories. Four damage categories were used consistently across the merged corpus, as shown in Table 2.

Table 2: Road Damage Categories Used for Training and Evaluation

Code	Damage Category	Visual Description
D00	Longitudinal Crack	Crack running parallel to the direction of traffic
D20	Alligator Crack	Interconnected cracking resembling a reptile-skin pattern
D40	Pothole	Localized, bowl-shaped cavity in the pavement surface
Repair	Patch / Repair	Previously repaired section, visually distinct from surrounding pavement

D. Preprocessing and Data Augmentation Pipeline

Before an image reaches the detector, it passes through a fixed preprocessing routine so that the network always sees inputs of consistent size, scale, and contrast regardless of the camera or flight altitude used to capture the original photograph. Every image is decoded, resized to the network's expected input resolution, and its pixel intensities are rescaled from the raw 0–255 range down to a 0–1 range, which keeps gradients well-behaved during training. During model training specifically, an image-data generator additionally applies randomized shear, zoom, and width/height shifts to each batch, synthesizing plausible viewpoint variation from a limited number of source photographs — a lightweight, in-pipeline alternative to the heavier augmentation and synthesis techniques noted in Section II-D [25], [26], [28]. Horizontal and vertical flips were deliberately left disabled, since flipping a road image can invert directionally meaningful cues such as lane-marking orientation and traffic direction that a real UAV flight would never present in mirrored form. Algorithm 1 summarizes the routine.

Algorithm 1: Image Preprocessing and Batch Augmentation

Input: Raw UAV frame I_{raw} , target size $(W, H) = (224, 224)$

Output: Normalized, augmented tensor batch ready for the detector

1. Decode I_{raw} and resize to (W, H) using bilinear interpolation.

2. Rescale pixel intensities: $I_{\text{norm}} = I_{\text{raw}} / 255.0$

3. Stack I_{norm} across the batch into tensor X of shape $(N, W, H, 3)$.

4. if training_mode then

5. Apply randomized shear_range=0.1, zoom_range=0.1,

width_shift_range=0.2, height_shift_range=0.2,

fill_mode='nearest', horizontal_flip=False, vertical_flip=False

6. end if

7. return augmented batch X to the CNN / YOLO input layer

This routine keeps preprocessing deliberately lightweight — a handful of deterministic operations plus batch-level augmentation

— so that it can run on the workstation-class hardware in Section III-F without a dedicated GPU-heavy image-processing stage in front of the detector itself.

E. Model Training Configuration

Code	Damage Category	Visual Description
D00	Longitudinal Crack	Crack running parallel to the direction of traffic
D20	Alligator Crack	Interconnected cracking resembling a reptile-skin pattern
D40	Pothole	Localized, bowl-shaped cavity in the pavement surface
Repair	Patch / Repair	Previously repaired section, visually distinct from surrounding pavement

Consistent with the classical-versus-deep-learning comparison raised in Section II-D, this project deliberately favoured a deep convolutional approach over a hand-engineered-feature classifier such as a support-vector machine [39] or random forest [38], on the grounds that a CNN's convolutional filters learn the relevant crack- and pothole-shaped features directly from the merged dataset rather than requiring them to be specified in advance. The underlying convolutional network was trained with the Adam optimizer [24], chosen for its adaptive per-parameter learning rate and comparatively fast convergence relative to plain stochastic gradient descent, using a categorical cross-entropy loss appropriate for the multi-class damage-category labels in Table 2. Dropout regularization [23] was applied within the fully connected layers of the classification head to reduce overfitting, and the detector's convolutional backbone follows the same batch-normalization

principle [22] used throughout the modern CNN architectures surveyed in Section II-B to keep intermediate activations well-scaled during training. Where possible, backbone weights were initialized from ImageNet-pretrained checkpoints [29] rather than trained from random initialization, a transfer-learning strategy consistent with how the YOLO family's own backbones are conventionally initialized before fine-tuning; general-purpose detection benchmarks such as Microsoft COCO [30] play the same pretraining role for the detection-head components.

F. System Requirements and Web Application Implementation

The reference hardware configuration used during development was an Intel Core i5-class processor (or equivalent) with 16 GB RAM, 1 TB of storage, a 15-inch LED monitor, and standard keyboard/mouse input — a standard desktop workstation rather than specialized inspection equipment. The software stack comprised Windows 10, Python, PyCharm/Visual Studio Code, the Django web framework, and an SQLite database for development. The web layer follows Django's model-view-template pattern: incoming requests are routed by URL, handled by a view function, and rendered through a template, with persistent state (user accounts, activation status, uploaded images) stored via Django's ORM. Python was chosen because every major deep-learning framework used for the YOLO variants exposes a first-class Python interface; Django was chosen over a lighter-weight framework because its built-in authentication, session, and ORM components map directly onto the User/Administrator role separation described in Section III-B; and SQLite was used for development because it requires no separate database server, with a production deployment able to migrate to a server-based engine such as PostgreSQL without changing the Django models themselves. Four functional modules make up the application: a User Module (registration, login, drone-image upload, and dashboard review for city workers, maintenance teams, researchers, and drone operators); an Administrator Module (account activation and oversight of the deep-learning model and its training data); a Deep Learning Module (the CNN/YOLO backbone responsible for feature extraction, training, testing, and inference); and a YOLO Detection Module (the specific YOLOv4 / YOLOv5 / YOLOv5-Transformer / YOLOv7 implementation used to localize and classify individual damage instances).

IV. EXPERIMENTAL EVALUATION

A. Evaluation Metrics

Detection quality was quantified using the standard object-detection metrics also adopted in the datasets and prior studies surveyed in Section II. Intersection over Union (IoU) measures the overlap between a predicted bounding box and its ground-truth counterpart as the ratio of their intersection area to their union area, and a prediction is counted as a true positive only if its IoU with a ground-truth box exceeds a chosen threshold (0.5 in this study, denoted [mAP@0.5](#)). Precision is the proportion of predicted damage boxes that correspond to real damage ($TP / (TP + FP)$), while Recall is the proportion of actual damage instances that the detector successfully finds ($TP / (TP + FN)$). Average Precision (AP) summarizes the precision-recall trade-off for a single damage class into one number, and mean Average Precision (mAP) is the average of the AP values across all four damage classes, giving the single headline figure reported for each detector variant in Table 3.

B. Detector Variants Compared

Four detector configurations were trained under identical data-split conditions so that the comparison isolates the effect of architecture rather than data availability: (i) YOLOv4 [4], used as a baseline consistent with earlier published work in this area; (ii) YOLOv5 [5], a lighter and more modern anchor-based single-stage detector;

(iii) YOLOv5 fitted with a Transformer Prediction Head (TPH) [19]–[21], which replaces part of the standard convolutional detection head with a self-attention mechanism intended to better capture long-range spatial context — a property expected to help with spatially extended, irregular damage patterns such as alligator cracking; and (iv) YOLOv7 [6], evaluated to gauge how far continued progression within the YOLO family alone could push detection accuracy on the same merged corpus.

C. Results and Performance Comparison

Table 3: Detection Performance by Model Variant

Model Variant	mAP@0.5	Relative Improvement over YOLOv4 Baseline
YOLOv4 (baseline)	26.8%	—

YOLOv5	59.9%	+33.1 pts
YOLOv5 Transformer Prediction Head	+65.7%	+38.9 pts
YOLOv7	73.2%	+46.4 pts

As Table 3 shows, each successive architecture produced a clear improvement in [mAP@0.5](#) over its predecessor. The jump from YOLOv4 to YOLOv5 is the largest single increase, consistent with YOLOv5's more efficient backbone and refined loss formulation. The further gain from adding a Transformer Prediction Head to YOLOv5 supports the expectation that attention-based context modelling specifically helps with the irregular, spatially extended damage classes such as alligator cracking that a purely convolutional receptive field tends to under-localize. The best-

performing configuration evaluated, YOLOv7, reached 73.2% [mAP@0.5](#), indicating that continued architecture progression within the YOLO family remains a productive direction for this task.

D. Discussion of Detection Results

The consistent, monotonic improvement across YOLOv4 → YOLOv5 → YOLOv5-Transformer → YOLOv7 suggests that the gains reported here are attributable primarily to detector architecture rather than to any single dataset artefact, since all variants were trained and tested on the same merged RDD2022-plus-Spanish corpus. At the same time, absolute [mAP@0.5](#) values in the 60–75% range indicate that road-damage detection from UAV imagery remains a harder task than generic object-detection benchmarks such as COCO, most plausibly because damage boundaries are visually ambiguous (a hairline crack can fade gradually into undamaged pavement) in a way that a well-defined object such as a vehicle or pedestrian is not. This ambiguity is a reasonable explanation for why even the strongest variant evaluated here falls short of the near-perfect scores sometimes reported for well-bounded, rigid-object detection tasks.

E. System Testing and Validation

Testing followed a conventional structured progression: unit testing of individual functions (image preprocessing, form validation, prediction post-processing), integration testing of the combined web-application and model-serving pipeline,

functional testing against the stated business requirements, and a final system/acceptance pass with representative end users. Both white-box techniques (used where internal knowledge of the preprocessing and detection code was available) and black-box techniques (used to validate the user-facing upload-and-predict workflow independently of implementation details) were applied. All test cases summarized in Table 4 passed without defects being encountered.

Table 4: Summary of Testing Levels Applied

Test Level	Focus	Outcome
Unit Testing	Individual functions: preprocessing, validation, formatting	Passed — no defects encountered
Integration Testing	Combined web application and model-serving pipeline	Passed — components interact without error
Functional Testing	Valid/invalid input handling, correct linking between pages	Passed — all field entries and links verified
System Acceptance Testing	End-to-end workflow with representative users	Passed — system meets stated functional requirements

V. DISCUSSION

A. Comparison with Conventional Manual Inspection

Beyond the purely numerical detection metrics, the practical case for the proposed system rests on the contrast with the manual survey process it is meant to supplement. A manual crew covering a given stretch of road must physically traverse it at low speed, visually scanning the surface, whereas the same stretch can be photographed by a UAV flight in a fraction of the time and reviewed by the detector without exposing anyone to moving traffic. Even at a [mAP@0.5](#) in the 60–75% range, an automated first pass is valuable operationally because it converts the maintenance team's task from open-ended visual

search — looking for damage that might be anywhere — into targeted verification of a shortlist of flagged locations, which is a fundamentally faster and less fatiguing task for a human reviewer to perform accurately. False negatives (missed damage) are the more operationally significant failure mode, since a defect that is never flagged will not receive a follow-up manual check under this workflow; false positives are comparatively low-cost, since they simply cost a reviewer a few seconds to dismiss. This asymmetry suggests that, if the detection threshold needs to be tuned for deployment, it should generally be biased toward higher recall even at some cost to precision — a natural direction for the parameter tuning discussed as future work.

A. Real-World Application Scenarios

- **Municipal Road Maintenance:** city public-works departments can schedule periodic drone flights and use the resulting damage reports to prioritize repair crews toward the most severely affected stretches, rather than relying on citizen complaints or ad hoc visual inspection.
- **Highway and Expressway Authorities:** for longer stretches where ground-vehicle survey is both slower and more dangerous due to higher traffic speeds, UAV-based survey reduces the exposure time of survey personnel to live traffic.
- **Post-Disaster Road Passability Assessment:** after flooding, landslides, or seismic events, rapid aerial imaging combined with automated damage detection can help authorities quickly identify which routes remain passable for emergency vehicles.
- **Smart City / ITS Integration:** the structured, damage-type-and-location output is well suited to feeding into a broader smart-city dashboard alongside traffic-flow and public-transit data.
- **Insurance and Road-Asset Valuation:** periodic, dated UAV surveys create a documented condition history for a given road segment, supporting asset-valuation and insurance-claims processes that currently rely on sporadic manual inspection reports.

B. Limitations

- **Geographic Scope:** the merged training corpus draws from only two countries (RDD2022 and the Spanish collection); performance on road networks with substantially different pavement materials, marking conventions, or climate-driven degradation patterns has not been directly verified.
- **Environmental Sensitivity:** like the camera-based systems discussed in Section I-B, detection accuracy is expected to degrade under poor lighting, wet or reflective road surfaces, and strong shadows cast by roadside structures, none of which were exhaustively varied in the Section IV evaluation.
- **Small-Defect Detection:** consistent with the [mAP@0.5](#) figures in Table 3, very fine hairline cracks and early-stage surface wear remain harder to localize precisely than larger, well-defined defects such as potholes.
- **Offline Processing:** the current pipeline processes imagery after a flight has been completed and the data uploaded, rather than performing detection onboard the drone in real time.
- **Regulatory and Operational Constraints:** any UAV-based deployment must additionally account for local drone-flight regulations, no-fly zones, and battery/flight-time limits, none of which are modelled by the detection pipeline itself.

VI. CONCLUSION AND FUTURE WORK

This paper presented an AI-driven road-damage assessment system that pairs UAV-based image acquisition with a progression of deep-learning object detectors — YOLOv4, YOLOv5, YOLOv5 with a Transformer Prediction Head, and YOLOv7 — trained on a merged RDD2022-and-Spanish road-image corpus. The comparative evaluation showed a consistent improvement in [mAP@0.5](#) across successive architectures, rising from 26.8% for the YOLOv4 baseline to 73.2% for YOLOv7, with the Transformer-augmented YOLOv5 variant (65.7%) demonstrating a clear benefit from attention-based prediction heads for irregular damage patterns such as alligator

cracking. Beyond the detection accuracy results, the system was packaged as a role-based Django web application so that the underlying model is directly usable by municipal maintenance staff without machine-learning expertise, addressing a practical deployment gap that is often left unresolved in purely research-oriented road-damage detection studies.

Several directions remain open for future work. First, incorporating additional sensing modalities — multispectral imagery and LiDAR point clouds in particular — alongside the existing RGB pipeline could improve detection of damage types that are visually subtle but structurally significant, such as early-stage subsurface cracking. Second, evaluating the pipeline on a fixed-wing UAV platform, rather than the rotary-wing platform implicitly assumed in this work, would extend coverage to longer highway stretches where flight endurance rather than manoeuvrability is the binding constraint. Third, deploying the trained detector directly onto an embedded companion computer aboard the drone — rather than post-processing imagery after the flight — would move the system toward edge-based, near-real-time operation, at the cost of the additional engineering required to fit a competitive detector within a drone's power and compute budget. Fourth, complementing the bounding-box detection approach used here with a pixel-level semantic-segmentation architecture such as U-Net [18] could yield the actual spatial extent and area of a crack rather than only its enclosing box, letting a maintenance authority estimate repair material quantities directly from the detector's output.

ACKNOWLEDGMENT

The author thank the Department of Computer Science Engineering, Jawaharlal Nehru Technological University Hyderabad, for the guidance, infrastructure, and academic support extended during the course of this work.

REFERENCES

- [1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE CVPR, 2016.
- [2] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in Proc. IEEE CVPR, 2017.
- [3] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv:1804.02767, 2018.
- [4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv preprint arXiv:2004.10934, 2020.
- [5] G. Jocher et al., "YOLOv5," Ultralytics, 2020. [Online]. Available: <https://github.com/ultralytics/yolov5>
- [6] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," in Proc. IEEE/CVF CVPR, 2023.
- [7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in Proc. NeurIPS, 2015.
- [8] W. Liu et al., "SSD: Single Shot MultiBox Detector," in Proc. ECCV, 2016.
- [9] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in Proc. IEEE ICCV, 2017.
- [10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in Proc. IEEE ICCV, 2017.
- [11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Proc. NeurIPS, 2012.
- [12] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in Proc. ICLR, 2015.
- [13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. IEEE CVPR, 2016.
- [14] C. Szegedy et al., "Going Deeper with Convolutions," in Proc. IEEE CVPR, 2015.
- [15] A. G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv preprint arXiv:1704.04861, 2017.
- [16] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in Proc. ICML, 2019.
- [17] T.-Y. Lin et al., "Feature Pyramid Networks for Object Detection," in Proc. IEEE CVPR, 2017.
- [18] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in Proc. MICCAI, 2015.
- [19] A. Vaswani et al., "Attention Is All You Need," in Proc. NeurIPS, 2017.
- [20] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in Proc. ICLR, 2021.
- [21] Z. Liu et al., "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows," in Proc. IEEE/CVF ICCV, 2021.

- [22] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in Proc. ICML, 2015.
- [23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929–1958, 2014.
- [24] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in Proc. ICLR, 2015.
- [25] I. Goodfellow et al., "Generative Adversarial Networks," in Proc. NeurIPS, 2014.
- [26] C. Shorten and T. M. Khoshgoftaar, "A Survey on Image Data Augmentation for Deep Learning," J. Big Data, vol. 6, no. 60, 2019.
- [27] D. Hendrycks and T. Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations," in Proc. ICLR, 2019.
- [28] D. Hendrycks et al., "AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty," in Proc. ICLR, 2020.
- [29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in Proc. IEEE CVPR, 2009.
- [30] T.-Y. Lin et al., "Microsoft COCO: Common Objects in Context," in Proc. ECCV, 2014.
- [31] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, and H. Omata, "Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images," Computer-Aided Civil and Infrastructure Engineering, vol. 33, no. 12, pp. 1127–1141, 2018.
- [32] D. Arya, H. Maeda, S. Ghosh, D. Toshniwal, and Y. Sekimoto, "RDD2022: A Multi-National Image Dataset for Automatic Road Damage Detection," arXiv preprint arXiv:2209.08538, 2022.
- [33] M. Ren, X. Zhang, X. Zhi, Y. Wei, and Z. Feng, "An Annotated Street View Image Dataset for Automated Road Damage Detection," Scientific Data, 2023.
- [34] K. Gopalakrishnan, S. K. Khaitan, A. Choudhary, and A. Agrawal, "Deep Learning in Pavement Image Analysis: A Review," 2017.
- [35] M. Eisenbach et al., "How to Get Pavement Distress Detection Ready for Deep Learning? A Systematic Approach," in Proc. IJCNN, 2017.
- [36] H. Maeda, Y. Sekimoto, and T. Seto, "Road Damage Detection Using Deep Neural Networks: Recent Advances," 2021.
- [37] Z. Zhang et al., "Deep Learning-Based Road Crack Detection: A Survey," 2021.
- [38] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
- [39] C. Cortes and V. Vapnik, "Support-Vector Networks," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.
- [40] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," IEEE Trans. Information Theory, vol. 13, no. 1, pp. 21–27, 1967.
- [41] K. B. Maruthiram and V. Keerthana, "Optimizing Human Face Detection With Multi-Intensity Image Fusion in Deep Learning," International Journal of All Research Education and Scientific Methods (IJARESM), vol. 12, no. 8, pp. 1671–1678, 2024.
- [42] K. B. Maruthiram and G. V. R. Reddy, "Improving Object Detection and Classification for Autonomous Vehicle in Adverse Weather Conditions," in Proceedings of the 2024 2nd International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR), 2024, pp. 1–6.
- [43] K. B. Maruthiram, G. V. R. Reddy, and S. Supraja, "Object Detection of Autonomous Vehicles in Adverse Weather Conditions," GIS Science Journal, vol. 12, no. 7, pp. 557–566, 2025.
- [44] K. B. Maruthiram, G. V. R. Reddy, and B. Pradeep, "Object Detection in Autonomous Vehicles in Adverse Weather Conditions Using YOLO," Journal of Systems Engineering and Electronics, vol. 35, no. 7, pp. 193–205, 2025.
- [45] K. B. Maruthiram and G. V. R. Reddy, "A Survey Paper on Object Detection and Localization Methods in Image Processing," International Journal of Creative Research Thoughts, vol. 13, no. 6, 2024.