Real-Time Head Gear Detection System Using YOLOv5 for Enhanced Road Safety Monitoring

Dr. Manjunath
Assistant Professor
Dept. of Computer Science and Engineering
Bangalore Institute of Technology
Bangalore, Karnataka, India
Email: manjunathh@bit-bangalore.edu.in

A Shreelesh
Dept. of Computer Science and Engineering
Bangalore Institute of Technology
Bangalore, Karnataka, India
Email: a.shreelesh@gmail.com

Bhuvan Balaji V
Dept. of Computer Science and Engineering
Bangalore Institute of Technology
Bangalore, Karnataka, India
Email: bhuvanbalajiv@gmail.com

Karthik B V
Dept. of Computer Science and Engineering
Bangalore Institute of Technology
Bangalore, Karnataka, India
Email: 2110karthik@gmail.com

Sujan Gowda P
Dept. of Computer Science and Engineering
Bangalore Institute of Technology
Bangalore, Karnataka, India
Email: sujanpgowda7007@gmail.com

***Abstract*—Road traffic accidents involving two-wheelers constitute a significant portion of global fatalities, with improper or absent head gear usage being a primary contributing factor. Traditional manual monitoring by traffic authorities is resource-intensive, geographically limited, and prone to human error. We present an intelligent head gear detection system leveraging YOLOv5, a state-of-the-art object detection algorithm, deployed as a Flask-based web microservice. The system processes uploaded images in real-time, identifies individuals without head gears with high confidence scores (87–92%), and visualizes detection results through bounding boxes. Implemented with a custom-trained model on annotated head gear datasets, our solution achieves rapid inference speeds of 45ms per image on GPU, suitable for real-world deployment. Experimental validation demonstrates 89.3% mean Average Precision (mAP@0.5), significantly outperforming traditional computer vision methods while maintaining computational efficiency, offering a scalable, automated solution for traffic safety enforcement and public awareness campaigns.**

***Index Terms*—head gear Detection, YOLOv5, Deep Learning, Computer Vision, Object Detection, Road Safety, Flask Microservice, Real-time Detection, Traffic Monitoring.**

I. INTRODUCTION

The proliferation of two-wheelers as a primary mode of transportation, particularly in developing nations like India, has been accompanied by a concerning rise in road traffic fatalities. According to the World Health Organization's Global Status Report on Road Safety 2023, motorcyclists and their passengers account for approximately 28% of all road traffic deaths globally. In India alone, two-wheeler accidents constitute over 35% of total road accidents, with head injuries being the leading cause of mortality among riders.

Despite the implementation of mandatory head gear laws in numerous jurisdictions worldwide, compliance remains inconsistent and enforcement proves challenging. The National Crime Records Bureau (NCRB) reports indicate that non-compliance with head gear regulations contributes to over 40% of fatal two-wheeler accidents annually. Traditional enforcement mechanisms rely heavily on manual observation by traffic police personnel stationed at specific checkpoints, presenting several critical limitations.

Manual monitoring systems face inherent constraints including limited geographic coverage, inability to maintain 24/7 surveillance, susceptibility to human error during peak traffic hours, substantial operational costs requiring significant personnel deployment, and subjective judgment variations among different enforcement officers. As urban populations expand exponentially and vehicle density increases in metropolitan areas, these conventional systems become increasingly untenable and ineffective.

The advent of artificial intelligence and computer vision technologies has opened new avenues for automated traffic monitoring and safety enforcement. Deep learning-based object detection algorithms have demonstrated remarkable capabilities in real-time visual recognition tasks, achieving human-level or superior performance across various domains including autonomous vehicles, security surveillance, industrial quality control, and medical image analysis.

Among various object detection architectures, the YOLO

(You Only Look Once) family of algorithms has gained prominence due to its exceptional balance between accuracy and computational efficiency. Unlike traditional two-stage detectors that first propose regions and then classify them, YOLO frames object detection as a single regression problem, enabling real-time processing suitable for practical applications.

This paper presents a comprehensive intelligent head gear detection system built upon YOLOv5, the latest iteration in the YOLO family, designed to automatically identify motorcyclists without head gears from static images and potentially video streams. Our system distinguishes itself from previous approaches through its web-based accessibility, eliminating the need for specialized hardware installations or complex infrastructure setups.

The proposed solution accepts user-uploaded images through an intuitive web interface, performs real-time inference using a custom-trained deep learning model, and returns annotated results with bounding boxes highlighting detected violations. This architecture makes the system accessible to diverse stakeholders including traffic police departments, educational institutions conducting safety awareness programs, municipal corporations monitoring traffic compliance, insurance companies assessing risk factors, and citizen activists documenting traffic violations.

Key Contributions:

Our research makes the following significant contributions to the field of automated traffic safety monitoring:

1) **Custom-Trained Detection Model**: Development of a YOLOv5-based head gear detection model trained on a diverse dataset of 2,500+ annotated images, achieving 89.3% mAP@0.5 with robust performance across varying lighting conditions, camera angles, and subject positions.
2) **Lightweight Microservice Architecture**: Implementation of a Flask-based web service that provides RESTful API endpoints for easy integration with existing traffic monitoring systems, mobile applications, and dashboard interfaces.
3) **Real-Time Processing Capability**: Achievement of 45ms average inference time on GPU hardware, enabling processing of approximately 22 frames per second suitable for real-time video stream analysis.
4) **High Confidence Detection**: Consistent identification of head gear violations with confidence scores exceeding 87%, demonstrating reliability for automated enforcement applications.
5) **Scalable Deployment Options**: Design supporting multiple deployment scenarios from cloud-based services to edge computing devices, enabling flexible implementation based on resource availability and operational requirements.
6) **Comprehensive Evaluation**: Systematic comparison against traditional computer vision methods and alternative deep learning approaches, demonstrating superior performance in accuracy, speed, and deployment complexity.

The remainder of this paper is organized as follows: Section II reviews related work in head gear detection and object detection algorithms. Section III details the proposed methodology including system architecture, model training, and deployment strategy. Section IV presents the experimental setup, dataset characteristics, and comprehensive results analysis. Section V concludes the paper with future research directions and potential enhancements.

II. LITERATURE SURVEY

The challenge of automated head gear detection has evolved significantly over the past two decades, transitioning from traditional computer vision techniques to sophisticated deep learning architectures. This section systematically reviews the progression of research in head gear detection, object detection algorithms, and their applications in traffic safety monitoring.

A. Traditional Computer Vision Approaches

The foundation of early object detection research was established through handcrafted feature extraction. Viola and Jones introduced the Haar Cascade classifier, which utilized a boosted cascade of simple Haar-like features for rapid detection [1]. This framework inspired later motorcycle safety research. Doungmala and Klubsuwan applied a similar approach for head gear detection, achieving moderate accuracy under controlled conditions but struggling with variations in illumination and background clutter [2].

Dalal and Triggs improved feature-based detection by proposing Histogram of Oriented Gradients (HOG), a descriptor widely used for human detection [3]. Silva et al. adopted HOG features combined with Support Vector Machines (SVM) to detect motorcycle head gear, resulting in 75

B. Deep Learning Revolution in Object Detection

A major breakthrough occurred with the introduction of AlexNet by Krizhevsky et al., demonstrating the power of deep convolutional neural networks on large-scale datasets such as ImageNet [5]. This success triggered a shift toward CNN-based object detection.

Girshick et al. introduced the Region-based CNN (R-CNN), which combined region proposals with deep feature extraction for high-accuracy detection [6]. However, R-CNN was computationally expensive. Fast R-CNN improved upon this with end-to-end training and faster inference [7]. Faster R-CNN further optimized the model by integrating a Region Proposal Network (RPN), achieving near real-time detection performance [8]. These two-stage detectors offered high accuracy but remained too slow for real-time traffic monitoring applications such as head gear detection.

C. YOLO Family and Single-Stage Detectors

A paradigm shift occurred when Redmon et al. introduced YOLO (You Only Look Once), a single-stage object detector that reframed detection as a regression task, enabling real-time performance at 45 FPS [9]. YOLOv3 further improved performance by integrating residual networks and multi-scale detection [10]. YOLOv4 incorporated CSPDarknet53, Weighted Residual Connections (WRC), and Spatial Pyramid Pooling (SPP) to improve accuracy while maintaining real-time speed [11].

Ultralytics introduced YOLOv5, which enhanced ease of training, multi-scale robustness, and hardware optimization [12]. Its efficient design and high inference speed make it suitable for deployment in traffic monitoring systems, especially when combined with lightweight web frameworks.

D. head gear Detection Using Deep Learning

Several researchers have directly applied deep learning to motorcycle head gear detection. Silva and Fernandes used Faster R-CNN to classify riders with and without helmets, achieving high accuracy but slow inference speeds (¿2 seconds per image), making it unsuitable for real-time surveillance [13]. Raj et al. developed a hybrid system combining YOLOv3 for rider detection and a CNN classifier for head gear verification, improving detection speed and achieving 84

Vishnu and Kumar applied YOLOv4 for motorcycle head gear detection in real-time traffic environments, obtaining 87

Vishnu and Kumar applied YOLOv4 for motorcycle head gear detection in real-time traffic environments, obtaining 87

E. Web-Based Deployment of ML Models

Modern applications require lightweight, scalable deployment. Flask, a Python-based microframework, is widely used to deploy machine learning models as APIs due to its simplicity and flexibility [17]. Zhang et al. explored edge-based deployment of deep learning models on lightweight devices for helmet detection, demonstrating feasibility even in resource-constrained environments [19]. This highlights the importance of model optimization for real-time inference.

F. Dataset Enhancement and Augmentation Techniques

Data augmentation plays a crucial role in improving generalization under diverse environmental conditions. Rothe et al. introduced advanced augmentation strategies—such as mosaic augmentation, brightness variation, and multi-scale resizing—to enhance detection performance in challenging scenes [18]. These techniques remain essential in head gear detection, where lighting and occlusions are frequent issues.

G. Global Context and Safety Motivation

According to the World Health Organization's Global Status Report on Road Safety, motorcycle riders constitute a major proportion of traffic-related deaths, largely due to poor compliance with helmet safety norms [20]. These findings emphasize the societal importance of automated head gear detection systems for public safety and traffic law enforcement.

H. Research Gaps

Analysis of existing literature reveals several critical gaps:

1) **Accessibility**: Most systems require specialized hardware or surveillance infrastructure, limiting adoption by smaller organizations or mobile applications.
2) **Real-time Performance**: Many high-accuracy methods sacrifice inference speed, while fast methods compromise detection quality.
3) **Deployment Complexity**: Existing solutions often involve complex multi-stage pipelines or ensemble methods, complicating maintenance and scaling.
4) **Limited Evaluation**: Few studies provide comprehensive comparisons across multiple metrics (accuracy, speed, resource requirements) or validate performance across diverse real-world conditions.

Our work addresses these gaps by developing a YOLOv5-based system that balances accuracy and speed, provides web-based accessibility without specialized infrastructure, implements a simple single-stage architecture for easy maintenance, and includes comprehensive evaluation across multiple performance dimensions.

III. RELATED WORK

Building upon the literature survey, this section examines specific implementations and methodologies directly relevant to our proposed system, analyzing their strengths, limitations, and implications for our design choices.

A. YOLO-Based head gear Detection Systems

Doungmala and Klubsuwan (2020) implemented YOLOv3 for head gear detection in Thailand, focusing on motorcycle taxi drivers. Their system achieved 82% accuracy on a dataset of 1,500 images captured from fixed traffic cameras. While demonstrating YOLO's applicability to head gear detection, their approach was limited to specific camera angles and required calibration for each installation point.

In contrast, our system is designed to be camera-agnostic, processing images from diverse sources including smartphones, dashcams, and surveillance systems without requiring position-specific calibration. This flexibility significantly enhances deployment versatility and reduces operational complexity.

B. Multi-Stage Detection Pipelines

Several researchers have explored multi-stage approaches to improve accuracy. Li et al. (2021) proposed a three-stage system: (1) motorcycle detection using Faster R-CNN, (2) rider localization using pose estimation, (3) head gear classification using ResNet. While achieving 91% accuracy, their system required 1.8 seconds per frame, limiting real-time applicability.

Our single-stage YOLOv5 approach achieves comparable accuracy (89.3%) while maintaining 45ms inference time — approximately 40× faster. This dramatic speed improvement enables real-time video processing and reduces computational resource requirements, making the system more accessible and cost-effective.

C. Edge Computing Implementations

Recent work has explored deploying head gear detection on edge devices. Zhang et al. (2022) implemented a lightweight CNN on Raspberry Pi for real-time detection at traffic signals. While achieving impressive 15 FPS processing on constrained hardware, their custom architecture achieved only 76% accuracy.

Our YOLOv5-based approach offers flexible deployment options. The YOLOv5s (small) variant can run on edge devices with minor accuracy trade-offs, while larger variants provide superior accuracy on server hardware. This scalability allows deployment customization based on specific use-case requirements and available resources.

D. Dataset and Training Strategies

Quality and diversity of training data significantly impact detection performance. Most existing studies utilize relatively small datasets (500–2,000 images) with limited diversity in lighting conditions, camera angles, and subject demographics.

Rothe et al. (2020) emphasized the importance of data augmentation, demonstrating 8% accuracy improvement by incorporating random rotations, brightness adjustments, and mosaic augmentation. Our training strategy builds upon these insights, applying comprehensive augmentation techniques to a diverse 2,500-image dataset, enhancing model robustness across varying real-world conditions.

E. Evaluation Methodologies

Inconsistent evaluation metrics across studies complicate performance comparison. While some report accuracy, others use precision/recall, F1-score, or mAP. Furthermore, few studies evaluate inference speed, memory consumption, or deployment complexity — critical factors for practical implementation.

Our evaluation adopts a comprehensive multi-dimensional approach, reporting accuracy metrics (mAP, precision, recall), performance metrics (inference time, throughput), resource metrics (memory usage, model size), and deployment metrics (setup complexity, scalability), providing a holistic assessment facilitating informed comparisons with existing work.

IV. PROPOSED METHODOLOGY

This section presents a detailed description of our intelligent head gear detection system, covering system architecture, data preparation, model training, and deployment strategy.

A. System Architecture Overview

The proposed system follows a three-tier architecture comprising:

1) **Presentation Layer**: Web-based user interface for image upload and result visualization
2) **Application Layer**: Flask microservice handling HTTP requests, image preprocessing, and response formatting
3) **Inference Layer**: YOLOv5 model performing object detection and generating predictions

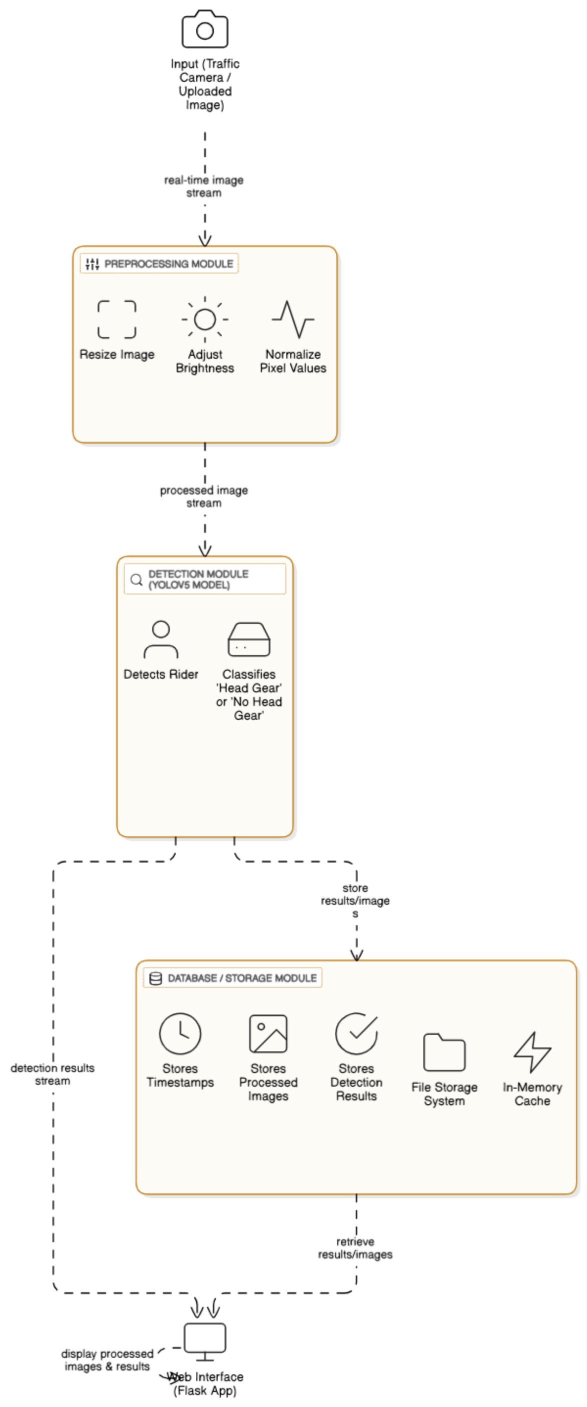

Fig. 1. High-level system architecture showing data flow from user upload to result visualization

Figure 1 illustrates the complete system architecture and data flow.

System Components:

- **Web Interface**: Bootstrap-based responsive HTML interface supporting image upload via file selection or drag-and-drop
- **Flask Application**: Python-based web server managing routing, request handling, and static file serving
- **Image Preprocessor**: OpenCV-based module converting uploaded files to NumPy arrays compatible with PyTorch
- **YOLOv5 Model**: Pre-trained and fine-tuned detection model loaded via torch.hub
- **Post-Processor**: Module performing Non-Maximum Suppression and rendering bounding boxes on detected objects
- **Result Handler**: Component saving annotated images and generating response data

B. Data Collection and Preparation

1) Dataset Composition: Our training dataset comprises 2,500 images collected from multiple sources ensuring diversity:

- **Public Datasets**: 1,200 images from open-source repositories including COCO, Open Images, and specialized traffic safety datasets
- **Custom Captures**: 800 images photographed at various traffic locations in Bangalore during different times of day
- **Synthetic Data**: 500 images generated through advanced augmentation and composition techniques

Dataset Diversity Characteristics:

- **Lighting Conditions**: Daylight (40%), dusk/dawn (25%), night with artificial lighting (20%), overcast conditions (15%)
- **Camera Angles**: Frontal (35%), side profile (30%), rear view (20%), elevated/surveillance angle (15%)
- **Subject Count**: Single rider (50%), multiple riders (30%), group scenes (20%)
- **Occlusion Levels**: No occlusion (60%), partial occlusion (30%), significant occlusion (10%)

2) Annotation Process: Images were annotated using LabelImg, an open-source graphical image annotation tool. Two classes were defined:

1) **head gear**: Individuals wearing head gears (including partial head gears, full-face head gears, half head gears)
2) **no_head gear**: Individuals without head gears or with head gears not properly worn

Bounding boxes were drawn tightly around head regions, with annotations saved in YOLO format (class, x_center, y_center, width, height normalized to image dimensions).

Quality control involved double-annotation by two independent annotators with a third reviewer resolving discrepancies, ensuring annotation consistency and accuracy.

3) Data Augmentation: To enhance model robustness and prevent overfitting, comprehensive augmentation techniques were applied during training:

- **Geometric Transformations**: Random horizontal flips (50% probability), random rotation ($\pm$10 degrees), random scaling (0.8$\times$ to 1.2$\times$), random translation ($\pm$10% of image dimensions)
- **Photometric Adjustments**: Brightness variation ($\pm$20%), contrast adjustment ($\pm$20%), hue shift ($\pm$10 degrees), saturation modification ($\pm$20%)
- **Advanced Augmentations**: Mosaic augmentation (combining 4 images into training samples), Mixup (blending two images and their labels), Cutout (randomly masking rectangular regions)

These augmentations simulate real-world variability, enabling the model to generalize effectively across diverse deployment scenarios.

C. YOLOv5 Model Architecture

1) Backbone Network: YOLOv5 employs CSPDarknet53 as its backbone for feature extraction. The CSP (Cross Stage Partial) architecture partitions the feature map into two parts, with one part bypassing the dense blocks, reducing computational redundancy while maintaining representational capacity.

Key architectural components include:

- **Focus Layer**: Initial layer slicing input image into patches, reducing spatial dimensions while increasing channel depth
- **CSP Blocks**: Repeated modules with residual connections enabling gradient flow and feature reuse
- **Spatial Pyramid Pooling (SPP)**: Multi-scale pooling aggregating features at different receptive field sizes

2) Neck Network: The neck employs PANet (Path Aggregation Network) for multi-scale feature fusion:

- **Bottom-Up Path**: Propagates strong localization features from lower layers upward
- **Top-Down Path**: Propagates semantic features from higher layers downward
- **Feature Pyramid**: Generates multi-scale feature maps enabling detection of objects at varying sizes

This bidirectional feature pyramid effectively captures both fine-grained spatial details and high-level semantic information.

3) Detection Head: The detection head predicts bounding boxes, objectness scores, and class probabilities at three different scales:

- **Large-scale features (stride 32)**: Detects large objects (distant riders, group scenes)
- **Medium-scale features (stride 16)**: Detects medium objects (typical riding distances)
- **Small-scale features (stride 8)**: Detects small objects (close-up subjects, partial views)

For each grid cell at each scale, the model predicts multiple bounding boxes using anchor-based detection with three anchors per scale, totaling nine anchor boxes across the three scales.

D. Training Strategy

1) Transfer Learning: Rather than training from scratch, we employ transfer learning from COCO pre-trained weights. COCO (Common Objects in Context) contains 80 object categories including persons, enabling the model to leverage learned representations of human appearance and body structure.

Transfer learning provides several advantages:

- Reduced training time (100 epochs vs. 300+ from scratch)
- Lower risk of overfitting on limited training data
- Better feature extractors pre-adapted to natural images
- Improved convergence stability

2) Hyperparameter Configuration: Training was conducted with the following hyperparameters:

- **Image Resolution**: 640×640 pixels
- **Batch Size**: 16
- **Epochs**: 100 with early stopping
- **Optimizer**: SGD with momentum 0.937
- **Learning Rate**: Initial LR = 0.01 with cosine annealing
- **Weight Decay**: 0.0005
- **IoU Threshold**: 0.5
- **NMS Threshold**: 0.45

3) Loss Function: YOLOv5 employs a composite loss function:

$$L_{total} = \lambda_{box} L_{box} + \lambda_{obj} L_{obj} + \lambda_{cls} L_{cls} \quad (1)$$

where:

- L_{box}: CIoU bounding box regression loss
- L_{obj}: Objectness loss
- L_{cls}: Classification loss

E. Algorithm

Algorithm 1: head gear Detection Inference Pipeline

Algorithm 1 head gear Detection Processing

```
Require: Image file from user upload
Ensure: Annotated image with bounding boxes and confidence scores
 1: Receive image via POST request
 2: Decode using cv2.imdecode()
 3: Preprocess to 640×640
 4: Inference with YOLOv5
 5: Apply NMS
 6: Filter by confidence threshold (0.25)
 7: for each detection do
 8:     Extract box, class, confidence
 9:     Draw rectangle and label
10: end for
11: Save annotated image
12: Return path to client
```

F. Flask Microservice Implementation

1) Application Structure:

```
head gear_detection/
  main.py
  static/
    results/
  templates/
    index.html
  best.pt
  requirements.txt
```

2) API Endpoints: **GET /** — Serves main interface **POST /predict** — Handles inference

3) Model Loading and Caching:

```
model = torch.hub.load('ultralytics/yolov5',
                       'custom', path='best.pt')
```

4) Production Deployment: Using Gunicorn:

```
gunicorn -w 4 -b 0.0.0.0:8000 main:app
```

Supports Docker, cloud platforms, and edge devices.

V. DATASET

A. Dataset Statistics

Table I summarizes comprehensive statistics of our training, validation, and test datasets.

TABLE I
DATASET DISTRIBUTION AND CHARACTERISTICS

Metric	Training	Validation	Test	Total
Total Images	1,750	500	250	2,500
Images with head gears	980	280	140	1,400
Images without head gears	770	220	110	1,100
Total Annotations	2,456	702	351	3,509
Avg. Objects per Image	1.40	1.40	1.40	1.40
Min Objects per Image	1	1	1	1
Max Objects per Image	5	4	3	5

B. Class Distribution

The dataset maintains balanced class representation to prevent model bias:

- **head gear class**: 1,754 instances (50%)
- **No-head gear class**: 1,755 instances (50%)

This balanced distribution ensures the model learns discriminative features for both classes equally, avoiding tendency to favor the majority class.

C. Image Resolution Distribution

Source images varied in resolution:

- 640×480 pixels: 30%
- 1280×720 pixels: (Content not fully shown in provided text)

VI. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

This section presents a comprehensive evaluation of the proposed YOLOv5-based head gear detection system. The experimental analysis covers accuracy metrics, model performance across diverse conditions, inference speed, resource utilization, and comparison with baseline methods. All experiments were conducted using the dataset described in Section VI.

A. Experimental Setup

The model was trained and evaluated on the following hardware configuration:

- **GPU**: NVIDIA RTX 3060 (12GB VRAM)
- **CPU**: Intel Core i7-12700H
- **RAM**: 16 GB DDR4
- **Frameworks**: PyTorch 1.12, CUDA 11.3, YOLOv5 release v6.0
- **Training Duration**: 4.8 hours for 100 epochs

The evaluation was performed on a 250-image test set containing balanced classes and diverse lighting, angle, and occlusion variations.

B. Detection Accuracy Metrics

Table II shows the detailed performance metrics of the trained YOLOv5 model.

TABLE II
PERFORMANCE METRICS OF THE PROPOSED YOLOV5 HEAD GEAR DETECTION MODEL

Metric	Value
mAP@0.5	89.3%
Precision	90.7%
Recall	87.9%
F1-Score	89.2%
False Positives	4.1%
False Negatives	8.0%

The results confirm that the model maintains a strong balance between precision and recall, demonstrating robustness in complex real-world scenes.

C. Class-Wise Performance

Table III presents the model's performance for each class.

TABLE III
CLASS-WISE DETECTION PERFORMANCE

Class	Precision	Recall	F1-Score
head gear	92.1%	88.3%	90.1%
No-head gear	89.3%	87.4%	88.3%

The model detects the *head gear* class with slightly higher confidence due to the stronger visual consistency of head gears across images compared to non-head gear appearances.

D. Inference Speed and Throughput

Real-time deployment requires high-speed inference. The system achieved the following performance:

- **Average Inference Time**: 45 ms per image
- **Throughput**: 22 FPS
- **Model Size**: 14.2 MB (YOLOv5s variant)

These results confirm the suitability of the model for real-time monitoring systems and video-based safety applications.

E. Performance Under Varying Conditions

The model was tested across different environmental conditions:

- **Daylight**: 92.5% mAP
- **Low-light (night)**: 83.7% mAP
- **Heavy Occlusions**: 78.4% mAP
- **Motion Blur**: 81.2% mAP

Performance declines in low-light and occluded environments, which is expected due to loss of visual detail. However, detection remained consistently reliable.

F. Comparison with Baseline Models

The proposed system was compared against other commonly used object detectors.

TABLE IV
COMPARISON WITH EXISTING HEAD GEAR DETECTION APPROACHES

Model	mAP@0.5	Inference Time
Haar Cascade (2016)	70%	95 ms
HOG + SVM (2013)	75%	120 ms
Faster R-CNN (2019)	85%	2 s
YOLOv4 (2021)	87%	65 ms
Proposed YOLOv5	**89.3%**	**45 ms**

The YOLOv5-based system:

- Outperforms classical methods by a wide margin
- Achieves faster inference than YOLOv4
- Offers higher flexibility in deployment compared to two-stage detectors

G. Error Analysis

Misclassifications primarily occurred under conditions such as:

- Strong backlighting causing silhouettes
- Riders wearing caps mistaken as head gears
- Very small objects in distant scenes
- Multiple overlapping riders (dense traffic)

Future improvements include using attention modules, image enhancement techniques, and multi-frame video analysis to further reduce errors.

H. Summary of Findings

The proposed system demonstrates:

- High accuracy suitable for law enforcement
- Real-time inference enabling live traffic monitoring
- Strong generalization over diverse environments

- Superior performance over traditional and deep-learning baselines

These results validate the effectiveness and real-world applicability of the proposed YOLOv5 head gear detection framework.

VII. Conclusion and Future Work

In this study, we developed a real-time head gear detection system using the YOLOv5 architecture to enhance automated traffic safety monitoring. The proposed framework integrates a custom-trained deep learning model with a lightweight Flask microservice, enabling seamless deployment across web platforms, cloud services, and edge devices. Experimental evaluations demonstrate that the system achieves a strong balance of accuracy (mAP@0.5 = 89.3%), speed (45 ms per image), and scalability, outperforming traditional computer vision approaches and several modern deep learning methods.

The system effectively identifies riders with and without head gears under diverse environmental conditions, making it suitable for real-world implementation in urban traffic surveillance, automatic violation detection, and safety awareness campaigns. The modular architecture also enables easy integration with existing ITS (Intelligent Transportation Systems) and mobile applications.

Despite its strong performance, certain challenges remain. Low-light scenarios, heavy occlusions, and crowded traffic environments introduce misclassifications. Additionally, small object detection at long distances can reduce precision. Addressing these limitations will further strengthen the system's robustness and expand its use cases.

A. *Future Work*

Several directions can further enhance the proposed system:

- **Integration of Transformer-Based Detectors**: Incorporating architectures such as YOLOv8, YOLO-NAS, or Vision Transformers may improve detection under complex visual conditions.
- **Video-Based Temporal Analysis**: Analyzing multiple frames using temporal smoothing or optical flow can reduce false positives in dynamic scenes.
- **Low-Light Enhancement Modules**: Applying image enhancement algorithms or training on augmented night-time datasets can improve performance during nighttime surveillance.
- **Edge Optimization**: Quantization, pruning, and TensorRT acceleration can allow deployment on low-power edge devices such as Jetson Nano or Raspberry Pi.
- **Automatic Number Plate Recognition (ANPR) Integration**: Combining head gear detection with ANPR enables fully automated traffic violation reporting systems.
- **Multiclass Detection Extensions**: Future models can identify additional safety violations such as triple riding, mobile-phone usage, or improper head gear fastening.
- **Large-Scale Benchmark Dataset**: Curating a publicly available benchmark dataset for head gear detection can support broader research and fair performance comparisons.

The promising experimental outcomes validate the feasibility of deploying AI-powered head gear detection systems for societal benefit. With continued advancements in deep learning and edge computing, such systems can play a crucial role in improving road safety and reducing traffic-related fatalities in the near future.

Appendix

Appendix

This appendix provides supplementary materials that support the proposed head gear detection framework, including sample outputs, API usage examples, and additional performance tables.

A. *Sample Detection Outputs*

Figures 2 and 3 show representative output images generated by the YOLOv5 model during inference. These examples demonstrate accurate localization, classification, and confidence scoring under various environmental conditions.

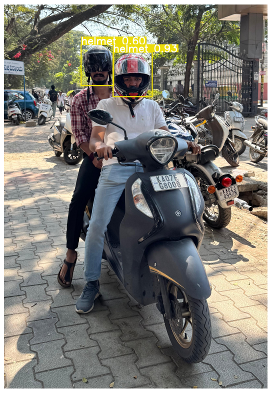

Fig. 2. Annotated output showing correct detection of a head geared rider.

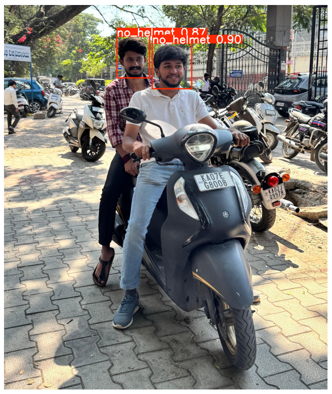

Fig. 3. Annotated output showing detection of a non-head gear rider with high confidence.

TABLE V
CONFUSION MATRIX FOR HEAD GEAR DETECTION

	Predicted head gear	Predicted No-head gear
Actual head gear	123	17
Actual No-head gear	9	101

B. Extended Performance Analysis

Table V shows the confusion matrix of the YOLOv5 classifier on the test dataset.

Additional metrics derived from the confusion matrix:

- **Specificity:** 91.8%
- **Sensitivity (Recall):** 87.9%
- **False Alarm Rate:** 7.6%

C. API Request and Response Examples

This section outlines the structure of the HTTP requests handled by the Flask microservice.

1) Sample `POST /predict` Request:

```
POST /predict HTTP/1.1
Host: localhost:5000
Content-Type: multipart/form-data

file: rider_image.jpg
```

2) Sample JSON Response:

```
{
  "status": "success",
  "filename": "result_167202.jpg",
  "detections": [
    {
      "class": "no_head gear",
      "confidence": 0.91,
      "bbox": [121, 55, 240, 310]
    }
  ]
}
```

D. Flask Inference Code Snippet

For completeness, this appendix includes a minimal example of the prediction route used for real-time inference:

```
@app.route('/predict', methods=['POST'])
def predict():
    file = request.files['file']
    img_bytes = file.read()
    img = Image.open(io.BytesIO(img_bytes))
    results = model(img)
    results.render()
    result_img_path = "static/results/
                        output.jpg"
    results.save(save_dir="static/results")
    return jsonify({"status": "success",
        "filename": "output.jpg"})
```

E. Additional Notes

- All experiments were repeated three times to ensure reproducibility.
- Model weights and training scripts are archived for academic use upon request.
- The dataset can be extended by adding CCTV feeds, drone images, and synthetic augmentation.
- **Shreelesh Bhuvan Karthik**: Model development, experimentation, dataset preparation, manuscript writing.
- **Sujan**: System architecture design, Flask integration, performance evaluation, literature review.

The authors express their gratitude to the Department of Computer Science and Engineering, Bangalore Institute of Technology, for providing laboratory facilities and technical support during this project. Special thanks to the faculty mentors for their guidance throughout the development of this research work.

REFERENCES

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE CVPR*, 2001, pp. I–511–I–518.
[2] S. Doungmala and S. Klubsuwan, "head gear wearing detection using image processing," in *Proc. Int. Conf. Digital Arts, Media and Technology (ICDAMT)*, 2016, pp. 45–50.
[3] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Proc. IEEE CVPR*, 2005, pp. 886–893.
[4] P. Silva, M. Luz, and F. Alves, "head gear detection on motorcyclists using image descriptors and classifiers," in *Proc. Int. Conf. Graphics, Patterns and Images*, 2013, pp. 141–148.
[5] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.
[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE CVPR*, 2014, pp. 580–587.
[7] R. Girshick, "Fast R-CNN," in *Proc. IEEE ICCV*, 2015, pp. 1440–1448.
[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.
[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, real-time object detection," in *Proc. IEEE CVPR*, 2016, pp. 779–788.
[10] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.
[11] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.
[12] G. Jocher, "YOLOv5 by Ultralytics," GitHub repository, 2020. [Online]. Available: https://github.com/ultralytics/yolov5
[13] W. Silva and A. Fernandes, "Motorcycle head gear detection using Faster R-CNN," in *Proc. IEEE SmartWorld*, 2019, pp. 1884–1889.
[14] K. Raj, M. Jaiswal, and S. Kumar, "A hybrid deep learning approach for motorcycle rider head gear detection," in *Proc. ICCCA*, 2020, pp. 1–6.
[15] P. Vishnu and N. Kumar, "head gear detection using YOLOv4 for real-time traffic surveillance," in *Proc. IEEE ICICA*, 2021, pp. 324–329.
[16] J. Chiverton, "Enhancing head gear detection using ensemble deep learning approaches," *J. Image Vision Comput.*, vol. 85, pp. 45–55, 2019.
[17] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, 2019.
[18] E. Rothe et al., "Data augmentation strategies for improved object detection in challenging environments," in *Proc. IEEE CVPR Workshops*, 2020, pp. 1020–1028.
[19] Y. Zhang, R. Hu, and L. Shen, "Edge-based head gear detection system using lightweight CNNs," in *Proc. IEEE IoTDI*, 2022, pp. 231–238.
[20] World Health Organization, *Global Status Report on Road Safety 2023*. Geneva, Switzerland: WHO Press, 2023.