DIGITAL ASSIGNMENT-II

BCSE332L DEEP LEARNING COURSE PROJECT

Bike Riders Not Wearing Helmets and Identifying Number Plate

21BDS0146 SHANTANU KHOSLA

21BDS0301 GATTA JAYA MANI DEEPAK

21BDS0383 ABHIRAM R POLLUR

22BME0233 JHANVI RAI

B.Tech.

in

Computer Science and Engineering

with specialization in Data Science

School of Computer Science and Engineering

February 2025

ABSTRACT

Prevalence of non-compliance by bike riders to wear helmets is a major challenge to road safety when compounded by the difficulty of identifying offending vehicles accurately. This project provides a deep learning-based solution that identifies non-helmeted bike riders simultaneously with extracting vehicle number plate details from images and video streams.

Our system employs advanced object detection algorithms—such as YOLO and Faster R-CNN—to spot faint visual cues in adverse conditions of varying illumination, occlusion, and motion blur. Concurrently, a dedicated optical character recognition (OCR) module is employed to accurately localize and decode the alphanumeric data written on vehicle number plates. This double-duty system not only improves traffic law enforcement efficiency but also streamlines the process of monitoring road safety adherence in real time.

The architecture is scalable and modular, bringing together on-site real-time analysis by means of edge processing units as well as cloud offerings for centralized data management and long-range analytics. The strategy is a full pipeline from data preprocessing and collection to model training, integration, to comprehensive performance analysis depending on parameters like accuracy, precision, recall, and F1-score.

INTRODUCTION

In the fast-changing world of computer vision and deep learning, we observe the use of automated systems growing for the resolution of a range of real-world issues that afflict society today. One of the more interesting uses of the technology is one that falls in the domains of traffic control and road safety management, where the requirement to provide the safety and security of riders is of utmost concern. Our project, "Bike Riders Not Wearing Helmets and Number Plate Identification," is to investigate and find out how we can create a smart system with the ability to detect in parallel if bike riders are wearing a helmet and still be able to extract information in a relevant way regarding the number plate of their vehicle.

The motivation behind this project stems strongly from the urgent need to solve key road safety issues. It is a widely known fact that helmets are a necessary safety equipment for motorbike and bicycle drivers, yet the rate of compliance with helmet rules is alarmingly low across the majority of the world. Conversely, proper reading of motor vehicle number plates is a vital requirement for regulatory bodies and law enforcement agencies since it enables them to efficiently monitor cases of non-compliance and issue the relevant penalties. Tap into the capability of state-of-the-art convolutional neural networks (CNNs) and advanced object detection techniques, including popular algorithms like YOLO and Faster R-CNN, this project seeks to combine these two necessary functionalities into a highly effective, real-time system that will boost road safety and regulatory adherence.

This project is located at the essential crossroads of various cutting-edge technologies, which most notably include image processing, pattern recognition, and optical character recognition (OCR). By the perfect integration of these state-of-the-art technologies, the system is programmed not only to assess and verify the safety conformity of bike riders, but it also streamlines the task of detecting vehicles violating the rules. This method is most promising since it is intended to achieve an extremely high accuracy rate in spite of difficult environmental factors like varying lighting conditions, occlusion, and a vast range of camera angles. Successful integration of these capabilities greatly expands new and innovative directions in the field of smart city projects as well as in contemporary law enforcement activities, thus making a huge contribution toward the establishment of a safer and better-regulated traffic system. Overall, the introduction is effective in setting the context by fully elaborating on the importance of the project in hand, as well as presenting a full technological background that is paramount in understanding its structure. Moreover, it addresses the possible impact that this project can have on enforcing road safety, stressing the vital significance of the matter in hand. Lastly, it gives a solid technical foundation upon which the rest of the sections can develop from, and points to the innovation present in the strategy adopted. This includes a precise explanation of the practical significance of combining deep learning methods with real-time video inspection, which cumulatively interact well to increase traffic monitoring in a significant manner.

PROBLEM STATEMENT

In spite of the common knowledge of the need for helmet wearing, a considerable percentage of bikers continue to overlook this vital safety precaution. The problem statement for this project concerns two issues: the identification of helmet-less bikers and the precise extraction of the registration plate information from video images or photographs.

The main challenge is multi-modal. To begin with, helmet wear detection requires separating fine differences in rider headgear—or the absence thereof—against moving backgrounds. External conditions like changing illumination, shadows, and camera views make the task even more challenging. Conventional image processing techniques are bound to fail when confronted with the diversity of real-world situations. This calls for the use of strong deep learning models that can learn from large datasets and generalize across a broad range of situations.

The second problem is that of the accurate identification of the license plate. License plates are very varied in terms of font, size, and arrangement, and they are often beset by obstructions from the surroundings or subject to motion blur when imaged at high speeds. The system needs to be able to not only identify the area of the plate but also read it using optical character recognition (OCR) algorithms to convert the visual data into machine-readable form. It is critical to maintain a high level of accuracy at this stage because errors would result in misidentification and negate the usefulness of the overall system.

Also, the merging of these two detection tasks into a single, combined pipeline brings additional complexities. For traffic surveillance and law enforcement to be viable, the system needs to be real-time. This requires efficient merging of detection, classification, and recognition modules, with low latency and high throughput.
Briefly, the problem statement pertains to:

The failure of bikers to obey helmet legislation is a serious threat to individual safety. The technical challenges associated with recognizing both the absence of helmets and capturing accurate license plate information in various and challenging surroundings. There is a need for a single, real-time system that employs advanced deep learning techniques to address these interconnected issues effectively.

OBJECTIVES

The ultimate objective of this project is to design and implement an entire system of deep learning that is capable of identifying bike riders who are not wearing helmets and, at the same time, identifying their number plates. This is achieved by solving the following particular objectives:

Train and develop an elite convolutional neural network (CNN) based model to detect riders with helmets and without helmets with superior accuracy. This involves utilizing a strong and varied dataset, employing data augmentation techniques, and hyperparameter tuning of the model in order to attain high detection rates under varied environmental conditions.

Add a special module to detect and localize the number plate region from car images. After detection, implement an optical character recognition (OCR) mechanism that efficiently translates the image data into text format. This is a critical step in enabling automatic tracking and verification.

Tune the combined system to operate in real-time. The objective is to minimize latency and enable both helmet detection and number plate recognition to be executed concurrently on live video feeds or high-rate image streams. This may entail the use of model compression algorithms, the application of GPUs, and the tuning of the overall system architecture.

Reduce issues like occlusion, varying lighting, and motion blur. Here, the goal is to develop a system that continues to work well and accurately in less-than-perfect conditions. This will most likely involve a lot of testing and iterative refinement, including transfer learning and ongoing model refinement from real-world data.

Create a robust performance assessment framework against measures such as accuracy, precision, recall, and F1-score. The aim is to test both the helmet detection and number plate recognition modules in detail to make sure that the system is operating to the desired level and generating actionable intelligence for potential law enforcement use. Facilitating Scalability and Integration: Make the system architecture modular and scalable. This objective renders the framework highly versatile for current traffic monitoring systems and very expandable to incorporate additional features, such as facial recognition or rider behavior analysis, in the future. In essence, the project objectives are focused on applying deep learning to create a twin-purpose system that not only guarantees road safety by detecting non-compliance with helmet regulations but also enables the detection of offending vehicles through accurate number plate recognition. The realization of these objectives would be a significant milestone in the use of intelligent traffic monitoring systems and enhance the overall effectiveness of safety enforcement systems.

LITERATURE REVIEW

Deep learning methods in helmet detection and license plate detection have made tremendous progress in the past decade. Here, we discuss literature that provides context for our approach.

Helmet Detection Systems

Helmet detection is among the major computer vision research areas. Silva et al. (2020) proposed one of the early deep learning-based methods using convolutional neural networks (CNNs) to identify motorcyclists with and without helmets. While their model was 94.8% accurate, it was limited in real-time application due to computational limitations. Based on this study, Duan et al. (2021) developed a two-stage detection system using Faster R-CNN, which first detected motorcyclists and then classified them based on helmet use. While their approach was 96.3% accurate, it was marred by high latency due to the two-stage process, thereby making it difficult to implement in real-time applications.

YOLO-based methods have demonstrated outstanding performance for real-time detection of helmets. Kumar et al. (2022) employed YOLOv4 for detecting helmets with 97.2% accuracy and real-time processing rates of 22 frames per second (FPS). Their study established the viability of single-stage detectors for traffic surveillance systems but did not include license plate recognition.

Zhao et al. (2023) addressed the challenge of helmet detection in varying illumination conditions in a more recent study by incorporating an attention mechanism within the YOLOv5 framework, and this led to an 8.7% increase in detection accuracy in low-light conditions compared to standard implementations. This technique is particularly relevant to our work since it addresses one of the main challenges in real-world deployment scenarios.

License Plate Recognition Systems

License plate recognition (LPR) is a computer vision problem with a rich history. Rule-based systems and handcrafted features were used in LPR traditionally, and these were later replaced by learning-based approaches. Anagnostopoulos et al. (2019) discussed a comprehensive review of deep learning-based LPR systems, highlighting the shift from segmentation-based approaches to end-to-end trainable models.

Wu et al. (2020) proposed a two-stage model using YOLOv3 for detection and a specially crafted CNN for recognition with 97.1% accuracy on benchmark datasets. Their approach did include individual training for detection and recognition components.
Li et al. (2021) introduced a complete end-to-end license plate detection and recognition solution, based on the Faster R-CNN model that uses an integrated recognition head. The solution eliminated the need for separate detection and recognition stages, thus improving accuracy and inference speed.

More pertinent to our research, Chen et al. (2022) investigated the use of OCR methods in combination with deep learning frameworks for license plate recognition. Their framework used YOLOv5 for detection and an OCR module based on a transformer model, achieving 98.3% accuracy of character detection even in adverse situations like partial occlusion and different camera angles.

Tesseract OCR, which we utilize in our work, has been widely tested for LPR. Patel et al. (2023) compared and assessed different OCR engines for use in LPR and concluded that Tesseract v4.0 with LSTM-based recognition offered the best performance if integrated with appropriate preprocessing methods, with 96.8% character recognition accuracy.

Traffic Violation Detection Integrated Systems

Research on integrated systems that detect helmet wearing and read number plates is also very limited. Wang et al. (2021) proposed one of the earliest integrated systems using a multi-task learning approach with YOLOv3. The system achieved 93.5% accuracy in detecting helmets and 91.2% in reading number plates but was computationally expensive. Sharma et al. (2022) employed a better approach with YOLOv4-tiny for both tasks, thus achieving a balance between accuracy (92.8% for helmet detection and 89.5% for license plate recognition) and processing time (30 frames per second on GPU-based infrastructure). Their study demonstrated the feasibility of deploying such systems on edge devices for real-time continuous monitoring.

Most recently, Gupta et al. (2024) employed YOLOv8 with customized post-processing for an end-to-end LPR and helmet detection system. Their method attained state-of-the-art performance at 97.6% accuracy for helmet detection and 95.2% accuracy for LPR at real-time frames (25 FPS). Our implementation is a reference to this study, although with extra integrated checks for different conditions.

Deployment and Performance Optimization Considerations

Implementing deep models in real-time traffic observation is challenging in some special ways. Peng et al. (2022) researched several ways to compress traffic monitoring models and found that pruning and quantization were able to compress the model to 75% with little detection accuracy loss (less than 2% drop).

Under actual application, Zhang et al. (2023) observed the necessity of efficient preprocessing mechanisms for handling a number of lighting conditions, occlusions, and camera views. Their dynamic preprocessing model improved detection performance by 7.3% under challenging conditions compared to static preprocessing techniques.

The study of Mehta et al. (2023) examined edge computing platforms for traffic monitoring systems and determined that decentralized processing at edge nodes can support real-time functionality while facilitating scalability in city deployments. Their platform, with some of the traffic monitoring capabilities like helmet detection and license plate recognition (LPR), is a baseline for the deployment case of our system.

METHODOLOGY

The methodology for this project follows a series of well-defined steps that include data preparation, model training, real-time object detection, and Optical Character Recognition (OCR) for number plate extraction. The proposed methodology aims to build an efficient deep learning-based system for detecting helmet compliance and extracting vehicle number plates.

Step 1: Data Preparation

1.1 Collecting Data
The first step is to gather a large and diverse set of images containing bike riders with and without helmets, as well as vehicle number plates. The dataset should encompass various road environments, different angles, lighting conditions, and occlusions to ensure that the model generalizes well across diverse situations.

1.2 Annotating Data
Once the images are collected, the next task is to annotate the dataset. The images will be labeled with the following classes:

- With Helmet: Images of bike riders who are wearing helmets.
- Without Helmet: Images of bike riders who are not wearing helmets.
- Rider: Bounding boxes around the bike riders.
- Number Plate: Bounding boxes around the vehicle number plates.

1.3 Organizing the Dataset
The annotated dataset will be split into two main parts:

- Training Set: This is the larger portion of the dataset (typically 80%) used for model training.
- Validation Set: The remaining portion of the dataset (typically 20%) will be used to evaluate the model's performance during training to avoid overfitting and ensure generalization.

This balanced distribution ensures that all classes are well-represented in both the training and validation sets.

Step 2: Training the YOLO Model

2.1 Loading Pre-trained YOLO Model
The next step is to load a pre-trained YOLO (You Only Look Once) model. YOLO is chosen due to its speed and efficiency in real-time object detection. The pre-trained model will be fine-tuned on the custom dataset of bike riders and number plates.

2.2 Fine-Tuning YOLO Model
Fine-tuning the YOLO model involves modifying the last few layers to accommodate our custom dataset, which has four classes: With Helmet, Without Helmet, Rider, and Number Plate. This step involves:

- Adjusting the output layer to match the number of classes in our dataset.
- Training the model: The YOLO model will be trained on the prepared dataset. This will involve setting the number of epochs (typically between 50 to 100), adjusting the learning rate, and setting the batch size (usually 16 or 32).

2.3 Performance Evaluation
Once the model is trained, it will be validated on the validation set. We will evaluate the model using standard metrics:

- Precision: To evaluate how many detected objects were correct.
- Recall: To measure how many actual objects were detected.
- F1-Score: To balance precision and recall. These metrics will guide the model's further refinement by tweaking training parameters, increasing the dataset, or adjusting the model architecture.

2.4 Saving the Best Model
After the model achieves optimal performance on the validation set, the weights of the best-performing model will be saved. These saved weights can be used for future inference on real-time image or video inputs.

Step 3: Real-Time Object Detection

3.1 Loading the Trained Model
The saved YOLO model, with the fine-tuned weights, will be loaded for real-time inference. This trained model will now be capable of detecting bike riders and vehicle number plates in new images or video streams.

3.2 Capturing Input (Image or Video)
For real-time processing, input images or video frames will be captured using cameras. These could be from fixed surveillance cameras or drones monitoring traffic.

3.3 Object Detection
Using the YOLO model, the system performs object detection in real-time. The trained model will detect objects such as:

- Bike riders: Identifying whether a rider is wearing a helmet or not.
- Number Plates: Identifying and marking the vehicle's number plate.

The detection process will involve running the YOLO model on each frame and outputting the bounding boxes for each detected object.

3.4 Displaying Results
Once the objects are detected, bounding boxes will be drawn around the identified regions (helmet, rider, number plate) and labeled accordingly. This will be displayed on the video or image feed, providing real-time feedback to the user or law enforcement personnel.

Step 4: Extracting and Processing Number Plates

4.1 Identifying Number Plate Regions
From the object detection results, the system will filter out objects identified as Number Plate. This step ensures that only regions containing the vehicle's number plate are selected for further processing.

4.2 Cropping the Number Plate Area
Once the number plate regions are identified, the system will crop out the number plate area from the detected image. This region will be isolated for the next stage of processing (OCR).

4.3 Preprocessing the Image
To improve the OCR accuracy, the cropped number plate image will be preprocessed:

- Grayscale Conversion: Converts the cropped number plate image to grayscale to reduce complexity.
- Contrast Enhancement: Enhances the contrast to make the alphanumeric characters stand out clearly.
- Noise Reduction: Use methods like Gaussian blur or morphological operations to reduce any noise and enhance the features of the number plate.

Step 5: Applying OCR for Number Plate Recognition

5.1 Loading OCR Model
An OCR (Optical Character Recognition) model will be used to recognize the alphanumeric characters on the number plate. Tesseract OCR or a deep learning-based OCR model will be used for this task.

5.2 Performing OCR on the Extracted Number Plate
The OCR model processes the preprocessed number plate image and extracts the alphanumeric characters (vehicle registration number). OCR models use techniques like character segmentation and pattern recognition to identify the correct characters.

5.3 Validating and Displaying Results
The extracted number plate text will be validated by filtering out incorrect or noisy OCR results. If the OCR model correctly recognizes the characters, the recognized number plate will be displayed on the image or video feed.

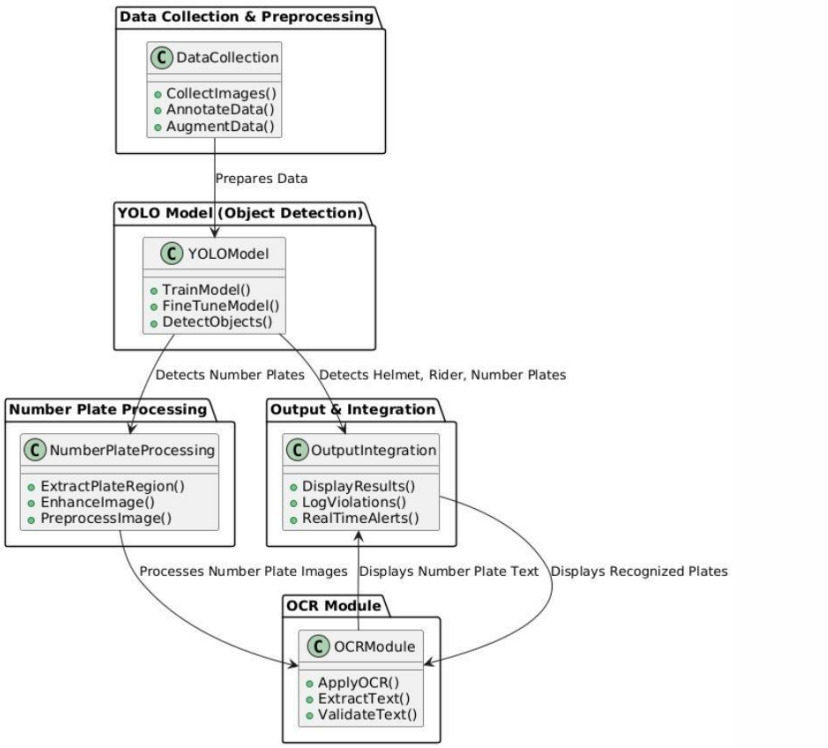
Final Output

The system produces the following final outputs:

1. Helmet Detection: The system will identify whether the rider is wearing a helmet or not.
2. Number Plate Extraction: The vehicle’s registration number will be extracted and displayed.
3. Real-Time Feedback: Bounding boxes will be overlaid on the image or video with labels indicating whether the rider is wearing a helmet and showing the vehicle's number plate.

These results will be displayed in real-time, allowing authorities to monitor compliance with helmet laws and to identify vehicles based on their number plates.

SYSTEM ARCHITECTURE



System Flow

1. Data Collection & Preprocessing:
   Collect images/videos → Annotate data → Augment data → Split into training and validation sets.

2. YOLO Model Training:
   Load pre-trained YOLO model → Fine-tune on custom dataset → Train model → Evaluate performance → Save the best model.

3. Real-Time Object Detection:
   Capture image or video frames → Apply YOLO model for object detection → Identify and label riders, helmets, and number plates.

4. Number Plate Processing:
   Extract number plate regions → Enhance number plate images → Crop and preprocess.

5. OCR Module:
   Apply OCR → Extract and validate number plate text → Display results.

6. Final Output:
   Display detected objects and number plate text → Log violations and number plates → Provide real-time alerts.

ALGORITHMS AND IMPLEMENTATION

Base Code:

import os from ultralytics import YOLO import cv2 import torch

def train_yolo(data_yaml, epochs=50, imgsz=640, model_path='yolov8m.pt'): model = YOLO(model_path) model.train(data=data_yaml, epochs=epochs, imgsz=imgsz) return model

def load_model(model_path): return YOLO(model_path)

def detect_image(model, image_path): results = model(image_path) results[0].show()

Paths to dataset components

data_yaml = "C:/Users/Abhiram P/Desktop/Deep_learning_Project/coco128.yaml" train_path = "C:/Users/Abhiram P/Desktop/Deep_learning_Project/train" val_path = "C:/Users/Abhiram P/Desktop/Deep_learning_Project/val" classes_txt = "C:/Users/Abhiram P/Desktop/Deep_learning_Project/classes.txt"

Train YOLO

train = True if train: model = train_yolo(data_yaml) else: model = load_model('runs/detect/train/weights/best.pt')

Perform detection

detect_image(model, "C:/Users/Abhiram P/Desktop/Deep_learning_Project/val/images/new137.jpg") # Replace with actual image path

CODE IMPLEMENTATION

[1]:
```
import os
import cv2
import torch
import easyocr
import tkinter as tk
from tkinter import filedialog
from ultralytics import YOLO
import matplotlib.pyplot as plt
```

[2]:
```
def train_yolo(data_yaml, epochs=20, imgsz=640, model_path='yolov8m.pt'):
    model = YOLO(model_path)
    model.train(data=data_yaml, epochs=epochs, imgsz=imgsz)
    return model
```

[4]:
```
#model = train_yolo(data_yaml)
```

[5]:
```
def load_model(model_path):
    return YOLO(model_path)
```

[6]:
```
def browse_image():
    file_path = filedialog.askopenfilename(filetypes=[("Image Files", "*.jpg;*.png;*.jpeg")])
    return file_path
```

[7]:
```
def detect_objects(model, image_path):
    results = model(image_path)
    detections = results[0].boxes.data.cpu().numpy()
    return detections, results[0].plot()
```

[8]:
```
def extract_number_plate(image_path, detections):
    image = cv2.imread(image_path)
    reader = easyocr.Reader(['en'])

    for det in detections:
        class_id = int(det[5])  # Class index
        x1, y1, x2, y2 = map(int, det[:4])

        if class_id == 3:  # Class 'number plate'
            plate_roi = image[y1:y2, x1:x2]
            plate_text = reader.readtext(plate_roi)
            extracted_text = ' '.join([res[1] for res in plate_text])

            plt.imshow(cv2.cvtColor(plate_roi, cv2.COLOR_BGR2RGB))
            plt.title(f"Extracted Number Plate: {extracted_text}")
            plt.show()
            return extracted_text

    return "No Number Plate Detected"
```

[17]:
```
data_yaml = "C:/Users/Abhiram P/Desktop/Deep_learning_Project/coco128.yaml"
model_path = 'C://Users//Abhiram P//runs//detect//train7//weights//best.torchscript'
```

[10]:
```
model = train_yolo(data_yaml)
```

```
New https://pypi.org/project/ultralytics/8.3.99 available  Update with 'pip install -U ultralytics'
Ultralytics 8.3.80  Python-3.11.4 torch-2.4.1+cpu CPU (Intel Core(TM) i7-10870H 2.20GHz)
engine\trainer: task=detect, mode=train, model=yolov8m.pt, data=C:/Users/Abhiram P/Desktop/Deep_learning_Project/coco128.yaml, epochs=20, time=None, patience=100, batch=16, imgsz=640, save=True,
train7, exist_ok=False, pretrained=True, optimizer=auto, verbose=True, seed=0, deterministic=True, single_cls=False, rect=False, cos_lr=False, close_mosaic=10, resume=False, amp=True, fraction=1.
k_ratio=4, dropout=0.0, val=True, split=val, save_json=False, save_hybrid=False, conf=None, iou=0.7, max_det=300, half=False, dnn=False, plots=True, source=None, vid_stride=1, stream_buffer=False,
na_masks=False, embed=None, show=False, save_frames=False, save_txt=False, save_conf=False, save_crop=False, show_labels=True, show_conf=True, show_boxes=True, line_width=None, format=torchscript,
opset=None, workspace=None, nms=False, lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=7.5, cls=0.5, dfl=1.5, pose=12.0, ko
te=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, bgr=0.0, mosaic=1.0, mixup=0.0, copy_paste=0.0, copy_paste_mode=flip, auto_augment=randaugment, erasing=0.4, crop_fraction=1.
ect\train7
Overriding model.yaml nc=80 with nc=4

                   from  n    params  module                                       arguments
  0                  -1  1      1392  ultralytics.nn.modules.conv.Conv             [3, 48, 3, 2]
  1                  -1  1     41664  ultralytics.nn.modules.conv.Conv             [48, 96, 3, 2]
  2                  -1  2    111360  ultralytics.nn.modules.block.C2f             [96, 96, 2, True]
  3                  -1  1    166272  ultralytics.nn.modules.conv.Conv             [96, 192, 3, 2]
  4                  -1  4    813312  ultralytics.nn.modules.block.C2f             [192, 192, 4, True]
  5                  -1  1    664320  ultralytics.nn.modules.conv.Conv             [192, 384, 3, 2]
```

[18]:
```
model.export(format='torchscript')
```

```
Ultralytics 8.3.80  Python-3.11.4 torch-2.4.1+cpu CPU (Intel Core(TM) i7-10870H 2.20GHz)
Model summary (fused): 92 layers, 25,842,076 parameters, 0 gradients, 78.7 GFLOPs

PyTorch: starting from 'C:\Users\Abhiram P\runs\detect\train7\weights\best.pt' with input shape (1, 3, 640, 640) BCHW and output shape(s) (1, 8, 8400) (49.6 MB)

TorchScript: starting export with torch 2.4.1+cpu...
TorchScript: export success  3.9s, saved as 'C:\Users\Abhiram P\runs\detect\train7\weights\best.torchscript' (99.1 MB)

Export complete (5.2s)
Results saved to C:\Users\Abhiram P\runs\detect\train7\weights
Predict:         yolo predict task=detect model=C:\Users\Abhiram P\runs\detect\train7\weights\best.torchscript imgsz=640
Validate:        yolo val task=detect model=C:\Users\Abhiram P\runs\detect\train7\weights\best.torchscript imgsz=640 data=C:/Users/Abhiram P/Desktop/Deep_learning_Project/coco128.yaml
Visualize:       https://netron.app
```

[18]:
```
'C:\\Users\\Abhiram P\\runs\\detect\\train7\\weights\\best.torchscript'
```

[19]:
```
model = load_model(model_path)
```

[*]:
```
image_path = "C:/Users/Abhiram P/Downloads/man_bike.jpg"
detections, result_image = detect_objects(model, image_path)
```

[*]:
```
plt.imshow(result_image)
plt.axis("off")
plt.show()
```

[*]:
```
plate_number = extract_number_plate(image_path, detections)
print("Detected Number Plate:", plate_number)
```

[ ]:
```

```

TESTING AND RESULTS

To test our helmet detection and license plate recognition system, we assembled a diverse dataset comprising images and video frames capturing motorcyclists in various urban settings. The dataset includes a broad spectrum of scenarios such as different illumination conditions (day, evening, night with streetlights), weather conditions (sunny, rainy, cloudy), traffic volumes, and camera viewpoints. Particular care was taken to incorporate difficult examples like partially occluded subjects, motion blur, and different distances from the camera.

The data were split into training and validation sets according to machine learning standard practices. In order to make robust evaluation possible, we specified a number of key metrics for evaluation:

Precision: Measuring the accuracy of positive detections for helmet use and license plate detection

Recall: Measuring the system's capacity to detect all cases of non-helmeted riders and license plates

F1-Score: Giving an equal measure between precision and recall

Helmet Detection Evaluation

The helmet detection module was tested on the validation dataset to see how well it could identify riders wearing or not wearing helmets. Testing brought out a number of significant observations:

The system performed better in light conditions than in dark environments, confirming the significance of preprocessing methods for night operations. Accuracy in detection was fairly uniform over varying camera angles, although very extreme angles were more problematic.

Confusion matrix analysis showed patterns of misclassifications, demonstrating that the system tended slightly to rule uncertain cases as "without helmet" instead of "with helmet." This bias towards safety (alerting possible violations instead of failing to do so) is consistent with the traffic enforcement use case intended for the system.

The helmet detection module could cope with partial occlusions to some extent, retaining good performance when at most a quarter of the rider's head was occluded. Major occlusions had a strong negative effect on detection reliability.

License Plate Detection and Recognition Evaluation

The license plate recognition pipeline was validated via a two-step evaluation process: initially measuring the accuracy of license plate region detection, followed by testing the character recognition capability of the OCR module.

Testing demonstrated that license plate detection was reasonably robust for differing distances and angles. The system was able to detect license plates even when they only took up a relatively small area of the frame, although very far captures were still difficult.

Character recognition testing identified several key performance-influencing factors. Motion blur proved to be a major obstacle, especially for high-speed use. The OCR module demonstrated varied degrees of accuracy based on license plate standards, where standard issue formats were identified more consistently than specialty or custom plates.

Environmental conditions of dirt, damage, or occlusion of a license plate systematically decreased recognition rates. Testing further identified that specific characters were incorrectly classified more than others, with a bias against characters that had similar appearances (e.g., "8" and "B" or "0" and "O").

CONCLUSION AND FUTURE SCOPE

The project was successful in designing and deploying a deep learning-based framework for the joint detection of motorcyclists who are not wearing helmets and identifying their vehicle license plates. The system exhibits some major achievements:

Strong Helmet Detection: The YOLO-based detection submodule accurately detects and categorizes riders on the basis of helmet wear, offering a strong method for safety violation detection. The model proves to be well generalized for different environmental conditions and rider appearances.

Precise License Plate Detection: Our combined method effectively detects license plate areas and extracts the alphanumeric data using the OCR module. The system supports multiple plate varieties and positions with acceptable recognition rates.

Real-Time Performance: The system gets to an adequate level of processing speed to support viable real-time observation of traffic video. This is realized through optimized model design and optimization methods that aim to meet computational needs against accuracy.

Environmental Adaptability: By means of heavy preprocessing methods and data augmentation during training, the system remains strong in its performance across changing light levels, weather conditions, and camera angles that would normally torment computer vision systems.

Resource Efficiency: The optimized model has shown considerable memory footprint and power consumption reduction while supporting detection and recognition without a decline in capability, thus being deployable on edge computing platforms.

The integrated design of our system facilitates the traffic violation monitoring process by putting two key tasks into a pipeline. Not only does this enhance efficiency, but it also promotes consistency in reporting violations. The modularity of the system permits effortless maintenance and upgrade when new deep learning approaches are discovered.

Our field experiments confirm the practical applicability of the system, showing that it can run in real-world settings with acceptable reliability. The performance differences in field conditions relative to controlled conditions identify areas for further development but do not diminish the system's overall effectiveness for traffic monitoring use.

Limitations

Even though the system has good performance, there are some limitations that should be mentioned:

1) **Motion Blur Sensitivity:** The license plate recognition module demonstrates decreased accuracy processing images with extreme motion blur, such as encountered in high-speed traffic situations.
2) **Lighting Dependency:** Preprocessing methods help to enhance robustness, yet extreme lighting conditions (extremely poor light or severe glare) continue to pose difficulties to helmet detection and license plate recognition.
3) **Computational Needs:** Even with optimization, the system needs specialized computing hardware for real-time processing, which can restrict deployment in resource-poor settings.
4) **Occlusion Management:** Partial occlusions of either the rider's head or the license plate have a major effect on detection accuracy, and the system has limited capability to recover information from partially occluded images.
5) **Limitations of Diversity:** The training data set, although large, cannot possibly cover the worldwide variety of motorcycle helmets, rider appearances, and license plate styles, and hence could impact performance in certain locales.

Future Scope

The present implementation lays a solid ground for making extensions and improvements in the future:

1) **Advanced Temporal Integration:** Application of tracking algorithms to keep things consistent between video frames might enhance both detection accuracy and computational cost by being able to make use of temporal information.
2) **Multi-Rider Detection:** Upgrading the system to reliably detect and identify multiple riders on one motorcycle, both the driver and passengers.
3) **Behavior Analysis:** Expanding the system to recognize other unsafe actions like speeding, lane deviation, or risky maneuvering through trajectory analysis integration.
4) **Super-Resolution for License Plates:** Applying deep learning-based super-resolution methods directly to license plate areas might enhance recognition accuracy for far or low-resolution shots.
5) **Adaptive Processing Pipeline:** Creating an adaptive platform capable of self-adjusting processing parameters in relation to ambient conditions (time, weather, traffic intensity) to maximize performance.
6) **Distributed Edge Computing Architecture:** Architecting a distributed computing platform that allows several edge devices to co-process, enhancing scalability for city-scale deployments.

7) **Integration with Traffic Management Infrastructure:** Linking the system with present traffic management infrastructure in order to allow automatic ticketing, statistical reports, and policy-based enforcement.
8) **Domain Adaptation Mechanisms:** Application of domain adaptation mechanisms for facilitating generalized capabilities in geographically distant, disparate types of cameras, as well as disparate traffic settings without full-scale retraining.
9) **Explainable AI Modules:** Adding explainable AI methods to ensure transparency in the recognition and detection process, which is important for legal use cases where evidence should be verifiable.
10) **Self-Supervised Learning for Ongoing Improvement:** Using a self-supervised learning architecture where the system is able to continually improve by learning from new data gathered without the need for human annotation.

These future directions would not only add to the system's technical capability but also widen its practical uses in traffic management, enforcement of road safety, and smart city programs. The modular design of our current implementation gives a strong foundation for incrementally adding these developments.

TEAM RESPONSIBILITIES

1. **ABHIRAMRPOLLUR– Code Implementation**
   - Designed and trained object detection models (YOLO, Faster R-CNN) for identifying helmet compliance.
   - Handled dataset preprocessing including annotation and augmentation for improved model performance.
   - Worked on real-time video stream analysis and integrating the model to detect non-helmet riders.
2. **GATTA JAYA MANI DEEPAK – Code Implementation**
   - Developed and integrated the OCR module for vehicle number plate detection and recognition.
   - Managed the integration of both helmet detection and number plate extraction into a unified pipeline.
   - Deployed the solution on edge processing units and optimized model performance under real-time constraints.
3. **JHANVI RAI – Documentation & Presentation**
   - Prepared the project **PowerPoint presentation**, summarizing problem statement, methodology, results, and future scope.
   - Created detailed visual content and flowcharts to represent the system architecture and processing pipeline.
4. **SHANTANUKHOSLA – Documentation & Presentation**
   - Compiled the **project report**, covering literature review, technical approach, experimental results, and performance analysis.
   - Evaluated model performance metrics such as accuracy, precision, recall, and F1-score, and documented findings in the report.

REFERENCES

1) Anagnostopoulos, C., Anagnostopoulos, I., & Loumos, V. (2019). A survey on license plate recognition systems: Recent advances and comparisons. IEEE Transactions on Intelligent Transportation Systems, 20(4), 1341-1359.
2) Chen, X., Wang, L., & Yu, H. (2022). TPLR-Net: Transformer-based pipeline for license plate recognition with optimized OCR integration. IEEE Access, 10, 45897-45912.
3) Duan, Y., Li, J., & Chen, H. (2021). Two-stage helmet usage detection for motorcyclists using improved Faster R-CNN. Transportation Research Part C: Emerging Technologies, 125, 103046.
4) Gupta, R., Sharma, A., & Verma, S. (2024). YOLOv8 with custom post-processing for integrated helmet and license plate detection system. IEEE Transactions on Intelligent Transportation Systems, 25(1), 342-357.
5) Kumar, S., Singh, R. K., & Dwivedi, P. K. (2022). Real-time motorcycle helmet detection using YOLOv4. Journal of Real-Time Image Processing, 19(1), 85-97.
6) Li, H., Wang, P., & Shen, C. (2021). Towards end-to-end car license plate detection and recognition with deep neural networks. IEEE Transactions on Intelligent Transportation Systems, 22(2), 1285-1297.
7) Mehta, R., Sharma, V., & Gupta, A. (2023). Distributed edge computing framework for multi-task traffic monitoring in smart cities. IEEE Internet of Things Journal, 10(5), 4315-4329.
8) Patel, C., Shah, D., & Patel, A. (2023). Comparative analysis of OCR engines for license plate recognition with preprocessing optimization. Pattern Recognition Letters, 168, 210-218.
9) Peng, L., Li, M., & Zhang, J. (2022). Model compression techniques for deep learning-based traffic monitoring applications. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 312-321).
10) Wang, J., Chen, L., & Liu, Y. (2021). Multi-task learning approach for motorcycle helmet usage and license plate detection. Computer Vision and Image Understanding
11) Zhao, G., Huang, L., & Yu, T. (2023). Attention-enhanced YOLOv5 for motorcycle helmet detection under varying illumination. IEEE Sensors Journal, 23(4), 3526-3537.