Proceedings of the International Conference on Industrial Engineering and Operations Management

Publisher: IEOM Society International, USA DOI: 10.46254/WC02.20250297
Published: October 14, 2025

Situation-Aware Deep Learning for Urban Rideshare Safety: System Design and Development of YOLO-Powered Object Detection and Visual Analytics

Wei-Lun Huang, Roosan Liyons and Roger J. Jiao
H. Milton Stewart School of Industrial and Systems Engineering
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology, Atlanta, GA, USA

Abstract

Rideshare scooters are increasingly popular in urban cities. However, rider safety remains a key challenge. This paper presents a situation-aware Urban Rideshare Safety (URS) framework that merges smartphone sensing with deep learning–based visual analytics. The system uses a smartphone's built-in camera to apply the YOLO-based object detection algorithm. It identifies road surfaces, sidewalks, and curbs in real time. A labeled urban ride image dataset is used to train and validate detection models. The framework has two layers: smartphone sensors collect data, and cloud servers handle deep learning. These servers generate structured visual outputs that support safety indicators. The results show deep learning object detection is feasible for urban rideshare contexts. This establishes a foundation for future multimodal systems that combine visual, vibrational, and reasoning models. Using accessible smartphone technology, the system creates a foundation for intelligent assistance to improve rideshare safety.

Keywords
Urban Rideshare Safety, YOLO, Deep Learning, Requirement Analysis, Object Detection, Visual Analytics

1. Introduction
Open-air rideshare options, such as bikes and scooters, have rapidly become increasingly popular in urban transit. Their low cost, flexibility, and simple deployment make them attractive and accessible to the public. However, unlike automobiles, rideshare equipment isn't governed by strict regulations and built-in safety features, which presents safety risks not only to riders but also to pedestrians and the surrounding areas. These concerns create an urgent need for approaches that can prevent potential accidents and improve rideshare safety.

Many rideshare bike and scooter companies have added mechanical features, such as suspensions or speedometers. While these conventional enhancements improve rider comfort, they fail to deliver real-time safety insights. A significant gap persists in the deployment of intelligent, cognition-based assistance systems. Such systems can proactively respond to the riding environment and support decision-making. Recent developments in context-aware systems present promising solutions for progress. Advances in mobile phone sensors and deep learning facilitate the monitoring of riding conditions and the dynamic interpretation of contextual information. Mobile device sensors capture data, while deep learning–driven computer vision enables robust visual data analysis. This transformation from passive sensing to situational intelligence supports real-time safety assessments and proactive rider assistance.

Object detection and visual analytics are critical components for situational awareness. Computer vision algorithms process digital images to extract information, facilitating object detection. Importantly, with the integration of high-resolution smartphone cameras, these capabilities can be achieved without specialized hardware. Among deep learning approaches, the YOLO family of models has demonstrated leading performance in real-time detection tasks, offering both speed and accuracy suited for mobile deployment. A visual analytics framework transforms sensor input into a

safety score, reflecting the rider's surroundings. This integration bridges the gap between perception and reasoning, laying the foundation for next-generation URS systems.

In this regard, this paper proposes the design and development of a YOLO-powered, situation-aware deep learning framework for urban rideshare safety. The remainder of the paper is organized as follows: Section 2 reviews related work on rideshare safety, context-aware applications, and computer vision. Section 3 presents requirement analysis and system modeling. Section 4 describes the system's design and architecture. Section 5 details implementation and data processing. Section 6 discusses training results, outlines limitations, and suggests directions for future research.

2. Related Work

Smart sensors and deep learning have enabled cognitive assistance in applications such as smartphones, vehicles, home devices, and healthcare (Jahangir *et al*, 2019). While autonomous systems still face perception and prediction challenges, simpler decision-making tasks are possible with lightweight hardware. Prior studies have applied these methods to pavement crack detection, elderly care, and elevator dispatching. Integrating simple sensors with efficient algorithms supports cognitive assistance and decision making, forming a basis for this work.

Smart sensors and deep learning have been studied in both industry and academia, enabling cognitive assistance across everyday applications such as smartphones, vehicles, home devices, and healthcare (Jahangir *et al*, 2019). While autonomous systems still face challenges in perception and prediction, simpler decision-making tasks can be achieved with lightweight hardware. Prior studies have shown effective applications of these methods in domains such as pavement crack detection, elderly care monitoring, and elevator dispatching. These studies show that the integration of simple sensors, combined with efficient algorithms, can support cognitive assistance and decision making, forming a foundation for the work presented in this paper.

2.1 Computer Vision and Deep Learning for Object Detection

Computer vision is widely used to classify and detect road elements, such as sidewalks, cracks, and holes. Convolutional Neural Networks-based (CNN-based) pavement image analysis can detect cracks and classify surface conditions (Zhang et. al., 2016). Similarly, other studies used K-Nearest Neighbor (KNN), Support Vector Machines (SVMs), and Random Forests with sensor data to detect road and riding conditions (Ashqar *et al*, 2020; Jahangir *et al,* 2015). These studies show sensor integration and deep learning are adaptable for real-world decision support, forming the background for the work presented in this paper.

2.2 Smartphone Sensors for Surface Monitoring

Ambient data detection using smart sensors has been widely adopted in home security and automotive systems. Advances in sensor technology have made these devices efficient and affordable, enabling deployment in many environments. Prior work shows that sensors, when integrated with decision strategies, can provide effective cognitive assistance. For example, research combined RFID tags, motion detectors, and cameras with Rough Set Theory and Case-Based Reasoning (CBR) to monitor activities in elderly care homes (Zhou *et al,* 2011). These studies show that smart sensors and decision-making algorithms can yield lightweight, intelligent cognitive systems. This forms a foundation for their use in urban rideshare safety.

2.3 Edge Detection as a Lightweight Visual Tool

Classical vision methods, such as canny filters and Hough Transform, have been used to detect edges, cracks, and curb boundaries (Yachida & Tsuji, 1980). Advances in computer vision have enabled part classification, template identification, and fault detection in manufacturing operations (Agin, 1980). These studies show that even simple image-processing techniques aid in safety monitoring and demonstrate the feasibility of using smartphone cameras as a data source for visual analysis and edge detection. not only practical in industrial contexts but can also inform riding condition assessment in Urban Rideshare Safety systems. Together, these works form an early foundation for vision-based safety research, not only in industrial contexts but also in riding condition assessment in rideshare safety.

3. URS Requirement Analysis and Functional Modeling

Developing a situation-aware framework for urban rideshare safety requires clear system requirements and functional modeling. The system must integrate sensing, perception, and analytics to enhance rider safety while operating within the constraints of smartphones and lightweight hardware.

3.1 Driving Factors for System Development

Recent advances in mobile computing and networks have enabled efficient data transfer and processing on consumer devices (Huang *et al,* 2012). Now, real-time images can be captured and analyzed in mobile environments. Advances in deep learning and computer vision provide powerful techniques for object detection and situational awareness. Models such as YOLO are designed to be both accurate and fast. This balance of accuracy and speed makes YOLO suitable for detecting roads, sidewalks, and curbs in real rideshare settings.

The proposed system builds on these developments by combining physical data acquisition with deep learning–based visual analytics. In this framework, the primary hardware layer collects data from the smartphone, while the secondary hardware layer performs object detection and analysis. These layers must operate in synchronization to provide reliable, real-time safety outputs. Recent development of 4G and 5G networks allows fast and stable data transfer between layers. This enables captured data to move to the secondary layer for analysis with little delay.

The goal is to study the feasibility of a system that leverages smartphone sensors for data acquisition and deep learning-based models for perception. By doing so, the framework creates situational awareness that highlights road surfaces, curbs, obstacles, and the surrounding environment. Figure 1 illustrates this information hierarchy, where raw data captured from smartphone sensors is transformed into contextual features through deep learning–based perception. The final stage provides interpretable outputs in the form of visual analytics that riders can easily understand.

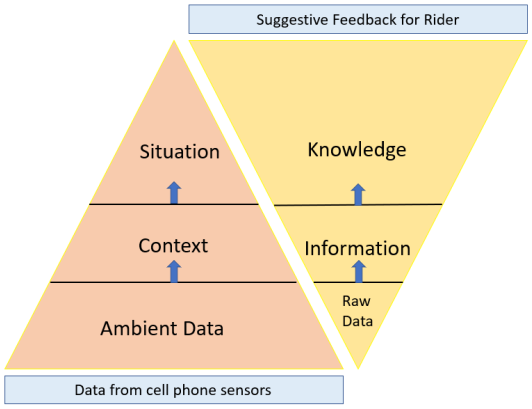

Figure 1. The Information Hierarchy for Ambient Data Analytics.

3.2 Situation-Aware Deep Learning (SaDL) System Requirements

As riders are already required to carry personal smartphones to access the features in the rideshare apps, smartphones become the natural device to support data collection. The sensors built into smartphones can be readily used to detect the ambient environment by extending the app's functionality. This step happens in the primary hardware layer, which includes the device and the app interface through which the rider interacts with the system.

The system must continuously observe ambient riding conditions and capture relevant data. Visual input is essential, as images provide the most reliable source of environmental information. These inputs are processed in the secondary hardware layer using deep learning–based object detection. YOLO models are particularly suitable, as they combine high accuracy with real-time performance (Huang et al., 2012). Detection results are then transformed into situational cues, such as bounding boxes, labels, and confidence scores that highlight roads, curbs, obstacles, and surrounding agents. A stable flow of information between acquisition and processing ensures that output remains synchronized with the riding environment.

In addition to functionality, the system must also satisfy key performance requirements to be viable in practice. Detection must operate close to real time, since delayed results reduce the value of safety feedback. Lightweight deployment is equally important, as the framework should run on consumer smartphones without requiring specialized hardware. Tests with devices such as the Google Pixel 3a indicate that this is feasible, with deep learning algorithms like YOLO and R-CNN able to achieve this goal. Algorithms that maximize efficiency while not sacrificing accuracy are key to obtaining the ideal results. Finally, the system must remain robust across diverse urban conditions, including

changes in lighting, weather, and device orientation. Techniques such as data augmentation during training can help maintain accuracy in these variable environments.

3.3 URS Functional Modeling

The functional model of the proposed URS illustrates how raw sensor inputs flow from collection to perception and analytics, ultimately creating situational awareness for riders. The model is designed to remain lightweight, practical, and deployable on smartphones. It consists of three functional layers. First, it begins with data acquisition, where the smartphone camera captures images or video frames during rides. This serves as the primary input for monitoring road and sidewalk conditions, curbs, and potential obstacles. The captured images are then processed through a YOLO-based model, which performs real-time object detection. The output consists of bounding boxes, class labels, and confidence scores for detected objects such as roads, sidewalks, pedestrians, vehicles, and obstacles, describing the objects present in the scene.

The final stage of the model is visual analytics, where the detection results are transformed into outputs that riders can interpret quickly. Instead of raw sensor readings, riders receive visual cues such as bounding boxes on the screen, object counts, or simple indicators of potential risks. This layered flow ensures that each component—data acquisition, YOLO-based perception, and analytics—operates in synchronization to provide real-time safety awareness. The detailed user input and system level activities are shown in the IDEF0 diagram in Figure 2.

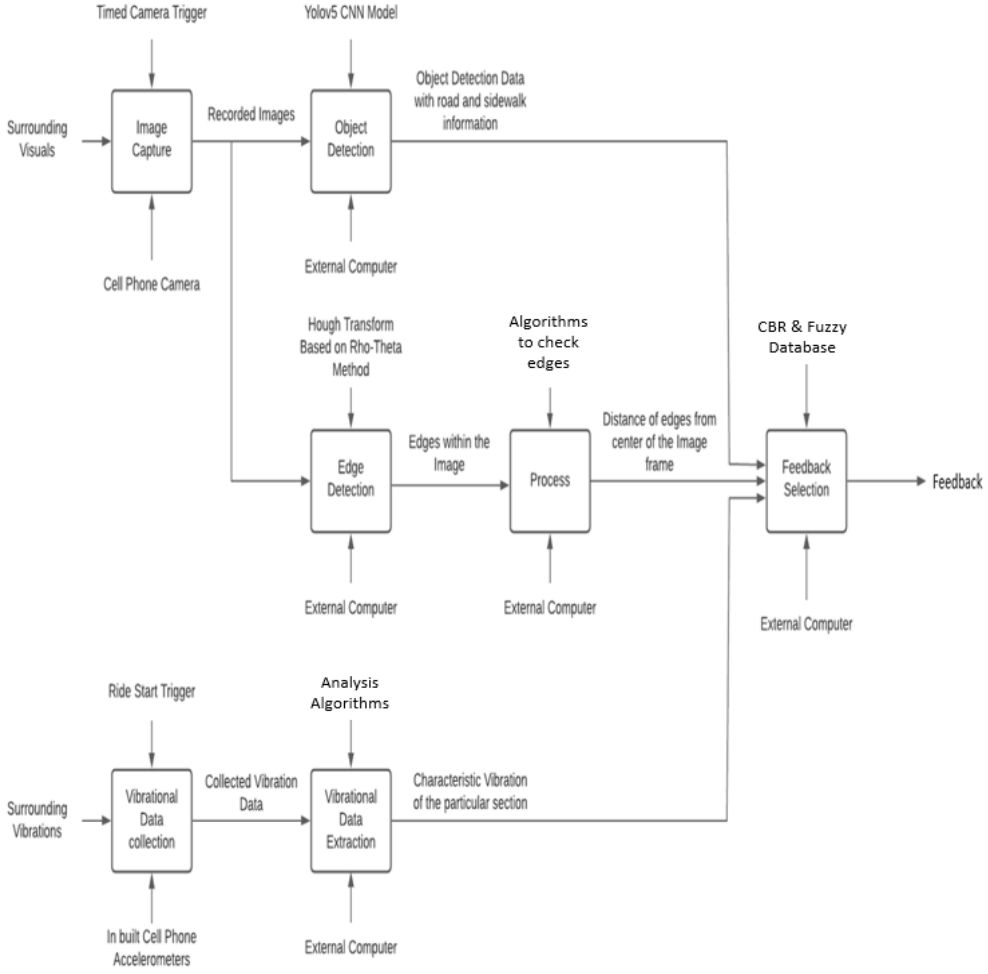

Figure 2. The IDEF0 Diagram of the Functional Model in SaDL for URS

4. SaDL Workflow Design and System Architecture

The system design translates these requirements into a working framework. We previously defined what the system must satisfy in the requirement analysis. This section describes how these needs are realized through architecture, modules, and workflow. The focus is on the integration of smartphone sensing with YOLO-based visual analytics. The design aims to provide a lightweight, real-time, and interpretable system that supports urban rideshare safety.

4.1 SaDL Workflow Design

The system moves from data input to processing and finally to interpretable outputs. Figure 3 presents this process. It begins with data collection, where the smartphone camera captures images of the riding environment at regular intervals. These images provide the raw context of the ride. The data is then transferred to a secondary layer through 4G networks, allowing heavy computation to be offloaded to lightweight servers or cloud systems when local resources are not sufficient. Once received, the data undergoes preprocessing, where images are resized or normalized to meet YOLO input requirements and vibration signals are smoothed to reduce noise. This ensures data quality before detection. In the detection stage, the YOLO model processes each image and generates bounding boxes, class labels, and confidence scores that identify roads, sidewalks, curbs, and other features of the environment. The results are refined during post-processing, where redundant detections are removed and the outputs are formatted into structured analytics. Finally, the system produces bounding boxes, labels, and confidence values as interpretable outputs. These outputs form the foundation for assessing safety and developing indicators that riders can quickly understand.

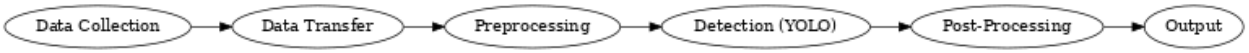

Figure 3. SaDL System Workflow

4.2 System Architecture

The system is designed as a two-layer architecture that completes the above-mentioned processes, linking smartphones with deep learning analytics. As shown in Figure 4, the primary layer is the sensors in the smartphone. It acts as the main sensing and collection device. The built-in camera periodically captures images of the road and the surrounding environment. This raw data is stored temporarily on the phone and then transmitted to the secondary hardware layer, where object detection and analysis are performed using deep learning.

The secondary layer is a lightweight server or cloud platform. Once data is received, preprocessing tasks such as image resizing, normalization, and noise filtering are applied. The YOLO model then processes the images in real time, performing visual analytics. It detects objects such as sidewalks, roads, and curbs, and outputs bounding boxes, labels, and confidence scores. By placing these computation tasks on the server, the system avoids overloading the phone while still providing near real-time feedback. The two layers should operate in real time. The smartphone collects and transfers data. The server processes the data and produces structured detection results. These results represent interpretable outputs that form the basis for visual analytics and future safety indicators. This architecture balances performance with accessibility.

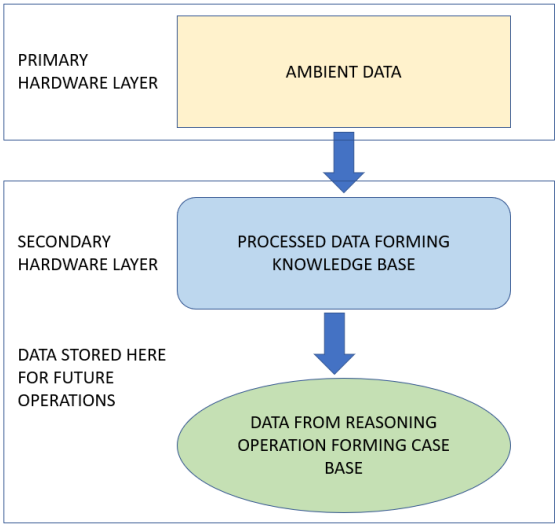

Figure 4. Primary and Secondary Hardware Layers in the SaDL URS System

The overall interaction between the two layers is further represented by using structured modeling tools. The IDEF3 diagram in Figure 5 extends the IDEF3 diagram in Figure 2 into a sequential representation, showing how data flows step by step from smartphone capture to preprocessing, YOLO detection, and structured analytics, ensuring that no stage becomes a bottleneck in real-time operation. Together, the two-layer design and the workflow models form a practical and deployable framework for urban rideshare safety. The smartphone serves as a natural entry point, since riders already use it to access rideshare apps, while lightweight cloud resources provide the computational power needed for deep learning. This balance allows the architecture to remain accessible, scalable, and capable of producing interpretable results that form the foundation for visual analytics and future safety indicators.

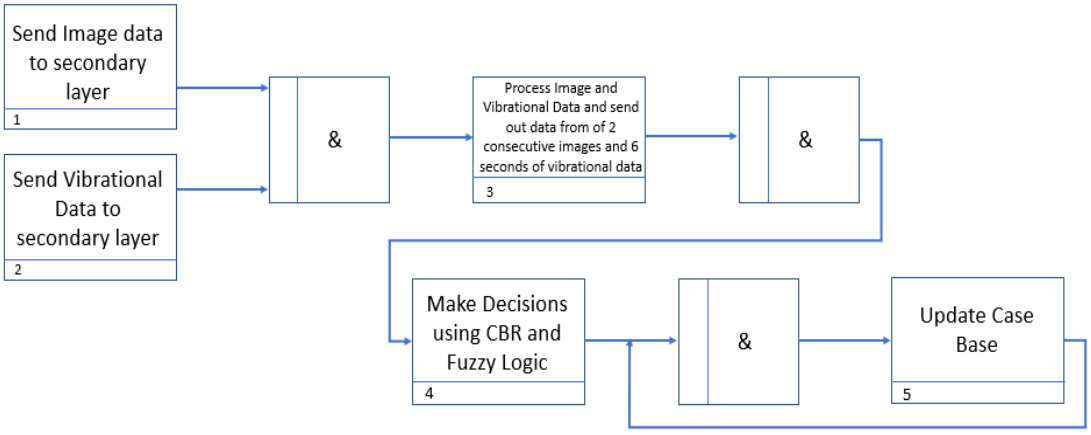

Figure 5. The IDEF3 Diagram Showcasing the Data Processing and Decision-Making Details in SaDL URS

4.3 Safety Metric Development

The YOLO-powered URS framework generates bounding boxes, labels, and confidence scores for detected objects, but riders need feedback that is simple and immediately useful. A safety metric offers a way to convert these raw outputs into an interpretable measure of riding conditions. By aggregating results across key categories such as sidewalks, curbs, and road surfaces, and applying weights to reflect their relative safety impact, the system could normalize scores into a range (e.g., 0–100) or classify them into levels such as Safe, Caution, or Unsafe. This would transform YOLO outputs into assistive guidance, enhancing situational awareness without overwhelming the user.

Although not implemented in this study, developing such a metric is a natural next step to extend the URS framework from detection and analytics toward a comprehensive safety support system for urban rideshare environments.

5. Development and Implementation of YOLO-Based Object Detection and Visual Analytics

The development phase focuses on implementing the YOLO-powered detection system for urban rideshare safety. While the overall architecture was outlined previously, this section describes how the system was built, how the dataset was prepared, and how the detection algorithm was applied to real-world data.

5.1 Visual Data Interactions in URS System Architecture

The system leverages smartphones as the primary sensing device, reflecting their ubiquity in rideshare operations. Riders already depend on smartphones to access rideshare services, making them the natural platform for safety monitoring. Two types of data streams are considered: visual data and vibrational data. While vibrational data isn't included in the conceptual design, the current work emphasizes visual data as the foundation for object detection.

Visual data is captured through a mounted smartphone camera at periodic intervals during rides. This provides snapshots of the riding environment, including sidewalks, curbs, and road conditions. These images are transferred to the secondary layer for processing. The interaction between the layers follows the workflow designed earlier: data acquisition, preprocessing, detection, and output generation. The comprehensive URS model would ideally be implemented in 4 steps, as shown in Figure 6. The present work implements the first two steps, focusing on image collection and visual data analysis with YOLO-based object detection.

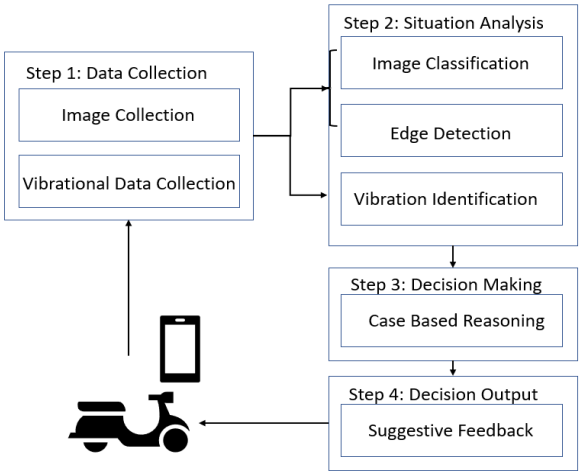

Figure 6. The Visual Data Interaction in the SaDL

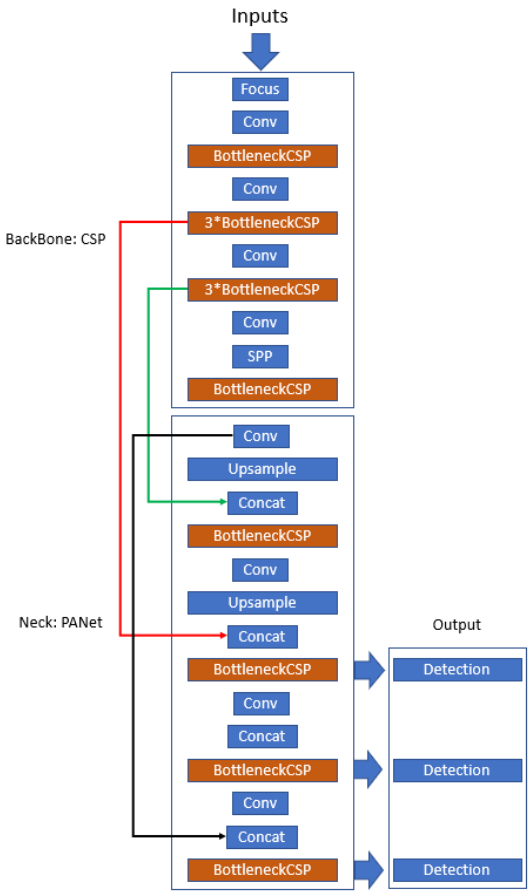

Figure 7. The Structure of YOLO Algorithm

5.2 YOLO-Powered Object Detection Algorithm

Object detection algorithms typically consist of a Backbone for feature extraction and a Head for predicting bounding boxes and class labels. More recent architecture also adds a neck to aggregate feature maps and improve detection

accuracy (Vandenhende et al., 2021; Ma et al., 2018). Among these, the YOLO (You Only Look Once) family has become widely used due to its efficiency and speed (Laroca *et al,* 2018; Maeda *et al,* 2018).

YOLO is a CNN–based algorithm for object detection. YOLOv5 represents one of the latest iterations in this family. It builds on YOLOv4 and improves both speed and accuracy (Yan *et al,* 2021). The architecture consists of three main components, including a backbone for feature extraction, a neck for multi-scale feature aggregation, and a detection head that outputs bounding boxes and class probabilities (Yang *et al,* 2017). These design choices enable high performance while maintaining efficiency. The backbone aggregates images and extracts visual features (Yan *et al,* 2021). The first focus layer reduces computational load by slicing the input image into patches. For a default input size, this step produces feature maps of a specific size. The BottleneckCSP module then extracts deep features through residual connections that combine a convolutional layer with batch normalization and Hardswish activation (Yan *et al,* 2021). To further expand the receptive field, a Spatial Pyramid Pooling (SPP) module applies three parallel max-pooling layers and outputs a fixed-size vector, improving robustness to scale variation (Yan *et al,* 2021).

The neck employs a Feature Pyramid Network (FPN) and Path Aggregation Network (PANet) to merge features across scales, enhancing detection performance for both small and large objects (Yang *et al,* 2017; Yan *et al,* 2021). The detection head then applies anchor boxes to generate bounding boxes, class probabilities, and confidence scores (Vandenhende *et al,* 2020; Ma *et al,* 2018). Figure 7 illustrates the structure of the YOLO algorithm and highlights the flow of features through its backbone, neck, and detection layers. YOLOv5 further benefits from lightweight design choices that make it suitable for mobile deployment. It can process up to 140 frames per second while remaining memory-efficient compared to other deep learning models (Yan *et al,* 2021). The algorithm utilizes activation functions such as Leaky ReLU and Sigmoid, and it is trained with gradient descent (Dubey & Jain, 2019; Tsmots & Ignatyev, 2019; Ruder, 2016). This balance of efficiency and accuracy makes it particularly well-suited for safety applications in dynamic urban rideshare environments.

In this work, different versions of YOLO have been performed to explore the validity, and YOLOv5 has been adapted to this system due to its balance of speed, accuracy, and computational efficiency compared with prior versions (YOLOv3/YOLOv4) and alternative detectors (e.g., EfficientDet). The YOLOv5 family includes several variants. Different YOLOv5 variants (e.g., v5s, v5l, v5x) provide trade-offs between accuracy and inference speed. The following metrics have been applied to directly measure how well each model detects road features.

- **Precision:** the proportion of correct detections among all predicted detections.
- **Recall:** the proportion of correct detections among all ground-truth objects.
- **mAP (mean Average Precision):** the overall detection accuracy across all classes.
- **Inference speed:** the average processing time per frame, showing the feasibility of real-time deployment.
- **Model size:** the storage required for deployment on lightweight servers.

Table 1 shows a comparison of YOLOv5 variants and other YOLO algorithms. The YOLOv5s variant was selected for this study, as it maintains a smaller model size and faster inference while still delivering competitive detection accuracy. The YOLOv5s model achieved a mean Average Precision (mAP 0.5) of 56.8%, with an inference speed of 98 ms per frame on CPU and a model size of 14 MB. These results confirm that YOLOv5s is effective for urban rideshare safety applications. While larger YOLO models achieve higher accuracy, their inference speeds are less suitable for real-time use. The performance of YOLOv5s shows that a practical balance of detection accuracy and speed can be achieved for lightweight, scalable systems.

Table 1. Performance Comparison of YOLO Variants

Model	Size (pixels)	mAP 0.5:0.95	mAP 0.5	Speed CPU (ms)	Model size (MB)
YOLOv5n	640	28.0	45.7	45	–
YOLOv5s	**640**	**37.4**	**56.8**	**98**	**14.0**
YOLOv5l	640	49.0	63.7	430	–
YOLOv5x	640	50.7	68.9	766	–
YOLOv3	640	–	–	0.053 s/pic	235.0
YOLOv4	640	–	–	0.017 s/pic	244.0
EfficientDet-D0	640	–	–	0.038 s/pic	15.0

5.3 Dataset Preparation and Training

To develop the dataset for YOLO training, a Pixel 3a smartphone was mounted on an e-scooter to capture images during real-world urban rides. This design choice reflects the intended use case, as riders already rely on smartphones to access rideshare services. As shown in Figure 9, the collected images were labeled using Makesense.ai, with bounding boxes drawn around key objects such as sidewalks, curbs, and road segments. The annotations were stored in YOLO-compatible text format. Object classes and dataset paths were then defined in YAML format, as shown in Figure 8. It was divided into training and validation sets with a 90/10 split. The dataset is then used to train the model to identify the ambient conditions in which the scooter is travelling.

```
train: ../train_data/images/train/  # train images (relative to 'path')
val: ../train_data/images/val/  # val images (relative to 'path')
test:  # test images (optional)

# Classes
nc: 2  # number of classes
names: ['Road','Sidewalk']  # class names
```

Figure 8. Defining Object Classes and Dataset Paths for Training and Validating Image Data in YAML Format

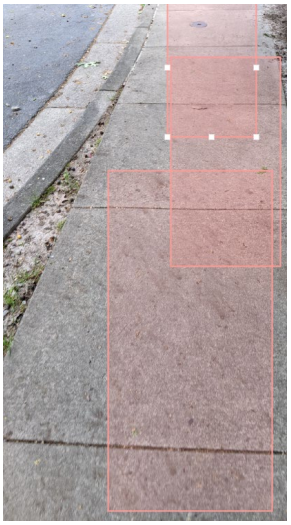

Figure 9. Labelling Sidewalk Images Using Makesense.ai

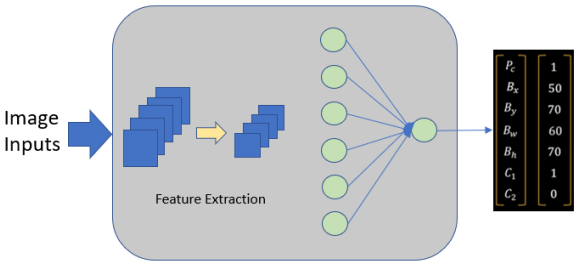

Figure 10. Structure of YOLO-Based Algorithm for Object Detection

The YOLOv5 framework processes input images by dividing them into structured tensors with 7 variables [*Pc Bx By Bw Bh C1 C2*]. Here, *Pc* is the probability of class (road/sidewalk), *Bx* and *By* represent the initial coordinate of the bounding box, *Bw* and *Bh* represent the width and height of the bounding boxes, and *C1* and *C2* represent the classes (Patel, 2020; Yan *et al,* 2021). Figure 10 demonstrates the structures involved in the YOLOv5 algorithm training, covering class probability, bounding box coordinates (x, y, width, height), and class identifiers. Each image will go from feature extraction through the backbone and prediction to the detection head.

The final outputs of the training stage are bounding boxes, class labels, and confidence scores for sidewalks, roads, and curbs. These structured detections form the core of situational awareness in the proposed system. More importantly, they demonstrate that raw smartphone inputs can be transformed into interpretable safety-relevant features through a lightweight deep learning framework. While this study focuses on the feasibility of YOLO for detection tasks, the results also lay the groundwork for broader safety assessment metrics to be developed in future.

6. Validation Experiment and Performance

6.1 Training Performance of YOLO

The proposed solutions are verified, and the YOLOv5 model was trained and tested on the prepared dataset of urban rideshare images collected with a Google Pixel 3a smartphone attached to the rideshare scooter. The evaluation focused on verifying whether YOLO can reliably detect key surfaces such as roads and sidewalks. Initial training with YOLOv5s on 300 labeled images successfully detected road and sidewalk segments, achieving confidence scores up to 85% for roads and 79% for sidewalks. These findings confirm that the YOLO-based algorithm can detect ambient riding conditions, validating its feasibility for urban rideshare safety. There are different variations of YOLOv5 that have different balances between accuracy and efficiency. Therefore, training should also be performed to compare the trade-off, particularly for the YOLOv5x, which has slower processing times but with slightly higher accuracy compared to YOLOv5s. Both were trained for 60 epochs and a batch size of 5. As shown in Figures 11 and 12, YOLOv5x yielded a slightly higher mAP@0.5 (27.1% vs. 24.2% for YOLOv5s), but required nearly eight times longer training time (1.040 hours vs. 0.137 hours). This illustrates the practical constraint that higher-capacity models may not always be optimal for lightweight, real-time deployment where computational efficiency is critical.

Extending YOLOv5s training to over 200 epochs improved detection accuracy, with the model reaching a mAP of 43.2% and confidence scores exceeding 70% across most test samples. Precision and recall values also rose steadily with more training epochs, as shown in Figures 13 and 14. Although the dataset size created limits on overall performance, the results confirmed the capability of YOLO-powered pathway as a strong baseline for the URS system. This is due to its balance of speed, accuracy, and resource efficiency, while pointing to future improvements through expanded datasets and extended training.

```
60 epochs completed in 0.137 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.4MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 2/2
                 all         14         91      0.262      0.513      0.242     0.0678
                Road         14         47      0.167      0.596      0.175     0.0361
            Sidewalk         14         44      0.358      0.431       0.31     0.0994
Results saved to runs/train/exp
```

Figure 11. Training Results of YOLOv5s for 60 Epochs

```
60 epochs completed in 1.040 hours.
Optimizer stripped from runs/train/exp2/weights/last.pt, 173.1MB
Optimizer stripped from runs/train/exp2/weights/best.pt, 173.1MB

Validating runs/train/exp2/weights/best.pt...
Fusing layers...
Model summary: 444 layers, 86180143 parameters, 0 gradients, 204.0 GFLOPs
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 2/2
                 all         14         91      0.339        0.5      0.271     0.0682
                Road         14         47      0.346      0.681       0.32     0.0717
            Sidewalk         14         44      0.331      0.318      0.221     0.0646
Results saved to runs/train/exp2
```

Figure 12. Training Results of YOLOv5x for 60 Epochs

```
212 epochs completed in 0.294 hours.
Optimizer stripped from runs/train/exp4/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp4/weights/best.pt, 14.4MB

Validating runs/train/exp4/weights/best.pt...
Fusing layers...
Model summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%
                 all         14         91       0.47      0.568      0.432     0.0935
                Road         14         47       0.38      0.681      0.462     0.0834
            Sidewalk         14         44       0.56      0.455      0.401      0.104
```

Figure 13. Training Results of YOLOv5x for 200+ Epochs.

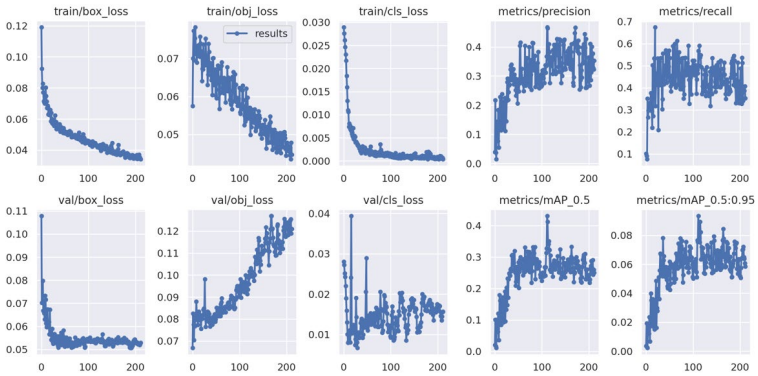

Figure 14. Mean Average Precision, Precision, and Recall Metrics after Training for 200+ Epochs.

6.2 YOLO Detection Results

The trained YOLOv5s model was validated on real-world test images, where it successfully identified roads and sidewalks with confidence scores up to 85% and 79%, respectively, as shown in Figure 15. These structured outputs highlight the model's ability to distinguish riding surfaces in urban settings and provide interpretable inputs for the broader URS framework. Although current experiments were limited to two object classes, these results demonstrate that the system can generate reliable and lightweight visual analytics, offering a foundation for future expansion toward comprehensive urban rideshare safety monitoring.

Figure 15. Roads (red) and Sidewalks (orange) Detection through YOLOv5 Algorithm.

7. Concluding Remarks

This work presents the development of an intelligent urban rideshare safety system based on YOLO-powered object detection with requirement analysis and system design. By leveraging smartphone-based data acquisition and deep learning, the system successfully detected roads and sidewalks with confidence scores exceeding 70%, achieving a balance of accuracy and efficiency suitable for lightweight deployment. Comparative analysis of YOLOv5 variants further underscored the trade-offs between accuracy and computational requirements. YOLOv5s provided the most effective balance of accuracy, efficiency, and model size, making it suitable for real-time use. In contrast, YOLOv5x offered marginally higher accuracy but at the cost of significantly longer training times, underscoring the trade-off between precision and computational feasibility.

The study establishes the feasibility of applying deep learning–based object detection to enhance rideshare safety. However, limitations remain. The relatively small dataset restricted the model's ability to generalize across diverse conditions such as varied lighting, weather, and road types. Moreover, the current system only validates visual detection, whereas the broader URS framework envisions multi-sensor integration and intelligent reasoning. For

example, accelerometer-based vibrational data could complement visual analytics to capture road roughness, while reasoning models could combine these multimodal inputs into actionable safety indices.

Future work should therefore expand in two directions. First, expanding the dataset to cover a broader range of urban conditions, including nighttime rides, adverse weather, and varied infrastructure. These will help improve robustness and reduce bias in detection performance. Second, integrating vibrational sensing and reasoning methodologies to make the URS system more comprehensive. By combining detection confidence scores with vibration-based metrics and applying reasoning, the URS system can generate interpretable safety indices and move from simple perception toward intelligent cognitive assistance, bridging the gap between detection outputs and real-world safety guidance.

References

Agin, G. J., Computer vision systems for industrial inspection and assembly. *Computer*, vol. 13, no. 5, pp. 11–20, 1980. https://doi.org/10.1109/mc.1980.1653613

Ashqar, H. I., Almannaa, M. H., Elhenawy, M., Rakha, H. A., and House, L., Smartphone transportation mode recognition using a hierarchical machine learning classifier and pooled features from time and frequency domains. *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 244–252, 2019. https://doi.org/10.1109/tits.2018.2817658

Dubey, A. K., and Jain, V., Comparative study of convolution neural network's ReLU and Leaky-ReLU activation functions. *Lecture Notes in Electrical Engineering*, pp. 873–880, 2019. https://doi.org/10.1007/978-981-13-6772-4_76

Huang, J., Qian, F., Gerber, A., Mao, Z. M., Sen, S., and Spatscheck, O., A close examination of performance and power characteristics of 4G LTE networks. *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pp. 225–238, 2012. https://doi.org/10.1145/2307636.2307658

Jahangiri, A., Rakha, H. A., and Dingus, T. A., Adopting machine learning methods to predict red-light running violations. *2015 IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 650–655, 2015. https://doi.org/10.1109/itsc.2015.112

Laroca, R., Severo, E., Zanlorensi, L. A., Oliveira, L. S., Gonçalves, G. R., Schwartz, W. R., and Menotti, D., A robust real-time automatic license plate recognition based on the YOLO detector. *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10, 2018. https://doi.org/10.1109/ijcnn.2018.8489629

Ma, R., and Niu, L., A survey of sparse-learning methods for deep neural networks. *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pp. 1–8, 2018. https://doi.org/10.1109/wi.2018.00-20

Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., and Omata, H., 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. https://doi.org/10.1111/mice.12387

Patel, D., What is YOLO algorithm? *YouTube*, 2020. https://www.youtube.com/watch?v=ag3DLKsl2vk

Ruder, S., An overview of gradient descent optimization algorithms. *arXiv preprint*, arXiv:1609.04747, 2016. https://arxiv.org/abs/1609.04747

Tsmots, I., Rabyk, V., and Ignatyev, I., Implementation of sigmoid activation functions on FPGA for neural networks. *Electronics and Information Technologies*, vol. 11, pp. 148–154, 2019. https://doi.org/10.30970/eli.11.4

Vandenhende, S., Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., and Van Gool, L., Multi-task learning for dense prediction tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–20, 2021. https://doi.org/10.1109/tpami.2021.3054719

Yachida, M., and Tsuji, S., Industrial computer vision in Japan. *Computer*, vol. 13, no. 5, pp. 50–63, 1980.

Yan, B., Fan, P., Lei, X., Liu, Z., and Yang, F., A real-time apple target detection method for picking robot based on improved YOLOv5. *Remote Sensing*, vol. 13, no. 9, p. 1619, 2021. https://doi.org/10.3390/rs13091619

Yang, J., Fu, X., Hu, Y., Huang, Y., Ding, X., and Paisley, J., PanNet: A deep network architecture for pan-sharpening. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 5449–5457, 2017. https://doi.org/10.1109/ICCV.2017.581

Zhang, L., Yang, F., Daniel Zhang, Y., & Zhu, Y. J. Road crack detection using deep convolutional neural network. *2016 IEEE International Conference on Image Processing (ICIP)*, 2016. https://doi.org/10.1109/icip.2016.7533052

Zhou, F., Jiao, J. R., Chen, S., and Zhang, D., A case-driven ambient intelligence system for elderly in-home assistance applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 2, pp. 179–189, 2011. https://doi.org/10.1109/tsmcc.2010.2052456

Biographies

Wei-Lun Huang currently is a graduate research assistant of the M.S. program in the School of Industrial and System Engineering (ISyE) at Georgia Institute of Technology. He earned his B.S. in Industrial Engineering and Engineering Management from National Tsing Hua University, Taiwan. He has practical experience through internships at Micron and ASML. His research interests include production and operation management, supply chain design, optimization modeling, and machine learning/AI applications. More information is available on his LinkedIn profile: https://www.linkedin.com/in/wlh1020/

Roosan Liyons received his M.S. in Mechanical Engineering from the Georgia Tech, where his research focused on developing intelligent cognitive assistance systems for urban rideshare safety. He also earned his B.S. in Mechanical Engineering from Georgia Tech with Highest Honors. His academic interests include engineering design optimization, finite element analysis, thermal design, and the application of machine vision in advanced mechanical systems. More information is available on his LinkedIn profile: https://www.linkedin.com/in/rliyons/

Roger Jiao is the editor-in-chief of the Journal of Engineering Design and an associate professor of mechanical and industrial engineering at Georgia Tech. Prior to joining the School of Mechanical Engineering at Georgia Tech in December 2008, he was an Assistant Professor and then Associate Professor in the School of Mechanical and Aerospace Engineering at Nanyang Technological University, Singapore. Before his career in Singapore, he was a Visiting Scholar in the Department of Industrial Engineering and Engineering Management at Hong Kong University of Science and Technology from 1998 to 1999. From 1993 to 1994, he was a Lecturer of Industrial Engineering in the School of Management at Tianjin University, China. From 1988 to 1990, he worked as an Associate Lecturer in the Department of Industrial Design at Tianjin University of Science and Technology, China. More info about his research: https://scholar.google.com/citations?user=9yikEHAAAAAJ&hl=en&oi=ao