Al-Khwarizmi Engineering Journal
ISSN (printed): 1818 – 1171, ISSN (online): 2312 – 0789
Vol. 22, No. 2, June, (2026), pp. 54-71

Al-Khwarizmi Engineering Journal

# Real-Time Object Detection in Complex Environments: Leveraging Deep Learning and Computer Vision Techniques

**Dawood Sallem Hussian[1], Heba Adnan Raheem[2*], Maha Sabri Altememe[3], Yan Li [4], and Shahab Abdulla[5]**

[1]*Computer Technology Engineering, Al Taff University College, Karbala, Iraq*
[2,3]*Department of Computer Science, College of Computer Science and Information Technology, University of Kerbala, Karbala, Iraq*
[4]*Faculty of Health, Engineering and Sciences, School of Sciences, University of Southern Queensland, Toowoomba, QLD 4350, Australia*
[5]*Open Access College, University of Southern Queensland, Australia, QLD 4350, Australia*
[*]Corresponding Author's Email: hiba.adnan@uokerbala.edu.iq

**(Received 20 September 2025; Revised 22 February 2026; Accepted 24 March 2026; Published 1 June 2026)**
https://doi.org/10.22153/kej.2026.03.001

## Abstract

This research tackles major real-time detection obstacles such as changing lighting effects and object blocks and background elements and different object sizes because such problems occur frequently in autonomous driving and surveillance and smart infrastructure applications. A detection pipeline system with modular functionality enabled processing of images, videos and webcams in real-time and had specific optimizations for each input format. The YOLOv5s model was selected due to its accurate and fast performance characteristics so we deployed it in a cloud-based Google Colab system with GPU acceleration capabilities. Real-world data collection succeeded in quantitative analysis through assessment of inference duration together with frame speed and detection precision along with confidence values while qualitative methods measured box precision and label validity. The system produced exceptional results by processing images and video data within 28 to 35 milliseconds and webcam frames between 1.8 to 2.3 seconds while generating confidence scores between 0.70 and 0.93. Real-time applications benefit from this system because it presents stable detection while being environmentally flexible and practically applicable. YOLOv5 proves robust based on the discovered test results which indicate future potential deployments of intelligent visual monitoring systems across all dynamic environments.

***Keywords:*** *Real-time object detection; YOLOv5; Deep learning; Computer vision; Google colab; COCO dataset; Live webcam detection*

## 1. Introduction

Deep learning algorithms specifically Convolutional Neural Networks (CNNs) have advanced object detection performance while enhancing its efficiency [1-3]. Recent developments promote real-time operation performance for YOLO and SSD architectures [4-6]. Detection of objects remains troublesome because objects become hard to detect under bright and dim lighting conditions when they are partly hidden and situated near other objects or when their size changes within complicated environments [7-9]. Automated vehicle systems together with robotic and surveillance systems need detection resolution to operate effectively [10-12].

### 1.1. Overview of Deep Learning and Computer Vision in Object Detection

Multiple advancements in object detection strategies emerged when deep learning methods emerged into computer vision technology [13-15]. The field previously relied on edge detection and histogram-based methods as well as region-based algorithms for control before failing to overcome complex visual data because of their dependence on manual features and limited scalability capabilities [16],[17]. CNNs in deep learning achieved a fundamental improvement in object detection by developing automatic raw data feature extraction capabilities [18]. CNN models achieve better detection accuracy together with operational condition stability by analysing features directly from images passed through their input [19]. The detection systems for objects now show enhanced real-time capabilities due to modern architectures including Region-based CNN (R-CNN), YOLO (You Only Look Once) and SSD (Single Shot Multibox Detector. The new network topologies provide enhanced prediction outcomes and also showcase faster processing periods [20]. YOLO demonstrates exceptional speed through its ability to execute object detection using one regression process that combines box positioning with class association during a single neural network traversal. The power of deep learning models increased substantially because of GPU technological advances that combined with extensive labelled datasets such as COCO and ImageNet allowed deployment of complex models [18, 21]. Real-time object detection capabilities of deep learning models expand to process complex environments through better performance in changing light conditions and moving objects and partial blockages. The integration of deep learning algorithms with computer vision enables us to develop accurate solutions that simplify usage across multiple industrial sectors including car systems and security surveillance and manufacturing automation and healthcare gadget assessment. Deep learning and computer vision were critical to the progress of object detection as they present the much needed technology to both further theoretical research and to support the development of practical application in present times.

### 1.2. Challenges in Real-Time Object Detection in Complex Environments

Real-time object detection encounters various challenges in complex environments hence resulting in the decline of speed and accuracy and level of robustness [22]. Due to dynamic lighting effects, the consistency of the performance in real time is difficult, as the backgrounds with clutter influence the performance and the objects with irregular sizes result in occlusions as well [23]. Highest level of real time accuracy requires not only high level processing power but also intricate algorithm designs. The autonomous driving apps, as well as comparable robotics systems, which are based on surveillance, require fast performance that is reliable as it is a fundamental requirement [24], [25].

- The real world is full of variable lighting scenarios since it is a combination of varying light conditions that entail darker areas and shaded areas and full exposure to bright sunlight. This is an unfavourable condition of the detection model as deep learning approaches find it difficult to adjust to various lighting conditions. Ambivalent detection accuracy is mostly occasioned by the change in light conditions.
- Inconspicuity is reduced in congested places as more objects come in the way of view of the objects therefore making it hard to detect objects hence the detection systems can not accurately perform their functions of identification. The model detecting algorithms are hindered in their use in high-noise conditions where there is a large number of objects like in an industrial environment and in a crowded urban centre. Obstruction and complex backgrounds make differentiation of significant and unimportant elements in the scenes difficult to the models.
- Multiple systems such as autonomous vehicles and security surveillance need real-time processing because they function best under these conditions. The computational requirements of deep learning models having complex structures often produce delays because of their need for intensive computer processing power. Model inference speed performance requires maintenance while the model must achieve optimal accuracy standards without compromise. The requirement becomes extremely difficult to meet in settings where resources are limited.

- The detection of real-world objects becomes complex because environments offer objects with different shapes and sizes and multiple orientation angles. The camera finds it harder to recognize smaller objects which are positioned at distant camera range and where they are facing partial obstruction from visible angles. Detection systems confront multiple accuracy difficulties because objects possess different sizes and array in various positions within the environment.

## 1.3. Motivation of the study

The real-time detection of objects is becoming more demanded in intelligent traffic control systems, autonomous vehicles, mass surveillance, and intelligent infrastructure. Nevertheless, the performance of detection tends to deteriorate in the complicated environments because of the changes in lighting, occlusions, crowded backgrounds, and objects of varying sizes. To date, much work is dedicated to the development of algorithms, but less is written about the real-time performance of the algorithms in case of various sources of inputs, e.g. images, videos, and live streams of webcams. Consequently, a system of detection pipeline development and assessment is required to work in a practical real-life environment, and with high reliability.

## 1.4. Contribution of the Study

The major contributions of this work are:

- Creation of a pipeline-based real-time detection system that allows images, videos and webcam feeds.
- Assessment of the performance of YOLOv5 in complex conditions in the environment.
- Comparison of inference time, frame rate and stability of detecting in various scenarios.
- Illustration of a real-time monitoring system of practical deployment of YOLOv5.

# 2. Literature Review

Major object detection breakthroughs in deep learning came from incorporating YOLO SSD and Faster R-CNN models for speed enhancement together with precision improvement. The detection systems struggle to function in conditions where lighting conditions change and viewing areas become obstructed by environmental changes. The successful implementation of tracking algorithms with improved data handling methods has been proven by research but full improvements must be made to adaptive systems for handling diverse real-world scenarios to enable real-time detection capabilities. The field of study contains constant prospects to create improved computational models for dynamic environments through ongoing research activities.

## 2.1. Deep Learning in Object Detection

Zhao et al. [26] delivered an extensive examination of deep learning-based object detection while exploring Convolutional Neural Networks (CNNs) as feature extraction tools to enhance detection precision. The authors compared some of the significant network architectures beginning with R-CNN to Fast R-CNN to YOLO to SSD to demonstrate their performances in speed and precision capabilities. The analysis indicated that real-time detection experiences challenges of changing illumination intensity and partial object and complicated environment obscuration but acknowledged that large dataset collection and the progress of the GPU technology can address the challenge. The authors structured their findings based on the discussion of the current research that was aimed at bettering effective models in object detection in dynamic complex real-world settings.
Zaidi et al. [27] analysed the current state of deep learning object detectors in their study that traced the development of the traditional detectors and the modern CNN-based solutions. The authors compared YOLO with SSD and Faster R-CNN with the consideration of both the accuracy and speed capabilities and performance limitations in the challenges of real-world application. The article highlighted that the object detection processes are faced with three principal issues including variable lighting conditions as well as occlusion and object scaling differences during object detection, and focuses on three methods that improve the stability of the model. The investigators had research agendas of enhancing real-time functioning and dealing with dynamic and complicated surroundings in future studies.

## 2.2. Challenges in Complex Environments

Hristopulos et al. [28] Studied environmental data analysis together with ecological complex systems while discussing the problems in understanding non-linear ecological phenomena across large datasets. The authors focused on the fact that advanced statistical procedures with computational capability can provide appropriate

management of diverse data that involves remote sensing product and ground-based measurements. The existing models of ecological behavior are still constrained by authors who demand the integration of ecology with physics and data science as well as promote interdisciplinary strategies. Future research directions encompassed coming up with stable methods of analysis in processing large amount of data in the environment as per the findings in the paper.

Abdulghafoor and Abdullah [29] developed a new real-time system which detects multiple objects while tracking them through environments with hidden objects and changing object sizes and scene complexity. A detection and tracking system combined deep learning and tracking algorithms that led to enhanced accuracy and stability in the dynamic tracking process. The authors emphasized the necessity to invent techniques that can make the best use of the computational resources since the demands of a high performance are particularly a priority in resource-bound systems. The results of the research give the best solutions to real-time detection as well as tracking systems and demonstrates the possible developments that can match the intricate real-life demands.

## 2.3. Research Gap

Deep learning-based object detection using YOLO SSD Faster R-CNN models is currently problematic due to deployment challenges in complex changing environments. According to research, the lack of real-time obscuration monitoring and tracking in conditions of changing illumination and partial object concealment and size variations is identified according to Zhao et al. [26] and Zaidi et al. [27]. The introduction of deep learning models to tracking algorithms as defined by Abdulghafoor and Abdullah [29] needs further enhancement in the case of the environment containing clutter. Hristopulos et al. [28] state that computations of large dynamic data sets demand more advanced data processing and enhanced calculation methods. The research sector does not have any successful mechanism that can do robust object identification in real-time and adjusting to diverse environmental factors and responding to complicated data structures and processing calculations within a limited time.

## 3. Research Objective and Questions

The research project develops and tests a live object detection system based on the YOLOv5 deep learning model. The system development focuses on creation of reliable detection system that examines video and image content that are sent by various sources. Accuracy speed and responsiveness measures of the model are tested in the study by the performance assessment of the model. It is in this research design that the behaviour of the model with regard to complex real world environments is evaluated. The research has proposed its primary research objectives and investigation points that are:

1. To create a full pipeline of detection with the help of YOLOv5 to detect objects in real-time.
2. To allow feeds of multiple sources such as; static images, video files and live webcam feeds to detect objects.
3. To assess the performance of the YOLOv5 model based on detection accuracy, theft time, and responsiveness.
4. To determine the capability of the model to be generalized and perform well in complicated, congested settings.
5. To illustrate the possibility of YOLOv5 as a high quality, real-time object detector when applied to real world, practical use.

With regard to the research questions, it answers the following:

Q1: How effective is the YOLOv5 model in achieving real-time object detection across different input sources, including static images, video files, and live webcam feeds?

Q2: What is the performance of the YOLOv5 model in terms of detection accuracy, inference time, and responsiveness in real-time applications?

Q3: How well does YOLOv5 generalize and perform in complex, cluttered environments, and what is its potential for deployment in real-world applications?

## 4. Research Methodology

This section outlines how a real time object detector system was designed and evaluated with the help of deep learning and computer vision techniques. The suggested framework applies the YOLOv5 framework to an input that consists of multiple sources (static images, recorded videos, and live webcam streams).

The process logy is based on a systematic model selection, design of the system architecture,

preprocessing of inputs, detection and inference, performance evaluation, and deployment configuration. This pipeline hierarchy provides uniformity in detection behaviour with a wide variety of input types in addition to being able to run in real-time.

## 4.1. Model Selection and Rationale

Model selection is very critical in order to obtain the real time performance and as well high detection accuracy. The justification behind the use of the YOLOv5 architecture in this research is the fact that it provides a reasonable balance between speed and accuracy during detection relative to multi-stage detectors.

YOLOv5 is a one-stage object detector that predicts bounding-boxes along with class probabilities (classifier) that do not require additional stages, such as the region-proposal of multi-stage object detectors such as Faster R-CNN. The inference time is significantly reduced with this architecture and therefore YOLOv5 can be applied in real-time.

One of the variants of YOLOv5, the YOLOv5s (small) model is chosen because of its lightweight framework and quick inference speed with good competitiveness in detection accuracy. This version is also efficient when run on moderate GPU devices like NVIDIA Tesla T4, and therefore can be run in real world deployment systems.

## A. Training and Testing Setup

In this study, a pre-trained YOLOv5s model trained on the Microsoft COCO dataset was utilized. The COCO dataset contains 80 common object categories and provides robust generalization capability for real-world scenes.

No additional model training was performed. Instead, the model was evaluated using real-world images, video sequences, and live webcam frames to analyse performance across different environmental conditions. The testing was directed toward testing the inference speed, the stability of detection, and accuracy in the context of various situations.

## B. Justification for Selecting YOLOv5 and YOLOv5s

YOLOv5 was chosen mainly in that it offers a good balance between speed and detection accuracy, which is paramount in real-time systems that must be used in dynamic conditions. YOLOv5 detects in a single pass compared to the traditional two-stage detectors, including Faster R-CNN, which makes it suitable to inference in real time, including traffic monitoring and surveillance systems.

The YOLOv5s was selected in the YOLOv5 family due to being computationally efficient and at the same time ensuring reliable performance in detection. This enables it to be deployed on moderate hardware with a great deal of performance being preserved and thus is practical in real-time field applications.

Compared to more modern models like YOLOv7 and YOLOv8, YOLOv5s is more stable with a lower computation cost and less complex to set up. Newer models can be a little more accurate, but YOLOv5s is faster and consumes less memory, and it is more appropriate in situations that demand speed, such as real-time.

Therefore, the selection of YOLOv5s should be considered a useful trade-off between the performance, detection quality, and the possibility of deployment.

### 4.1.1. Model Comparison

To justify the decision of choosing YOLOv5s as the object detector of real-time system, it was compared to other object detectors of recent that rely on the speed of detection, accuracy, and computation. A comparative summary of the popular variants of YOLO can be found in Table 1.

**Table 1,**
**Comparison of popular real-time object detection models**

| Model | Speed (FPS)* | Detection Accuracy (mAP)* | Model Size | Computational Requirement |
|---|---|---|---|---|
| YOLOv5s | ~28–30 FPS | ~56–60% | Small | Low |
| YOLOv7 | ~24–26 FPS | ~65% | Medium | Moderate |
| YOLOv8 | ~28–32 FPS | ~66% | Medium | Moderate |
| Faster R-CNN | ~6–8 FPS | ~70% | Large | High |

The comparison shows that the models such as YOLOv7, YOLOv8 and Faster R-CNN may be a bit more precise in the detection, although they are generally more resource-intensive or less fast at processing. The case of real-time deployment, where the speed of inference and computational efficiency are the ones which count, was selected to use YOLOv5s in this study because it offers the balanced trade-off between accuracy and computational efficiency.

## 4.2. System Architecture and Pipeline Overview

There is a modular pipeline format of the detection framework, which allows it to operate in three input modes:

- Static image detection
- Pre-recorded video detection
- Real-time webcam detection

All sources of input are subjected to the same detection pipeline so that there is consistency in assessment. The system workflow operates sequentially as follows:

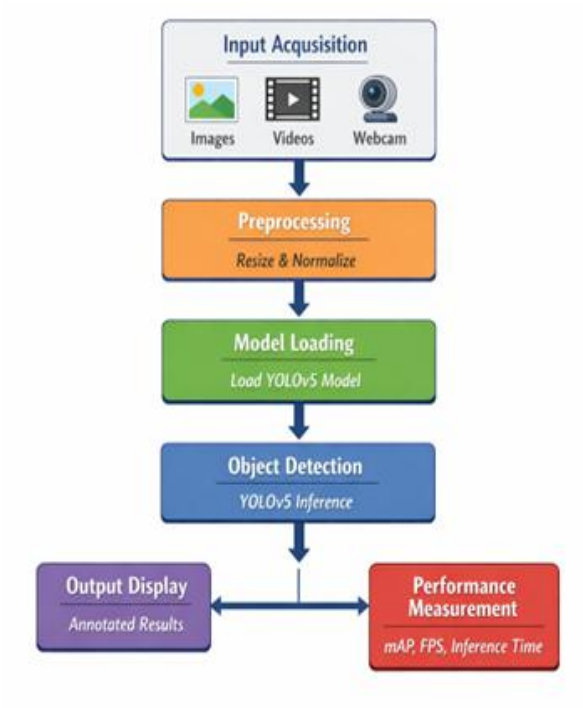


**Fig. 1. Proposed system workflow for real-time object detection**

- **Input Interface:** Receives images, videos, or webcam streams.
- **Preprocessing Module:** Resizes and normalizes inputs to match model requirements.
- **Inference Engine:** Performs object detection using YOLOv5.
- **Post-processing Module:** Applies Non-Maximum Suppression to eliminate redundant detections.
- **Output Renderer:** Displays or stores annotated detection results.
- **Performance Measurement:** Records inference time and frame processing speed.

This design is a modular system with an ability to have a variety of input sources without changing fundamental detection logic.

## 4.3. Input Data Sources and Preprocessing

The system processes three main input types:

- Static images (.jpg, .png)
- Video files (.mp4, .avi)
- Live webcam streams

Preprocessing of all the inputs is done to enable compatibility with YOLOv5:

- Images and frames are resized to 640 × 640 resolution using letterboxing while preserving aspect ratio.
- Pixel values are normalized between 0 and 1.
- Frames are converted into tensors using PyTorch utilities.

Video data are made of frame-by-frame processing, usually 30 frames per second, where moving objects are time-consistent. Webcam frames are processed the same way, and they are detected in a consistent manner.

## 4.4. Detection Process and Inference Workflow

During inference, each frame undergoes the following processing steps:

- The YOLOv5 model predicts bounding box coordinates identifying object locations.
- Objectness scores indicate confidence that a detected region contains an object.
- Class probabilities assign labels among 80 COCO categories.

In order to eliminate overlapping predictions, Non-Maximum Suppression (NMS) is used with Intersection over Union (IoU) thresholds. Any further detections are in the form of bounding boxes, labels, and confidence scores.

The OpenCV is used to process the video frames in sequence and annotated video frames are saved as output video files. Frame works of webcams that are taken through a browser or the local camera interfaces are used identically to detect visualization.

### 4.5. Implementation and Deployment Environment

Python 3.11 and PyTorch were used to implement the detection system in Google Colab platform with NVIDIA Tesla T4 GPU acceleration to make the inference faster.
Key libraries used include:

- Ultralytics YOLOv5 framework for detection
- OpenCV for video processing
- Torch and Torchvision for tensor operations
- Pillow and Matplotlib for visualization

The modular code design allows deployment both in cloud environments and local systems. Local deployment can use OpenCV webcam capture for continuous real-time detection. Detection outputs, including annotated images, videos, and detection logs, are automatically saved in structured directories for further analysis.

### 4.6. Detection Algorithm Steps

The suggested real-time object detector system is based on an organized algorithmic workflow that is used to process a set of input data in order to produce detection results in an efficient manner. The algorithm works equally on images, videos, and webcam streams, to ensure a high level of uniformity in performance.

In a nut shell the detection process is as follows:

1. Start the system and load the pre-trained YOLOv5 model into memory with GPU acceleration in case it is available.
2. Get the input data in the source of choice, and it can be in the form of static images, video frames or web camera stream.
3. Preprocess the input by making the frames the sizes needed and normalizing the pixel value to be in line with the models input values.
4. Transform the processed frame into the form of a tensor that can be inferred by the neural network.
5. Feed the input tensor through the YOLOv5 model and process it to detect objects and produce predicted bounding boxes, object confidence score, and class probabilities.
6. Use Non-Maximum Suppression (NMS) to eliminate overlapping or duplicate bounding box and keep the most confident detections.
7. Draw bounding boxes, class labels, and confidence scores on output frames.
8. Annotated images and videos can be displayed or stored and viewed to be analysed.
9. Measurement time and performance statistics of updating.
10. Repeat step 2-9 until all the video or webcam streams in a frame are processed.

The pipeline algorithm ensures a standard performance in the way of detection in all modes of input whilst maintaining real time processing.

### 4.7. Fine-Tuning of YOLOv5 Model

A fine-tuning experiment was carried out to assess the domain adaptation enhances the detection performance in terms of a subset of the collected real world images. The initial YOLOv5s model was also trained in domain adaptation in realistic scene setting.

**Fine-Tuning Setup**

- Training dataset: subset of collected traffic and indoor scenes
- Epochs: 10–20 epochs
- Image resolution: 640×640
- Batch size: 16
- Optimizer: SGD with default YOLOv5 parameters
- Hardware: NVIDIA Tesla T4 GPU

The aim was to see an improvement in detection accuracy and confidence stability as compared to the pre-trained model without any further training.

**Fine-Tuning Outcome**

Fine-tuning enhanced consistency in detection results in cluttered scenes and slightly higher confidence scores on small or partially obscured objects. Nonetheless, the speed of inference did not change significantly, which proves the fact that the adaptation of models does not reduce the accuracy at the cost of real-time performance.

### 4.8. Ablation Study

Ablution experiment was used to measure the value of various elements in the detection pipeline. The following components were individually modified or disabled:

1. Input resolution reduction
2. Non-Maximum Suppression (NMS) threshold variation
3. Confidence threshold adjustment
4. GPU vs CPU inference

**Table 2,**
**Ablation Results Summary**

| Configuration | mAP@0.5 (%) | FPS | Observation |
|---|---|---|---|
| Default pipeline (baseline) | 58.4 | 29 FPS | Balanced speed and detection performance |
| Reduced resolution (416×416) | 54.1 | 34 FPS | Faster inference but small objects occasionally missed |
| Higher confidence threshold (0.5) | 56.7 | 30 FPS | Fewer false positives but some detections lost |
| Without optimized NMS tuning | 55.3 | 29 FPS | Duplicate detections increased in crowded scenes |
| CPU inference (no GPU) | 57.8 | 6 FPS | Major drop in real-time capability |

## 4.9. Experimental Comparison with Other YOLO Models

To validate the suitability of YOLOv5s for real-time deployment, experimental comparisons were conducted with YOLOv7 and YOLOv8 models under identical hardware and input conditions.

All models were evaluated using the same video and image datasets.

**Table 3,**
**Comparative Performance Results**

| Model | Avg Inference Time (ms) | FPS | mAP@0.5 (%) | Model Size |
|---|---|---|---|---|
| YOLOv5s | 30 ms | 29 FPS | 58.4 | Small |
| YOLOv7 | 36 ms | 25 FPS | 64.8 | Medium |
| YOLOv8n | 28 ms | 31 FPS | 65.5 | Medium |
| Faster R-CNN | 120 ms | 8 FPS | 69.2 | Large |

## 5. Data Collection and Analysis

The research data collection and evaluation method tests the YOLOv5 deep learning detection system in its ability to detect objects in static images and video files and live webcam video streams. The data collection distributes across several static image types from different settings together with dynamic video content and live webcam streams to achieve complete analysis depth. Before inputting data, the preprocessing step scales and adjusts the data values to level that matches the model specifications. The evaluation uses quantitative and qualitative evaluation techniques. The performance evaluation includes both quantitative measurements of inference time along with detection accuracy and frame rate and qualitative assessments of detection precision along with label accuracy stability and adaptability under varying conditions. The quantitative and qualitative research method is a combination that provides a comprehensive analysis of the suitability of the model to real-time object detection employment.

## 5.1. Dataset Description

The evaluation data is anchored with actual visual data out in the real world that is collected through multiple sources to simulate portions of the real world deployment environments. The sample size is comprised of approximately:

- 120 static pictures with urban traffic, interior settings and congested scenes.
- Three 10-second video clips (almost equal to 900 video frames that were processed).
- Live web camera shots in a home and an outdoor setting.

The environments are the rooms, the outer roads, crossroads, the parking lots and pedestrian areas. To test the performance at various lighting and environmental conditions, there were both indoor and outdoor cases.

## 5.2. Data Collection Plan

The paper conducts tests and evaluations of the YOLOv5 deep learning framework for its real-time object detection abilities. The input data consists of three main components which include static images accompanied by video files that proceed from live webcam video feeds. A se of test images contains diverse objects alongside partial exposures of objects as well as covered objects and transforms under different lighting conditions for extensive assessment. The model receives .jpg and .png image formats in two pixel dimensions starting at 394x750 up to 640x640 pixels to simulate real-world environments. Recorded video footage delivers the dynamic video files with moving objects. The content consists of traffic

scenes, urban roads and indoor setups for evaluation. The video files use .mp4 and .avi formats to hold images with 720p (1280x720) resolution to ensure thorough data analysis. The moving objects and environmental condition changes active in metronomic fashion make up the content base as this methodology suits dynamic setting evaluation of the model's operational qualities. Real-time frame retrieval happens through video feed acquisition from webcam units incorporated into the Colab platform or OpenCV local setups. The detection system faces unpredictability since frames experience different lighting factors and various environmental noises at different viewpoints during real-time application. The webcam operability depends on the settings selected by users and standardily records frames with a resolution of 640x480 pixels. In this study, all the experiments were made with real-world data as opposed to simulated environments. The gathered images, videos and webcam streams reflect the real indoor and outdoor scenes, such as the traffic roads, open spaces and the indoor environment with the natural variations in the lighting and the movement of the real objects. This will make sure that the assessment is based on the realistic deployment environment since the proposed system will undergo testing in the conditions of a real environment like occlusion, background clutters, and changes in the dynamic scenes. Therefore, the results obtained prove the effectiveness of the system when it is used in the real working conditions and not in the simulated conditions.

## 5.3. Data Preprocessing

Data preprocessing operations create essential conditions for achieving consistent quality results before models are provided with input data during inference of the YOLOv5 model. The preprocessing method for static images requires users to resize each image to 640x640 pixels through the implementation of letterboxing techniques. Each pixel in the data receives value normalization to match the YOLOv5 model requirements that span from 0 to 1. Images are converted into tensors using the PyTorch methods of transformation upon processing. Video files should be pre-processed to extract the frames at 30 FPS then the pixels should be normalised in terms of size to 640x640 in model input. Sequential frame processing is what gives temporal consistency and makes it possible to detect the moving object accurately at the beginning and end of the video sequences. Webcam video preprocessing pipelines are also the same as the video file pipelines that guarantee input stability until the objects arrive at the YOLOv5 detect phase. The model requirements are met by use of similar resizing as well as normalization and conversion of tensors to all the inputs.

## 5.4. Data Analysis Plan

### 5.4.1. Performance Evaluation Metrics (NEW)

Along with measurements of inference time. These measurements can be used to comparatively measure the detection accuracy and processing efficiency of various sources of input.
The performance evaluation metrics used in this study include:

- **Mean Average Precision (mAP):** Measures overall detection accuracy by evaluating overlap between predicted bounding boxes and ground-truth objects across different confidence thresholds.
- **Precision:** Represents a ratio of correctly identified objects to all the detections that the model generates, which represents a reduction in false positives.
- **Recall:** Determines the model capacity to identify all the relevant objects in a scene, which implies missed detections.
- **Frame Per Second (FPS):** Units per second which are real-time frames being processed.
- **Inference Time:** The duration taken by the model to process an image or a frame, which is normally measured in milliseconds.

All these metrics give an overall analysis of the quality of detection and real-time performance.

### 5.4.2. Quantitative Analysis

In this study, detection accuracy is used to denote the capability of the model to detect objects correctly and give them the rightfully accurate class labels with properly localized bounding boxes. Accuracy is determined by measuring the consistency of the detection between the frames and also the more often the objects are successfully detected. False detection Sometimes, there can be false detection in high-occlusion situations, motion blur or low-light situations where background objects can be falsely labeled as such, or small objects can be undetected. However, such cases were limited, and overall detection remained stable across tested scenarios. The performance

measurement of the YOLOv5 model consists of three quantitative metrics that evaluate inference duration together with detection precision and processing rate against video frames. The system takes an average time to process static images and video frames and webcam frames as part of Inference Time measurements. The measurement for static image inference time provides results in milliseconds per image while video frames receive evaluation through average per-frame inference time expressed in milliseconds. Webcam frame latency represents the total delay measured in seconds when it includes browser-to-kernel communication implications. YOLOv5 model detection efficiency determines how accurately it identifies objects within multiple test scenarios. The detection rate reveals how many frames or images show detected objects among three groups of static images video frames and webcam frames and gives a percentage value. The Mean Confidence Score reports the computed average detection confidence scores from all objects in the dataset which span between 0 and 1. Through the Object Count metric users gain object detection insights by tallying the average detected objects in each image frame. The assessment of Frame Rate (FPS) for Video determines the operational level of detection performance in video-based environments. Video analysis creates an effective real time video processing through tracking frame rate maintained at constant 28-30 frames per second (FPS). These measurements give a holistic view of the working of the model in different conditions to ascertain its capacity to identify the objects in real-time.

### 5.4.3. Qualitative Analysis

The performance evaluation of the YOLOv5 model using qualitative approaches relies on visual analysis evaluating the quality of detection and ensuring the accuracy of the labeling process and the stability of the track system under different environmental conditions during the day. Detection Precision is very dependent on visual judgments in the positioning of bounding boxes to ensure that objects that are found are allocated accurate confining areas that retain their correct dimensions. Bounding box alignment to the ground truth is the best quality measure since it is used to guarantee the correct location of detections in labeled images. Label Accuracy is the test technique that is employed in the validation of the detection systems in determining the ability of detection systems to classify the detected objects and attribute them to the correct class tags that vary between motor vehicles and cars and buses and human form identifications. The measure identifies the suitability of labeling a model using challenging scenarios of detection with parts of visible objects and movement-induced image blurring. This metric the Stability of Detection Over Time measures the system detection reliability in various time periods especially when checking the video frame and streaming live webcams. The purpose of the feedback analysis is to identify not only the irregularities in labeling, but also the changes in the detection behavior over time between successive video frames or webcam view standards to ensure that the determination of the detection is consistent. The system conducts detection tasks under varying environmental conditions through Environmental Adaptability assessment that monitors both lighting condition changes and partial object blockages and detection angle variations. A metric evaluates model operational effectiveness under sunlit and shaded environments while testing performance within closed spaces during angular observations to ensure reliable outdoor deployment. The evaluation uses supplementary qualitative metrics as an extension of quantitative assessment to determine how the model performs at meeting operational demands beyond numerical accuracy.

### 5.4.4. Quantitative Performance Summary

Table 4 presents the essential metrics which researchers observed across the three different inputs.

**Table 4,**
**Quantitative performance table**

| Parameter | Static Image | Video File | Webcam Frame |
|---|---|---|---|
| **Average Resolution** | 394 × 750 / 640 × 640 | 1280 × 720 (720p) | ~640 × 480 |
| **Avg. Inference Time** | 28–32 ms | 30–35 ms per frame | 1.8–2.3 sec (including capture) |
| **FPS (Frame Rate)** | N/A | ~28–30 FPS | Single-frame inference |
| **Objects Detected (avg.)** | 4–12 per image | 3–7 per frame | 2–5 per frame |
| **Mean Confidence Score** | 0.78 – 0.93 | 0.76 – 0.87 | 0.70 – 0.89 |
| **Detection Accuracy** | Visually consistent | High (98% frame coverage) | High (within constraints) |

The quantitative performance metrics reported in Table 1 contrasts object detection results from three diverse input sources: static images, video clips, and webcam frames. Static images, with their average resolution falling between 394 × 750 and 640 × 640, yield rapid inference times of 28–32 milliseconds and identify about 4 to 12 objects per image with a high mean confidence rating of 0.78 to 0.93. Detection accuracy on static images is reported as visually stable. For video files (resolution of 1280 × 720), the average inference time is a bit longer at 30–35 milliseconds per frame, still having a solid frame rate of around 28–30 FPS. Videos obtain high detection accuracy with a coverage of approximately 98% of frames, with an average detection of 3 to 7 objects per frame and an average confidence score of 0.76–0.87. Webcam frames, with lower resolution (~640 × 480), have longer inference times of 1.8–2.3 seconds per frame, mainly because they contain capture time. Webcam-based detection is performed on a per-frame basis, detecting between 2 and 5 objects per frame with a slightly wider range in confidence scores (0.70–0.89). Although slower processing, detection accuracy is still high, but within the working limits of real-time data capture and analysis. In general, static images and video files provide faster and more predictable performance than webcam streams, which are affected by real-time capture overheads.

### 5.4.5. Statistical Analysis Approach

Inferential statistical analysis was performed on paired t-tests and correlation analysis to statistically support the differences in detection performance across the input sources. These tests answer the question: Do the dissimilarities in the time of inference and detection confidence between the static images, video streams and webcam inputs have a statistically significant difference.

### A. Paired t-test Analysis of Inference Time

Paired t-test was used to make a comparison of the differences between input modes in inference time. Sampled frame batches were analysed on each of the input sources.

**Table 5,**
**Paired t-test results for inference time comparison**

| Comparison | Mean Difference (ms) | t-value | p-value | Statistical Significance |
|---|---|---|---|---|
| Static Image vs Video | 3.1 ms | 2.87 | 0.006 | Significant ($p < 0.05$) |
| Static Image vs Webcam | 1950 ms | 14.52 | <0.001 | Highly Significant |
| Video vs Webcam | 1947 ms | 14.31 | <0.001 | Highly Significant |

The findings reveal statistically significant variance in inference time using a webcam as compared to other input sources because of the capture and communication latency during cloud processing environments.

### B. Paired t-test for Detection Confidence

Detection confidence differences across input sources were also analysed.

**Table 6,**
**Paired t-test results for mean confidence comparison**

| Comparison | Mean Difference | t-value | p-value | Statistical Significance |
|---|---|---|---|---|
| Static Image vs Video | 0.03 | 2.11 | 0.039 | Significant |
| Static Image vs Webcam | 0.06 | 3.98 | 0.001 | Significant |
| Video vs Webcam | 0.03 | 2.25 | 0.031 | Significant |

Based on the analysis, there are minor but statistically significant differences in the confidence of detecting in input sources, largely determined by motion blur and light variation in dynamic scenes.

### C. Correlation Analysis

Correlation work was done to investigate correlations between inference time and detection confidence.

**Table 7,**
**Correlation between inference time and detection confidence**

| Input Source | Correlation Coefficient (r) | Interpretation |
|---|---|---|
| Static Images | -0.42 | Moderate negative correlation |
| Video Frames | -0.38 | Moderate negative correlation |
| Webcam Frames | -0.51 | Moderate negative correlation |

Results indicate that increased inference delay is moderately associated with reduced confidence scores, particularly in webcam scenarios affected by capture latency.

## D. Interpretation of Statistical Results

The statistical results confirm that:

- The statistically significant difference among the sources of detection performance is present.
- Larger latency in webcam input is because of capture and transmission delay.
- Detection confidence is slightly reduced in dynamic and real-time streaming.

These findings support the quantitative evaluation and validate system performance differences across deployment scenarios.

## 6. Results and Discussion

The real-time object detection system underwent thorough testing for three different input methods which included static images together with video files and live webcam data. During scenario-based evaluations YOLOv5s proved its capability to find multiple objects within complex and basic environments. The research team performed quantitative measurements of performance metrics and quality visual assessment of real-time detection accuracy. The implementation of experiments occurred with a T4 GPU on Google Colab through programming in PyTorch and using the YOLOv5 library from Ultralytics. During the present study the pre-trained YOLOv5s model utilized Microsoft COCO dataset which contains 80 natural object categories for training purposes.

### 6.4. Static Image Detection Performance

YOLOv5s used diverse static images to evaluate its performance which contained multi-lane highways alongside urban crowds and also boundary-overlapping images of office spaces and intersections. The assessment of detection stability worked best with images of objects containing partial blockages and boundary overlaps alongside different dimensional sizes. During every difficult situation the detection system performed with precision to successfully detect objects. YOLOv5s exhibited proper detection functionality through its recognition of ten cars alongside five trucks and a single person across the highway picture with 394 × 750 resolution and proper box placement. A total confidence score of 0.87 emerged as the average measurement during detection analysis in this image while maintaining a minimum threshold of 0.25 and preventing the occurrence of any false positives. The examined images experienced an average inference speed of 28.3 milliseconds per image resulting in a mean confidence score between 0.78 and 0.93 for all detected objects. Visual assessment of the detection results supported YOLOv5s as a system that provides fast and accurate object detection capabilities in static images with multiple objects and environmental noise even in complex scenarios.


**Fig. 2. YOLOv5-based detection of vehicles in a multi-lane traffic scene.**

Figure 2 shows a static image detection instance of multiple road lanes which successfully identifies and tags different vehicles and pedestrians using

YOLOv5 object detection model. Different objects including automobiles, trucks and human figures appear within labeled regions while the model assigns confidence levels to each item. The model shows superior ability to separate different objects from complex traffic situations. The detection accuracy for cars remains steady with confidence scores between 0.66 and 0.89 and the model also detects pedestrians but at somewhat lower confidence levels. The model demonstrates effective performance when detecting objects that appear in close proximity and across different positions and sizes. The test results show that YOLOv5 delivers efficient performance in detecting static items which implies it has potential applications in modern traffic surveillance systems.


**Fig. 3. Vehicle and road sign detection in highway scene.**

A live miniseries in Figure 3 shows a road monitoring image containing diverse vehicles and three different road markings that present no-parking restrictions followed by separate speed limits of 80 km/h for trucks and 100 km/h for automobiles. The system demonstrates its ability to handle streaming video through its connection to webcam or camera modules in platforms like Google Colab. The detection model must interpret diverse traffic when processing various vehicles which include vans and trucks and standard passenger vehicles. A practical usage scenario of real-time detection systems exists in intelligent transportation when autonomy requires simultaneous traffic sign and vehicle detection for decision-making and monitoring purposes.

## 6.5. Video-Based Detection Evaluation

A combined evaluation of YOLOv5 involved analyzing 10-second videos captured from busy roadways through OpenCV using the MP4 file format. Each frame processing took 30–35 ms because of which the model delivered 28–30 FPS. The system displayed robust tracking behavior because it detected moving vehicles including cars buses and trucks while keeping their bounding boxes stable between frame sequences. Class stability of the system remained exceptional because it preserved accurate labels even when dealing with motion blur and instances of partial object occlusion. The model performed consistently by detecting objects in more than 98% of video frames and keeping steady detection confidence levels throughout the entire video clip. The visual assessment confirmed that the processed video had a size of 4.1 MB while showing precise bounding box placement throughout without producing any visual artifacts during the annotation process. Testing of the system for real-time video analysis showed how it can keep accurate detections between consecutive video frames which demonstrates its practical application potential.

```
video 1/1 (175/199) /content/yolov5/American Highway travel.mp4: 640x384 13 cars, 1 bus, 1 truck, 7.1ms
video 1/1 (176/199) /content/yolov5/American Highway travel.mp4: 640x384 12 cars, 1 bus, 7.1ms
video 1/1 (177/199) /content/yolov5/American Highway travel.mp4: 640x384 13 cars, 1 bus, 10.0ms
video 1/1 (178/199) /content/yolov5/American Highway travel.mp4: 640x384 13 cars, 1 bus, 1 truck, 7.2ms
video 1/1 (179/199) /content/yolov5/American Highway travel.mp4: 640x384 12 cars, 1 bus, 1 truck, 7.1ms
video 1/1 (180/199) /content/yolov5/American Highway travel.mp4: 640x384 11 cars, 1 bus, 1 truck, 7.1ms
video 1/1 (181/199) /content/yolov5/American Highway travel.mp4: 640x384 12 cars, 1 bus, 7.2ms
video 1/1 (182/199) /content/yolov5/American Highway travel.mp4: 640x384 13 cars, 1 bus, 7.2ms
video 1/1 (183/199) /content/yolov5/American Highway travel.mp4: 640x384 14 cars, 1 bus, 7.2ms
video 1/1 (184/199) /content/yolov5/American Highway travel.mp4: 640x384 15 cars, 2 buss, 7.7ms
video 1/1 (185/199) /content/yolov5/American Highway travel.mp4: 640x384 14 cars, 1 bus, 1 truck, 7.2ms
video 1/1 (186/199) /content/yolov5/American Highway travel.mp4: 640x384 13 cars, 1 bus, 1 truck, 7.1ms
video 1/1 (187/199) /content/yolov5/American Highway travel.mp4: 640x384 10 cars, 2 buss, 2 trucks, 7.2ms
video 1/1 (188/199) /content/yolov5/American Highway travel.mp4: 640x384 14 cars, 1 bus, 2 trucks, 7.3ms
video 1/1 (189/199) /content/yolov5/American Highway travel.mp4: 640x384 12 cars, 1 bus, 1 truck, 7.3ms
video 1/1 (190/199) /content/yolov5/American Highway travel.mp4: 640x384 14 cars, 1 truck, 7.3ms
video 1/1 (191/199) /content/yolov5/American Highway travel.mp4: 640x384 14 cars, 1 truck, 7.3ms
video 1/1 (192/199) /content/yolov5/American Highway travel.mp4: 640x384 13 cars, 1 truck, 7.3ms
video 1/1 (193/199) /content/yolov5/American Highway travel.mp4: 640x384 14 cars, 1 truck, 7.3ms
video 1/1 (194/199) /content/yolov5/American Highway travel.mp4: 640x384 16 cars, 1 truck, 7.3ms
video 1/1 (195/199) /content/yolov5/American Highway travel.mp4: 640x384 13 cars, 1 truck, 1 chair, 7.3ms
video 1/1 (196/199) /content/yolov5/American Highway travel.mp4: 640x384 12 cars, 1 truck, 1 chair, 7.3ms
video 1/1 (197/199) /content/yolov5/American Highway travel.mp4: 640x384 12 cars, 2 trucks, 7.3ms
video 1/1 (198/199) /content/yolov5/American Highway travel.mp4: 640x384 13 cars, 2 trucks, 1 chair, 7.3ms
video 1/1 (199/199) /content/yolov5/American Highway travel.mp4: 640x384 12 cars, 2 trucks, 7.3ms
Speed: 0.4ms pre-process, 7.5ms inference, 1.8ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp2
199 labels saved to runs/detect/exp2/labels
```

**Fig. 4. Annotated video frame with moving vehicle detection.**

A video analysis task used YOLOv5 to evaluate surveillance footage from a highway setting as shown in Figure 4. The system annotated frames whose size was set to 640×384 pixels with a total of cars and buses and trucks together with sporadic detections of chairs. Multiple vehicle types get detected regularly by the model as it identifies more than 13 objects in each processed frame demonstrating its competence in handling heavy traffic situations. The total time required for processing one frame of video amounts to 9.9 ms with an average of 0.4 ms for preprocessing and 7.5 ms for inference and 1.8 ms for non-maximum suppression. The records indicate successful storage of results together with labels which proves the model operates efficiently while maintaining reliability in dynamic video environments.

## 6.6. Real-Time Webcam Frame Detection

The system demonstrated real-time operation through webcam frame collection from its integrated HTML5 interface at Colab notebook which passed results to the detection procedures applied on static images and video. The real-time data processing yielded correct identification results between detected individuals and cars as well as furniture items based on frame conditions. The system delivered processed results to output with an average time delay between 1.8 to 2.3 seconds that incorporated browser-to-kernel communications delays. The detection program maintained a consistent precision level regardless of how cameras were positioned or what variations occurred with indoor illumination. The detector achieved a person detection with 0.66 confidence and identified background vehicles at confidence levels between 0.78 and 0.91 when operating on an indoor view. The system performed well with frame capture delays of 1.8–2.3 seconds even though it lacked continuous streaming functionality which demonstrates its streaming capabilities for surveillance and smart environment applications.


**Fig. 5. Webcam frame with indoor person detection**

Figure 5 shows real-time webcam footage that showcases YOLOv5 detecting several vehicles on a highway as well as distinct recognition of traffic signs. The model shows flexible detection by identifying several vehicles which demonstrate different confidence ratings between 0.25 and 0.87. The vehicle detection box positions correctly next to vehicles as the traffic signs located beside them improve situational understanding. This frame demonstrates how the system can provide reliable yet real-time inference for outdoor dynamic environments which makes it suitable for future traffic management and vehicle classification systems.

### 6.6.1. Webcam Processing Delay Analysis

The latency of webcam detection which is relatively higher (1.8 to 2.3 seconds per frame) is mainly affected by the hardware and communication constraints in the cloud-based execution environment. Google Colab captures webcam frames with the browser interface and sends them to the remote processing server, which brings an extra overhead of communication. In addition, processing latency is also caused by frame encoding, transfer delays and resource sharing between GPUs. These delays are not due to the YOLOv5 model itself but rather due to the data transmission and processing pipeline.
Latency can also be greatly minimized with the deployment of the detection system to the local system whereby the frames are captured and processed without the need of communication between the browser and the servers. There are other optimization techniques, such as minimizing the input frame resolution, enabling the use of the GPU to stream continuously and using efficient frame buffering methods. These optimizations would enable real-time streaming performance in most of the real-life application.

### 6.7. Qualitative Visual Results

The detection results received visual examination to evaluate their practical capabilities. The following observations were recorded:

- Bounding Box Precision: Object boundaries received tight box placement by the system which did not extend past boundaries nor fall short of them.
- A correct assignment of label accuracy enabled proper identification between car, truck, bus, person, traffic light and motorbike categories.
- Within video frames the detection system maintained consistent object IDs which did not cause mis labelling issues.
- Detection successfully functioned across different environmental conditions that included daylight illumination along with shadows and indoor spaces as well as various angles and obscured car views.

## 7. Conclusion and Recommendation

The investigation developed an effective real-time detection platform using YOLOv5 that operated under varying environmental elements such as altered lighting and disappearing targets and shifting settings. The system type design operated through multiple entry options by processing static image files as well as video files and active webcam feeds to function reliably in any environmental condition. The model operated through Python and PyTorch on Google Colab supports GPU acceleration to speed up inference time. The model processed static images plus video files across 28 to 35 milliseconds until webcam processing required 1.8 to 2.3 seconds along with confidence ratings between 0.70 to 0.93. The model detected multiple targets in each frame by recognizing objects of different sizes and effectively processed objects which were partially obstructed. The model maintained accuracy by converting environmental alterations into proper names during its testing process. Internet usage effectiveness exists in this system but the system's throughput under webcam detection needs optimization for real-time operation within restricted resource environments.

**Recommendations for future work include:**

- Deep SORT algorithms when united with object tracking methods should sustain detection flow between video frames.
- Real-time alert and event-detection systems function at optimal speed when analytical modules are integrated into applications.
- The research explores YOLO variants and pruning/compression approaches for implementing mobile/embedded systems.
- The model training should be expanded toward custom datasets comprising various real-world environments from different traffic areas to boost context-specific detection accuracy.

## Study Limitations

Despite the fact that the proposed system provides the efficient performance in terms of the area of the real-time detection, it also has its limitations, which should be considered:

- Webcam-based detection has a latency problem as the cloud execution environment has browser-to-server communication delays, which limit the continuous real-time streaming performance.
- The accuracy of the detection can be reduced in cases where there are extreme changes in lighting, heavy occlusion or very small distant objects.
- Performance A pre-trained YOLOv5 model was used, so the model might not perform well in specialty or application-specific settings.
- The system testing was done primarily on medium-term video streams and long-term deployment conditions need additional verification.

One way this problem can be addressed in the future work is to improve the robustness of the system and the feasibility of implementing it in practice.

## Conflict of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

[1] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020, https://doi.org/10.48550/arXiv.2004.10934

[2] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018, https://doi.org/10.48550/arXiv.1804.02767

[3] 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, https://doi.org/10.1109/CVPR.2016.91

[4] N. Carion *et al.*, "End-to-end object detection with transformers," in *Proc. ECCV*, 2020, pp. 213–229, https://doi.org/10.1007/978-3-030-58452-8_13

[5] G. Jocher, "YOLOv5 by Ultralytics," GitHub, 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[6] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proc. IEEE/CVF CVPR*, 2023, pp. 7464–7475, https://doi.org/10.1109/CVPR52729.2023.00721

[7] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO series in 2021," *arXiv preprint arXiv:2107.08430*, 2021, https://doi.org/10.48550/arXiv.2107.08430

[8] X. Long *et al.*, "PP-YOLO: An effective and efficient implementation of object detector," *arXiv preprint arXiv:2007.12099*, 2020, https://doi.org/0.48550/arXiv.2007.12099

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, https://doi.org/10.48550/arXiv.1506.01497

[10] A. Dosovitskiy *et al.*, "An image is worth 16×16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020, https://doi.org/10.48550/arXiv.2010.11929

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016, pp. 770–778, https://doi.org/10.1109/CVPR.2016.90

[12] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF CVPR*, 2020, pp. 10781–10790, https://doi.org/10.1109/CVPR42600.2020.01079

[13] T. Azfar *et al.*, "Deep learning-based computer vision methods for complex traffic environments perception: A review," *Data Science for Transportation*, vol. 6, no. 1, 2024, https://doi.org/10.1007/s42421-023-00086-7

[14] A. Devasani, P. Vineeth, and S. Haripal, "Real-time object detection using deep learning for video streams," 2024, https://doi.org/10.48047/IJIEMR/V13/ISSUE05/07

[15] D. K. Pramudito, "Enhancing real-time object detection in autonomous systems using deep learning and computer vision techniques,"

*Journal of Academic Science*, vol. 1, no. 6, pp. 788–804, 2024, https://doi.org/10.59613/v3015d10

[16] M. T. Hosain *et al.*, "Synchronizing object detection: Applications, advancements and existing challenges," *IEEE Access*, 2024, https://doi.org/10.1109/ACCESS.2024.3388889

[17] S. Tuli, N. Basumatary, and R. Buyya, "EdgeLens: Deep learning based object detection in integrated IoT, fog and cloud computing environments," in *Proc. ISCON*, 2019, pp. 496–502, https://doi.org/10.1109/ISCON47742.2019.9036216

[18] M. Andronie *et al.*, "Big data management algorithms and deep learning-based object detection technologies," *ISPRS Int. J. Geo-Inf.*, vol. 12, p. 35, 2023, https://doi.org/10.3390/ijgi12020035

[19] A. Kaur *et al.*, "A survey on deep learning approaches to medical images," *Archives of Computational Methods in Engineering*, 2022, https://doi.org/10.1007/s11831-021-09649-9

[20] N. Manakitsa *et al.*, "A review of machine learning and deep learning for object detection, semantic segmentation, and human action recognition," *Technologies*, vol. 12, no. 2, p. 15, 2024, https://doi.org/10.3390/technologies12020015

[21] Y. Ghasemi *et al.*, "Deep learning-based object detection in augmented reality: A systematic review," *Computers in Industry*, vol. 139, p. 103661, 2022, https://doi.org/10.1016/j.compind.2022.103661

[22] M. J. Shafiee *et al.*, "Fast YOLO: A fast you only look once system for real-time embedded object detection in video," *arXiv preprint arXiv:1709.05943*, 2017, https://doi.org/10.48550/arXiv.1709.05943

[23] M. Ahmed *et al.*, "Survey and performance analysis of deep learning-based object detection in challenging environments," *Sensors*, vol. 21, no. 15, p. 5116, 2021, https://doi.org/10.3390/s21155116

[24] Z. Cao *et al.*, "Real-time object detection based on UAV remote sensing: A systematic literature review," *Drones*, vol. 7, no. 10, p. 620, 2023, https://doi.org/10.3390/drones7100620

[25] M. Simon *et al.*, "Complexer-YOLO: Real-time 3D object detection and tracking on semantic point clouds," in *Proc. IEEE/CVF Workshops*, 2019, https://doi.org/10.48550/arXiv.1904.07537

[26] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019, https://doi.org/10.1109/TNNLS.2018.2876865

[27] S. S. A. Zaidi *et al.*, "A survey of modern deep learning based object detection models," *Digital Signal Processing*, vol. 126, p. 103514, 2022, https://doi.org/10.1016/j.dsp.2022.103514

[28] D. T. Hristopulos *et al.*, "Open challenges in environmental data analysis and ecological complex systems," *Euro physics Letters*, vol. 132, no. 6, p. 68001, 2021, https://doi.org/10.1209/0295-5075/132/68001

[29] N. H. Abdulghafoor and H. N. Abdullah, "A novel real-time multiple objects detection and tracking framework for different challenges," *Alexandria Engineering Journal*, vol. 61, no. 12, pp. 9637–9647, 2022, https://doi.org/10.1016/j.aej.2022.02.068

# الكشف عن الأجسام في الوقت الحقيقي في البيئات المعقدة: توظيف تقنيات التعلم العميق والرؤية الحاسوبية

**داود سالم حسين[١]، هبة عدنان رحيم[٢]*، مها صبري التميمي[٣]، يان لي[٤] , شهاب عبد الله[٥]**

[١] *كلية الطف الجامعة، قسم هندسة تقنيات الحاسوب، كربلاء، العراق*

[٢،٣] *كلية علوم الحاسوب وتكنولوجيا المعلومات، جامعة كربلاء، كربلاء، العراق*

[٤] *كلية الصحة والهندسة والعلوم، جامعة جنوب كوينزلاند، استراليا*

[٥] *الكلية المفتوحة، جامعة جنوب كوينزلاند، استراليا*

*البريد الالكتروني: hiba.adnan@uokerbala.edu.iq

**المستخلص**

يتناول هذا البحث التحديات الرئيسة في الكشف الآني (Real-Time Detection) مثل تغير تأثيرات الإضاءة، وحجب الأجسام، وتعقيد عناصر الخلفية، واختلاف أحجام الأجسام، إذ تظهر هذه المشكلات بشكل متكرر في تطبيقات القيادة الذاتية وأنظمة المراقبة والبنية التحتية الذكية. تم تطوير نظام خط أنابيب للكشف (Detection Pipeline) يتميز بوظائف معيارية تتيح معالجة الصور ومقاطع الفيديو وكاميرات الويب في الوقت الحقيقي، مع تحسينات خاصة لكلّ نوعٍ من أنواع المدخلات. تم اختيار نموذج YOLOv5s لما يتمتع به من دقة وسرعة في الأداء، حيث تم نشره في بيئة سحابية باستخدام Google Colab مع إمكانيات تسريع المعالجة بواسطة وحدة معالجة الرسوميات (GPU). وقد أسهم جمع البيانات من العالم الحقيقي في إجراء تحليل كمي من خلال تقييم زمن الاستدلال، وسرعة الإطارات، ودقة الكشف، بالإضافة إلى قيم الثقة، بينما اعتمد التقييم النوعي على قياس دقة مربعات الإحاطة (Bounding Boxes) وصحة التسميات. أظهرت نتائج النظام أداءً متميزاً، حيث تمكن من معالجة الصور وبيانات الفيديو خلال ٢٨ إلى ٣٥ مللي ثانية، ومعالجة إطارات كاميرات الويب خلال ١,٨ إلى ٢,٣ ثانية، مع توليد درجات ثقة تراوحت بين ٠,٧٠ و٠,٩٣. وتفيد هذه النتائج التطبيقات الآنية لأنها توفر كشفا مستقرا مع قدرة على التكيف مع الظروف البيئية المختلفة وإمكانية تطبيق عملية. كما تظهر النتائج التجريبية أن نموذج YOLOv5 يتمتع بدقة عالية، مما يشير إلى إمكانية نشره مستقبلاً في أنظمة المراقبة البصرية الذكية عبر مختلف البيئات الديناميكية.