Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

Research Article

Automated Object Detection and Count Estimation Based on Machine Learning Models

[1] Rafil Mohammed Jameel
University of Information Technology and Communications (UoITC)
Baghdad, Iraq
rafel.mohammed@uoitc.edu.iq

[2] Mohammed Abomaali
College of Information Technology Engineering,
Al-Zahraa University for Women
Karbala,Iraq
aboalmaali@alzahraa.edu.iq

ARTICLE INFO
Article History
Received: 30/07/2025
Accepted: 05/10/2025
Published: 30/11/2025

ABSTRACT
Object detection and counting is a crucial problem in various areas like autonomous machines, health sector and industrial automation. This paper presents a comparative study between two Machine Learning (ML) methods, depending on Single Shot Multi Box Detector (SSD) with You Only Look Once (YOLOv3) and MobileNetv3 to focus on the object detection and counting. The well kown dataset (COCO) used to evaluate the two models where 44 images chosen randomly for testing. In addition SSD and MobileNet v3 compared in a real time mode. The performance metrics used in this paper were confidence scores, average over image and processing time. The results indicate that there was an tradeoff between accuracy and speed. Both models showed a fast inference time, which made it suitable for real-time applications, although (YOLOv3) was more confident in some cases because of its complex architecture. The performance difference is caused by the design of the model: MobileNet's simple structure aims at lightweight, while (YOLOv3)'s complex network enhance robustness in detection. This work highlights The necessity for application-aware model selection balancing between the fast performance on edge devices (such as drones) versus accuracy for accuracy-precision applications (like medical imaging). The findings provide practical insights for deploying ML-driven object detectors across various domain.

Keywords: Object detection, Object counting, Machine Learning(ML), (SSD), (YOLO), MobileNet, (COCO) dataset.

1. INTRODUCTION

ML, which is a data analysis technique, serves as techniques that automate the building of analytical models. It is one of the branches of Artificial Intelligence (AI) and it involves the development of systems capable of autonomously learning by identifying patterns in data, and using those patterns to draw conclusions without human involvement. This is important because it allows the companies to identify trends on the consumer's behavior and acts subsequently helps in creating a new product. ML algorithms are ubiquitous in many current successful companies (e.g., Uber, Google, and Facebook). ML has emerged as a crucial factor in the marketplace for several organizations [1].

Image processing is a subset of computer vision. It includes various manipulation and analysis of the image to create another new image, such as edge detection using masking techniques or extracting the information from an image. Object detection and identification includes the process of recognize the object type and features in the input image, which then takes some automatic action. Object detection refers to the task of locating and recognizing an object in an image along with its orientation. The object classified by AI) algorithms, like neural networks (NN) [2]. Object detection is an important research area of computer vision, Deep Learning (DL), AI and so on. It is vital process to implement more advanced computer vision jobs, such as target tracking, event detection, behavior analysis and semantic scene understanding. The aim is to be able to detect the requested object in an image accurately, classify it correctly, and find the exact location of the object with respect to each target. It is widely adopted in the areas of autonomous vehicle navigation, video and image retrieval, intelligent video surveillance, medical image analysis. [3]. The development of ML techniques have transformed computer vision, particularly the object detection domain. Despite of the great progresses made in this field challenges still exist in accurately localizing and counting objects

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

under complex conditions including scale variations, occlusions, clutters and illumination changes. Traditional detection systems often face these limitations and issues with localization, categorization or counting of the number of objects [4].

To address these limitations, there is a strong demand for robust ML models that reliably detect, localize and quantify objects in various contexts. This model should be used to handle variations of object size, orientation and appearance without accuracy deterioration in bounding box estimation and classification. The primary objective is to create an efficient and flexible solution that obtains high accuracy in detection and count estimation, hence improving the state-of-the-art for practical applications in uncontrolled situations. This study improves the domain of computer vision by:

1. Providing an empirical comparison of (YOLOv3) and (SSD)-Mobile Net performance characteristics.
2. Developing an integrated framework for both offline and real-time object detection.
3. Quantifying the relationship between object density and detection metrics.

The rest of the paper is organized as follows: Section 2 presents a review of related work. Section 3 provides a detailed description of the proposed system. Section 4 illustrates the results and discussion of the proposed system. Section 5 indicates to the paper's conclusion.

2. RELATED WORKS

The Many studies in the domain of object detection and counting are presented below: In [5], the study investigates poorly supervised object detection and counting with (CNNs), employing partially labeled datasets that indicate simply the existence of objects without specifying their locations. The findings show that weak labels can predict the object locations, which reduces annotation cost while keeping comparable performance. The method matched with fully supervised techniques use accurate bounding boxes, it highlights the trade-offs between annotate accuracy and efficiency. The results indicate the efficiency of poorly supervised learning for object detection and counting especially in contexts where extensive labeling is impractical.

In [6], Mask (R-CNN) used for object detection, counting, and segmentation. This DL approach automatically finds objects, and learns to produce precise instance masks in parallel with the object categories while balancing the precision of the mask and class predictions. Running at five (FPS), it provides both segmentation masks and bounding boxes for deep object processing. In [7] an automated system for object detection, segmentation, and counting via (DL), using Mask (R-CNN) to perform pixel-wise segmentation it also use (YOLO) for fast object recognition presented. In this study, the methodology generate accurate instance masks and counting objects with confidence, creating an emphasis on the estimate fruit yield in agricultural applications. Mask (R-CNN) achieves up to 5 (FPS), striking the balance between precision and speed, while for real-time detection performance improvement, put(YOLO). The approach, coded in OpenCV with TensorFlow, demonstrated to be potentially feasible for automatic object recognition application especially for fruit counting within precision agriculture

In [8], the authors introduce Few-Shot Object Counting and Detection (FSCD), a novel two-stage approach that leverages an uncertainty-aware few-shot detector to generate pseudo ground-truth bounding boxes and a refined training strategy to mitigate pseudo-label errors. The method, evaluated on the (FSCD)-147 and (FSCD-LVIS) datasets, outperforms state-of-the-art few-shot baselines in terms of counting accuracy and detection performance with robustness to low data conditions. The method improves few-shot learning by effectively reducing pseudo-label noise while maintaining state-of-the-art accuracy in object localization and counting.

Acloud-based DL object counting system introduced in this study. The proposed system improved by a novel (DBC-NMS) technique and hyper parameter tuning. (CARPK, SKU110K) and a proprietary pill dataset are used to evaluate this work, it achieves a Root Mean Square Error (RMSE) of 1.20 and a Mean Absolute Error (MAE) of 1.03, which indicates higher accuracy in dense object detection. The system combines low-cost hardware with cloud computation, for the best cost-performance ratio towards realistic counting applications. A summary of related works is given in Table (1).

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

TABLE I Summary of Related Works

Ref No.	Dataset	Used approach	Result
[1]	PASCAL (VOC) 2007 and 2012.	(CNN) and (MESA) for finding the distance between two target images may be defined as the largest absolute difference between the sums over all box sub-arrays.	The integral of the resulting target images resulted in some random number, which did not correspond with the number of objects present in the image.
[2]	(CVPR) 2015 and 2016	(CNN), and Mask (R-CNN)	Mask (R-CNN) gets good quality of outcome.
[3]	Two datasets Cats & Dogs and Cars dataset.	(CNN) using Open (CV), Tensor Flow, and (YOLO).	The accuracy was approximately 97% in counting the fruits.
[4]	Two new datasets named FSCD-147 and FSCD-LVIS has been used.	Dual-phase training strategy. Initially, pseudo-(GT) bounding boxes are created from dot annotations, followed by training Counting-DETR on these pseudo-(GT) boxes for predicting bounding boxes for a new object class.	The (MAE) was 20.38 for counting, and the value of (AP50) was 41.90 for detection.
[5]	(CARPK) and (KU110K) datasets and custom dataset of small pills.	(YOLO), Distance between Circles-Non Maximum Suppression (DBC-NMS), CenterNet model and hyper-parameter tuning, and using cloud-based (DL) software server.	The (MAE) was 1.03, while the (RMSE) was 1.20 on Pill dataset, and the (MAP) was 23.7, while the Frames Per Second (FPS) was 33.6 when utilizing Tiny (YOLO).

3. MATERIALS AND METHODS

3.1 Machine Learning (ML)

ML, a fundamental domain of (AI), allows systems to independently identify patterns and derive insights from data. Through the utilization of computing improvements, (ML) improves data analysis, measure, and interpretation across various domains. Its ability to effectively handle massive datasets is what has allowed it to gain popularity and drive development in data-driven decision-making [10].

The use of (ML) appears in various fields, including medicine, economics, and climatology, by facilitating effective feature extraction, classification and prediction. It depends on data accuracy frequently surpasses traditional methods; enhance performance in complex analytical tasks [11].

To improve an efficient prediction mode the ML process involves a methodical process. This process starts with formulation of the problem by defines the task, and choose an appropriate (ML) algorithm.

Next, collecting data from diverse sources for training and evaluating the model [12]. The following step is data prepossessing, which handle missing values, scales features and fixes outliers to guarantee data quality. Feature selection or extraction subsequently selects or changes important features to improve the model performance. Constructing the ML model step choose appropriate algorithm depending on the type of problem, data attributes, and the desired outcomes [13].

Then, the model-training step include splitting the data into training and testing groups, parameters fine-tuned by diminishing the loss function. Finally, to evaluate the model performance metrics used like as accuracy, precision, recall, or (MSE) to verify generality ability. This iterative procedure ensure reliability and strength in (ML) applications [14]. Figure 1 demonstrates the (ML) stages

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

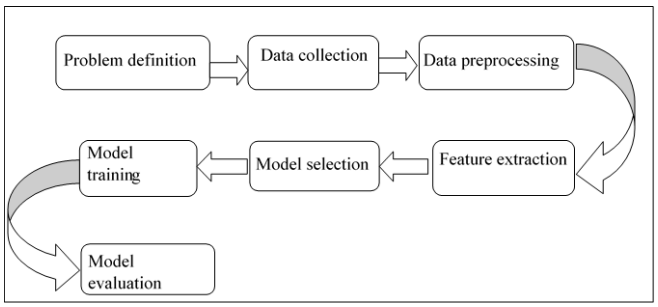

Fig 1. *The steps of (ML)*

3.2 Object Detection. A one of computer vision techniques detects and classifies the objects in images or videos using its bounding boxes (rectangular markers) by locate objects with its class labels. It allows multi-object recognition in a single frame through overlapping bounding boxes, providing model trained on the related object categories. Figure 2 illustrates the concept of object detection [15].

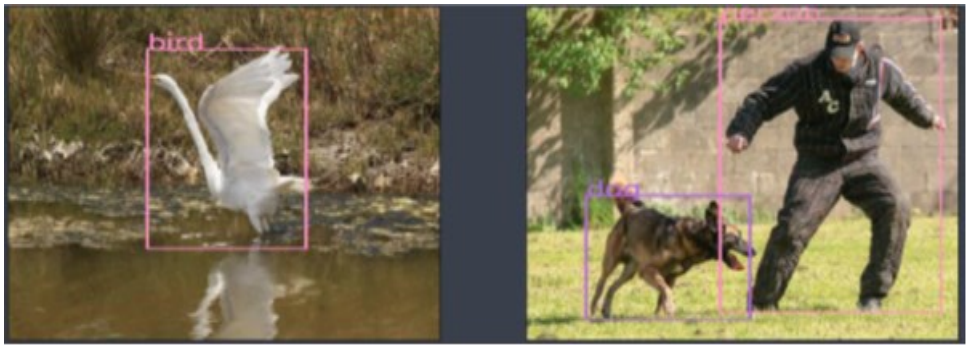

Fig 2. *The concept of object detection [15]*

ML) has become highly effective for object detection and counting through a structured three-stage procedure: dataset preparation with annotated images, model selection (ranging from traditional ML to rich DL architectures such as CNNs, YOLO or Faster R-CNN, and model training using gradient based optimization on error in detection. Various datasets and architecture selection are very important in model performance [16].

3.3 Object Counting

Object counting includes count and detect objects in images or videos using various computational methods. Model use density estimation (summing density maps) for image counting, regression techniques (map features to counts). Video counting incorporates temporal methods such as optical flow, frame differencing and RNNs to locate objects over frames. An object-counting example shown in Figure 3 [19].

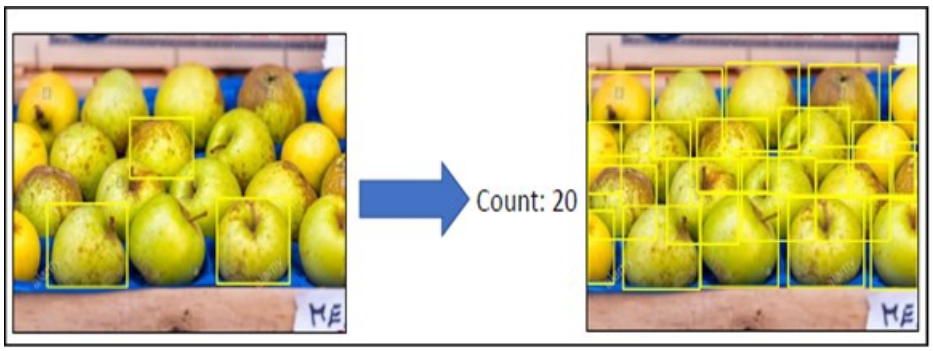

Fig 3. *The concept of object counting [19]*

3.4. The Mobile Net (SSD) Model

The Mobile Net-(SSD) model is one of the most efficient (DL) architecture used for detect objects in real-time, which consists of the lightweighted MobileNet as a backbone and SSD framework. MobileNet uses depth-wise separable

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

convolution which separates standard convolution into depth-wise and point-wise convolution, dramatically reducing computation complexity [20]. Depth-wise convolution applies one filter per input channel and pointwise convolutions combine the outputs through 1×1 conv. This operation can both decrease the number of parameters and the FLoating-Point Operations (FLOPs), in the meantime preserve detection performance [21]. The (SSD) framework leverages multi-scale feature maps for detecting objects at various sizes, with predefined default boxes and prediction heads for class scores and bounding box offsets [22]. The combination of MobileNet with (SSD) includes replacing traditional backbones (e.g., VGG) with MobileNet and adjusting detection heads to match reduced feature dimensions. This results in a compacted model appropriate for mobile and embedded devices, attaining a balance between speed and accuracy. Training usually requires transfer learning based on ImageNet, data augmentation, hard negative mining following [14], and optimizing using by SGD with momentum [23].

3.5. The (YOLO V.3) model

(YOLO v3) model is an end-to-end single-stage new framework that exceeds the previous detector models such that accuracy of detection will be higher and real-time speed preserved. In contrast to two-stage detectors, (YOLOv3) is capable of performing object localization and classification within a single forward pass in an efficient manner [24]. The design uses a revised Darknet-53 model with residual connections and more layers for better feature extraction. (YOLOv3) presents a multi-scale prediction that incorporates three distinct grid sizes (13×13, 26×26 and 52×52) for detection of the objects in different scales to solve the difficult problem of detecting small object. Furthermore, it uses logistic regression to derive class prediction instead of Softmax, which in turn enables multi-label classification. The model also built using Feature Pyramid Networks, which fuses high-resolution features with semantically rich deep features to benefit detection for objects at different scales [25].

4. THE PROPOSED SYSTEM

Two models used in the proposed system: Mobile Net with (SSD) and (YOLO v3) to detect and count objects separately. For offline image, the system applied to detect objects using two models and for online image, it uses only Mobile Net (SSD) model. Figure 4 shows the models of the proposed system.

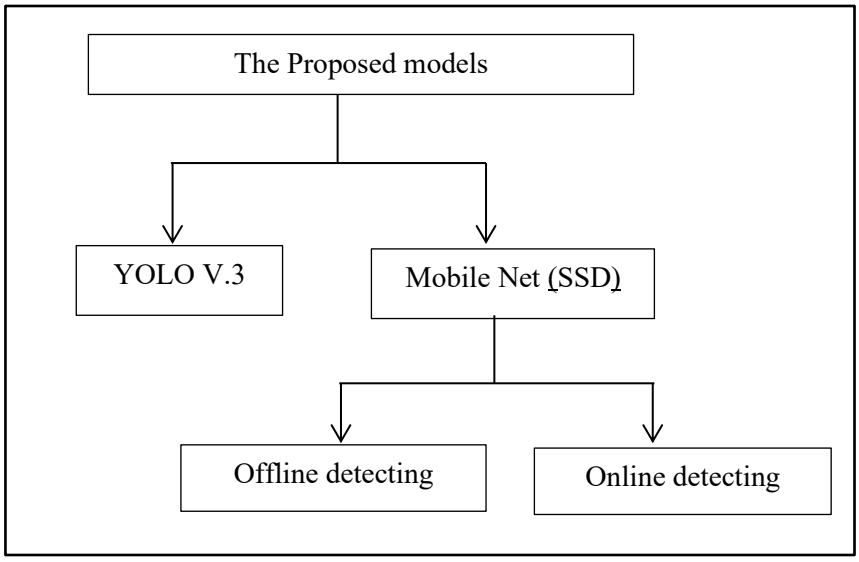

Fig. 4 *The block diagram of the proposed models*

4.1The Dataset

The proposed method utilizes the Common Objects in Context (COCO) dataset, a prevalent benchmark in computer vision for object detection, segmentation, and image captioning. This dataset contains varied images with labeled bounding boxes, segmentation masks, and written descriptions, enabling multi-task learning and algorithm evaluation. It encompasses 91 object types, including individuals, vehicles, and household items, across diverse indoor and outdoor environments, thereby facilitating comprehensive model training. COCO's comprehensive annotations, especially its instance-level segmentation masks and contextual captions, support advanced tasks like precise object localization and visual-language understanding. Its standardized evaluation

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

protocols (e.g., mAP metrics) make it indispensable for comparative research in object detection (e.g., YOLO, Faster R-CNN) and beyond. [26].

4.2. The Proposed System using Mobile Net (SSD) model

This model contains six main stages. Begins with loading the image stage, then preparing for object detection, loading object detection model, detecting objects in the image and draw bounding boxes, labeling the detected objects, counting the objects in the image, computing the average of confidence value, computing the time of detected objects, and finally, displaying the results of detected objects. The general block diagrams of this proposed system are illustrated in Figure 5.

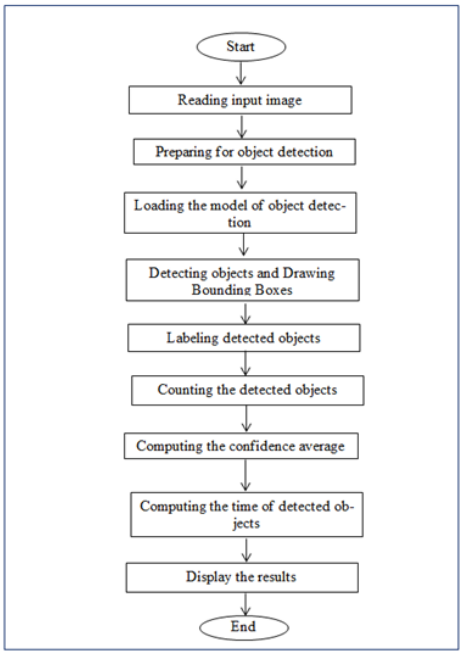

Fig. 5 *The general block diagrams using (SSD) Model*

4.2.1 Preparing for Detecting objects

At this level, there are two critical steps to establish the object classification framework. First, creating a list of class names to store the names of the object classes which are recognized by the object detection model. Second, specifying the file path that contains the object class names. The work utilized the 'COCO names' file, a typical component of (COCO) dataset, a comprehensive collection of labeled images employed for training object detection models. The "COCO names" dataset includes (91) object class names, such as human, bicycle, vehicle, motorcycle, airplane, bus, train, truck, boat, and others.

4.2.2 Loading the object detection model

In this level, the implemented object detection system employs a pre-trained Single Shot MultiBox Detector (SSD) model with MobileNetV3 as its backbone network, leveraging weights learned from extensive training on labeled (COCO) datasets. The (SSD) architecture enables efficient single-step identification by concurrently predicting object classes and bounding boxes using multi-scale feature maps and preconfigured anchor boxes that are refined using learned image features. For successful deployment on edge devices, the MobileNetV3 is designed by leveraging inverted residuals with linear bottlenecks, adaptive channel-wise feature recalibration using squeeze-and-excitation modules and smaller as well as faster depth wise separable convolutions. Model deployment requires two important files: (i) a configuration file, which is a description of the network architecture and other hyper parameters; and (ii) weight-file with static parameters resulting from large fine-tuning training for transfer learning with frozen parameters not to be able further modification at inference time.. The integration of the (SSD) detection method with MobileNetV3's efficient feature extraction gets an optimal ratio between detection accuracy and computing efficiency in real-time applications.

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

4.2.3 Object Detection with a Trained Model

The Mobile Net v3 - based (SSD) model comprises three primary steps: (feature extraction, prediction, decoding and post-processing) layers. At first, the feature extraction phase uses convolutional and pooling layers to extract the image's features. The convolutional layers employ diverse filters for the image at various scales and resolutions, while pooling layers are utilized to decrease the feature maps and diminish the spatial dimensions.

Secondly, the prediction layers are used for predicting the probability of each object class in the feature map and the location of each class in this map. Lastly, the decoding and post-processing step layers are for removing the overlapping and redundant bounding box for the same object. The steps that follow describe the procedure for detecting and counting objects in image:

1. Resizing the input image into (320x230) pixels.
2. Normalizing the values of an image between (1 and -1) by dividing the pixels by (127.5). Therefore, the intensities of pixels are ranged between (-1) and (1), which is commonly used for pre-trained models.
3. Calculating the mean subtraction by subtracting from each channel of the image before normalizing. Mean subtraction computation centers the data around zero for enhancing the model's performance.
4. Adjusting the channel order by swapping between the blue and red channels in the image to convert it to RGB format in order to match the pre-defined model format.
5. Utilizing (CNNs) to extract features and classify them based on learned weights. The (CNN) architecture contains: Convolutional Layers: These layers extract features from the image by applying filters that slide across the image.

   a. The Activation Functions: adds non-linearity to the network, such that it understand the complicated relationships among features.

   b. Pooling Layers: By reducing the spatial dimensions of feature maps, these layers enhance the model's ability to resist variations in the images.

   c. Batch Normalization is a technique that normalizes the activations of each layer to enhance training performance and stability.

6. Using Non-Maximal Suppression (NMS) method to eliminate redundant bounding boxes for the same object .
7. Computing the Confidence, only detections with confidence values below (0.5) are considered invalid. The value of the minimum confidence is (0.5). The algorithms (1) and (2) show the algorithms of two first steps of the (SSD).

Algorithm (1) Extracting the Feature
Input: Input Image
Output: Feature Maps

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

Begin:

Step1: Looping through convolutional layers.

- Convolution of the filters.
- Computing (ReLU) activation function for feature maps.
- Computing pooling max to downsample the feature maps.

-Repeating for multiple convolutional branches.

Step2: Generating separate feature maps

End

Algorithm (2) Prediction Layers

Input: Feature Maps

Output: Confidence Value and Bounding the Box

Begin:

- Looping for each feature map:-

Step1: Computing Softmax function for convolution layer.

Step2: Computing the probability for each object in image at each location using Eq. $P(c \mid x) = \exp(y_c) / \Sigma_i \exp(y_i)$

Step3: Computing linear activation function for convolution process.

Step4: Determining the anchor boxes using predicting offset (dx, dy, dw, dh)

End

4.2.4. Labeling Objects

Depending on the (COCO) dataset used for training, a box is bounding around each object in the input image at this stage, and each object is labeled with its class ID.

4.2.5 Counting Objects

In this stage, the number of objects in each input image is calculated and shown.

4.2.6 Computing Confidence and Time

In this stage, the average of confidence value for each detected object is computed, and the time of detected object is also computed.

4.2.7 Displaying the Results

In this stage, the values of the detected time, value of confidence average, counting of the objects, bounded box for each object, and label of each object, are all displayed.

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

The Proposed System using (YOLO) model

In this method, the (YOLO V.3) model with its pre-trained weights is used. It consists of six primary stages. Performing image analysis for object detection using (YOLO), including loading the detection model, processing results, applying Non-Maximum Suppression (NMS), drawing bounding boxes, labeling objects, counting them, calculating detection time, average confidence, and displaying the final results.

4.3.1 Reading the Image

In this stage, the input image is read by loading the path to the image file, then the image is converted into two dimensions' array, A wide variety of image formats, like (JPEG), (TIFF) and (PNG) are supported.

4.3.2 Preparing Image for Object Detection

In this stage, the image size is resized into smaller size (fx=0.4, fy=0.4) for faster processing.

4.3.3 Loading the Model

The (YOLO v3) model is loaded by loading the weights file, the configuration file, and reading a list of classes from the (COCO). File names are at this stage. The weight file stores the pre-trained weights of the (YOLO V3) model, which were acquired during training on a substantial dataset. The configuration file is the (YOLO) neural network structure, which includes a sequence of convolutional and fully connected layers. The network is comprised of numerous blocks, each block have multiple layers. Typical layer types consist of convolutional layers, batch normalization layers, and shortcut connections. The input layer, usually represented by the image size (width and height), while the output layer is indicated by the anchor boxes number , classes to be detected, the confidence value of each object, and the time for detecting each object. The layers of the (YOLO) models are responsible for predicting bounding boxes and class probabilities. Layers numbers dictate the anchors number and anchor boxes dimensions for each (YOLO) layer. The Rectified Linear Unit (ReLU) activation function is used and the loss function is employed for network training.

4.3.4 Detecting the Objects

In this stage, the objects are detected by processing the input image, generating the feature map of this image. A forward neural network is used for detecting objects and extracting (the ID of classes, the confidence value, and the detected time). Algorithm (3) shows the steps of detecting the objects.

4.3.5 Applying (NMS)

In this stage, the (NMS) technique is used to remove the bounding boxes overlapping by removing the redundant bounding boxes and keeping only the most confident ones for each detected object. Therefore, the set of non-overlapping bounding boxes remains by sorting the bounding boxes depending on their confidence scores. So, the boxes with higher confidence scores are placed. The confidence value of the object must be higher than the confidence threshold, and it removes the overlapping boxes by checking if there are other boxes with higher confidence scores that overlap significantly with it.

Algorithm (3) Detecting the Objects
Input: Input Image, (YOLO) model, list of class names
Output: Detecting objects, confidence score, class_ name of objects

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

Begin:

Step1: Processing the input image by the following:

- Resizing the image by multiply the height and width *0.4.
- Normalizing the values of image.
- Specifying the image size by (416*416) pixel.
- Scaling the image. So, image's pixel value between (0,1).

Step2: Generating separate feature maps.

Step3: Performing a forward pass through the neural network.

Step4: Extracting confidence scores and class predictions.

Step5: If the confidence score exceeds a threshold (0.5):

- Calculate bounding box coordinates relative to the resized image size.
- Append bounding box coordinates to the boxes list.
- Append confidence score to the confidences list.
- Append class ID to the class_ids list.

End

4.3.6 Drawing Bounding Boxes and Labeling

In this stage, the bounding box is drawn around the detected object, displaying the label of the class, confidence value, and the detected time.

4.3.7 Counting the Objects and Displaying Results

In this stage, the average confidence of all detected objects is calculated, the detected objects number is calculated, the total detected time for objects is displayed, and finally, detected objects with bounding boxes, the label of objects, the average confidence, and the average time are displayed.

5. Results and Discussion

The proposed system was implemented using Python as a scripting programming language, and TensorFlow, PyTorch, and OpenCV libraries for object detection. PyCharm is utilized as an environment, (COCO) as a dataset, and the time library for computing time. The proposed system contained two models: Mobile Net (SSD) and (YOLO V.3) models. In the Mobile Net (SSD) model, the detection and counting of objects in images were applied both offline and online; and the (YOLO V.3) model was applied on offline images. The proposed system was applied to (44) images, which were selected randomly from the (COCO) dataset.

The Mobile Net (SSD) model detects and counts objects in images in offline mode, calculates the average time and confidence, and displays the objects with various boundary colors.

Figure (6) displays the original image, whereas Figure (7) shows the image after object detection. While Figure (8) shows the image objects detection and counting in online mode using camera. Also, the (YOLO V.3) model is applied for detecting and counting objects in images. Figure (9) shows the image after object detection by using (YOLO V.3).

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

Fig. 6: The original image

Fig. 7: The image after detecting objects

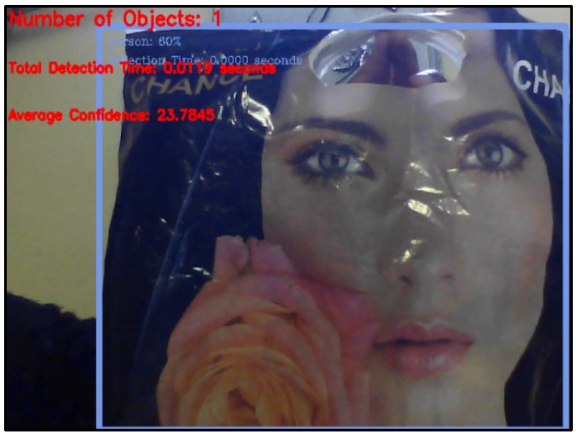

Fig. 8 The image after detecting objects real-time using (SSD) model

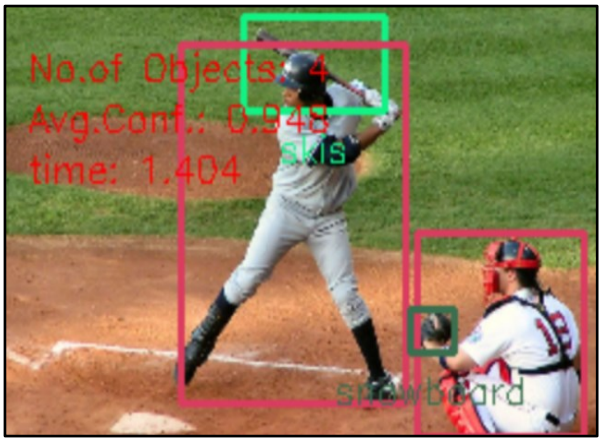

Fig. 9 The image after detecting objects using (YOLO. V3) model

As shown in Fig. (7), the number of the detecting objects are (3), the average of detecting time is (0.0010), and the average confidence for objects in the image is (0.1969). While in Fig. (9), the number of detected objects in image was (4), the average time of detecting objects was (1.404), and the average confidence for all objects in image was (0.948). Table (2) shows the values of confidence average using Mobile Net with (SSD) for all images, and figure (10) shows the histogram of confidence average for each image by using the same model

TABLE II The confidence values for all images using Mobile Net SSD model

Image No.	Confidence Value	Image No.	Confidence Value
1	0.2319	23	0.3474
2	0.0505	24	0.0511
3	0.2514	25	0.1969
4	0.7462	26	0.3881
5	0.1532	27	0.1720
6	0.3281	28	0.8503
7	0.1097	29	0.1103
8	0.1688	30	0.1337
9	0.1955	31	0.1514
10	0.1254	32	0.0834
11	0.7400	33	0.3623
12	0.1094	34	0.0584
13	0.232	35	0.1707
14	0.1671	36	0.0974
15	0.3209	37	0.0299
16	0.0556	38	0.0868
17	0.1623	39	0.2748
18	0.8006	40	0.0864

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

19	0.1365	41	0.1722
20	0.2637	42	0.0456
21	0.1710	43	0.0422
22	0.0723	44	0.1082

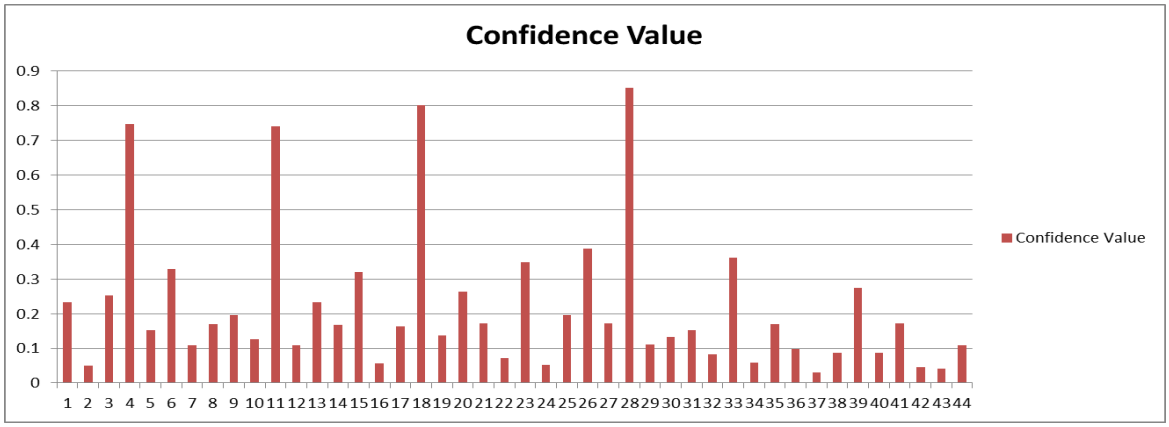

Fig. 10 *The histogram of confidence average for each image using Mobile Net with (SSD) model*

As shown in Table (2) and Fig. (10), the value of the confidence average for each image was different from one image to another which is depending on the number of objects which was detected and the value of confidence for each object in each image. Image (28) got the best confidence average value, while image (37) got the lowest confidence average value. Table (3) shows the values of confidence average of the (YOLO V.3) model of the same images, and Figure (11) shows the histogram of confidence average for each image using the (YOLO V.3) model.

TABLE III The confidence values for all images using (YOLO V.3) model

IMAGE NO.	CONFIDENCE VALUE	IMAGE NO.	CONFIDENCE VALUE
1	0.90	23	0.889
2	0.879	24	0.907
3	0.971	25	0.948
4	0.903	26	0.886
5	0.807	27	0.819
6	0.890	28	0.921
7	0.752	29	0.889
8	0.768	30	0.760
9	0.805	31	0.86
10	0.778	32	0.756
11	0.925	33	0.810
12	0.858	34	0.642
13	0.833	35	0.966
14	0.778	36	0.949
15	0.836	37	0.802
16	0.830	38	0.789
17	0.860	39	0.757
18	0.952	40	0.933
19	0.827	41	0.836
20	0.855	42	0.740
21	0.882	43	0.754
22	0.746	44	0.862

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

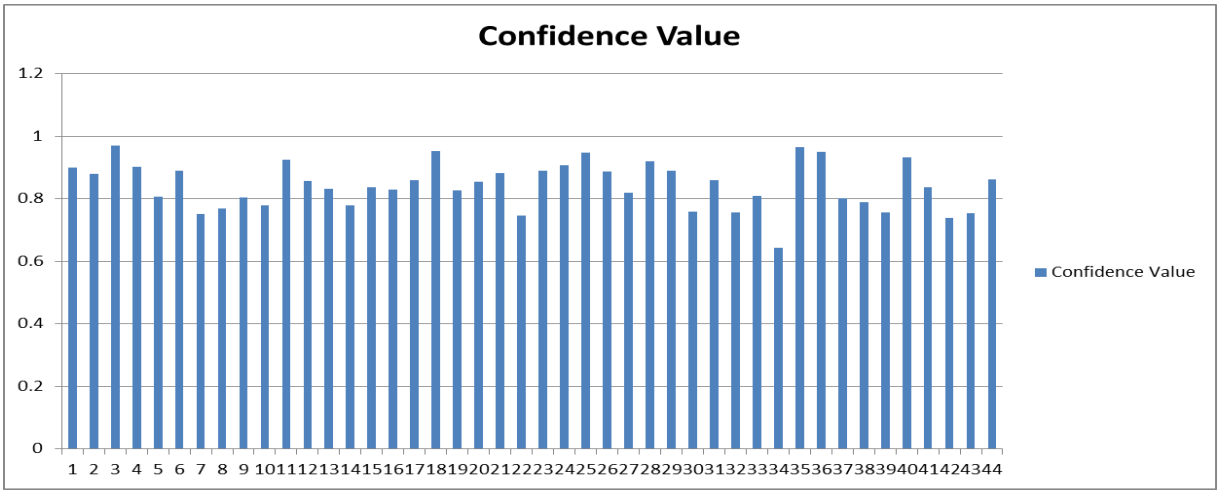

Fig. 11 *The histogram of confidence average for each image using (YOLO) model*

As shown in table (3) and Fig. (11), the value of the confidence average for each image was different from one image to another which is depending on the number of objects which was detected and the value of confidence for each object in each image. Image (3) got the best confidence average value, while image (34) got the lowest confidence average value. As shown in (Tables (2) and (3)) and (Figures (10) and (11)), the average confidence was calculated by computing the confidence of each detected object in the image, with a confidence threshold set at (0.5). The average confidence of the (YOLO V.3) model was higher than that of the Mobile Net with (SSD) model. Table (4) and Figure (12) display the average time used to detect objects in image for the Mobile Net with (SSD) model and the histogram of time average for each image respectively.

TABLE IV *The time average for each image using Mobile Net with (SSD) model*

Image No.	Avg. Time	Image No.	Avg. Time
1	0.0010	23	0.0000
2	0.0010	24	0.0020
3	0.0009	25	0.0010
4	0.0010	26	0.0010
5	0.001	27	0.0010
6	0.0010	28	0.0000
7	0.0009	29	0.0010
8	0.0010	30	0.0012
9	0.0010	31	0.0020
10	0.0010	32	0.0010
11	0.0000	33	0.0010
12	0.0000	34	0.0000
13	0.000	35	0.0000
14	0.0010	36	0.0015
15	0.000	37	0.0030
16	0.0000	38	0.0000
17	0.0000	39	0.0010
18	0.0000	40	0.0000
19	0.0010	41	0.0010
20	0.0000	42	0.0010
21	0.0000	43	0.0030
22	0.002	44	0.0000

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

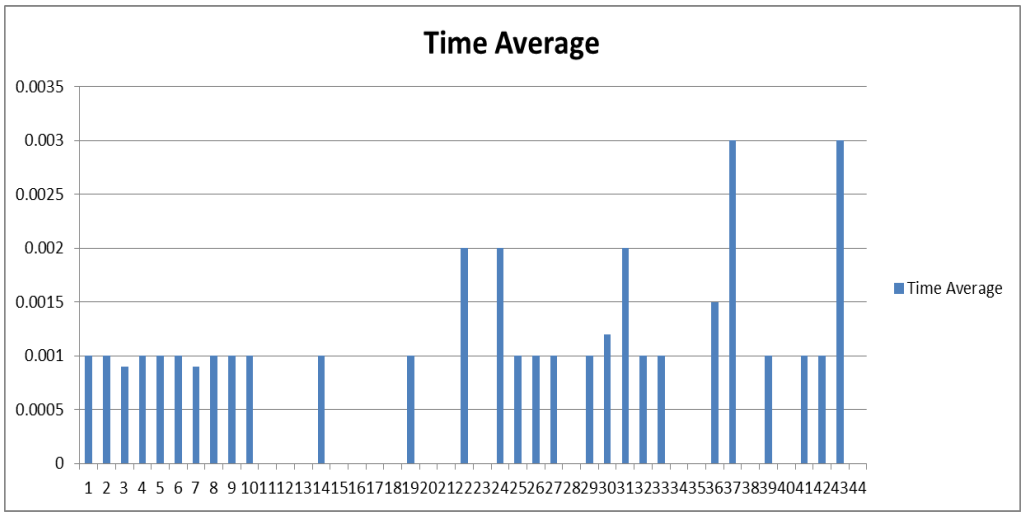

Fig. 12 The histogram of time average for each image using Mobile Net with SSD model

As shown in Table (4) and Fig. (12), the value of the time average for each image was different from one image to another which is depending on the number of objects which was detected. The time average value for more than one image was zero, while image (37) and (43) got the highest detected time average. Table (5) and Fig. (13) display the average time used to detect objects in image for the (YOLO V.3) model and the histogram of time average for each image respectively.

TABLE V *The Time Average Using (YOLO V.3) Model*

Image No.	Avg. Time	Image No.	Avg. Time
1	1.718	23	1.412
2	1.785	24	1.591
3	1.689	25	1.404
4	1.815	26	1.570
5	1.820	27	1.470
6	1.800	28	1.506
7	1.769	29	1.444
8	1.787	30	1.440
9	1.747	31	1.393
10	1.761	32	1.425
11	1.909	33	1.374
12	1.717	34	1.556
13	1.845	35	1.810
14	1.777	36	1.405
15	1.747	37	1.668
16	1.864	38	1.488
17	1.721	39	1.501
18	1.817	40	1.617
19	1.749	41	1.406

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

20	1.944	42	1.465
21	1.929	43	1.387
22	1.612	44	1.579

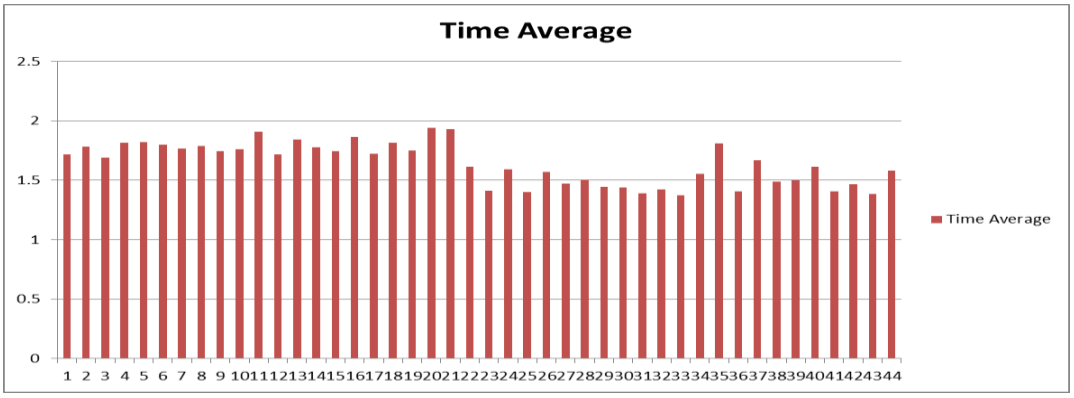

Fig. 13 The histogram of time average for each image using YOLO model

As shown in Table (5) and Fig. (13), the value of the time average for each image was different from one image to another which is depending on the number of objects which was detected. Images (20) and (21) got the highest detected time average, while image (43) got the lowest detected time average.

As shown in (tables (4) and (5) and Figures (12) and (13)), the average time had been resulted from calculating the processing time of each detected object in image. By comparing between Mobile Net (SSD) and (YOLO V.3) models, the average time of Mobile Net (SSD) model was less than the average time of (YOLO V.3).

5.2 The Discussion of Results

As shown in the results, two object detection models--MobileNet with (SSD) and (YOLO V.3) by analyzing their confidence scores and processing times across multiple images. Confidence averages were calculated per image based on individual object detection scores, with a threshold set at (0.5). Results indicated variability in confidence averages depending on the number of detected objects and their respective confidence values. As an example, the (SSD) model indicated that image (24) obtained the highest average confidence, whereas image (37) recorded the lowest. On the other hand, (YOLO V.3) indicate that image (3) had the maximum average confidence, while image (34) had perform poorly. The superior average confidence that achieved by (YOLO V.3) compared to the (SSD) model, demonstrating enhanced reliability of detection.

Another important metric is the processing time, which obtained by computing the detection time average for each object inside each image. The (SSD) model achieved better in time, with certain images processing instantly (average time = 0), whereas images (37) and (43) had the highest detection time. In contrast, (YOLO V.3) showed its highest time averages for images (20) and (21), while image (43) indicates the fastest performance. The analysis of the comparison demonstrated that MobileNet (SSD) surpassed (YOLO V.3) in speed, whereas (YOLO V.3) outperform in detection confidence. These inconsistencies arise from architectural differences between the two models; since (SSD) prefers performance (speed), the (YOLO) choose accuracy.

The results underlined the importance of model selection on system effectiveness, especially when trading-off speed against accuracy. MobileNet (SSD) has a fast and high confidence object detection architecture ideal for real time applications whereas (YOLO V.3) robustness in detection led to the lower speed. We validated these models on the (COCO 2017) dataset to demonstrate their applicability to offline and online image analysis. Against (i.e., in comparison with) available methodological alternatives (e.g. OpenCV, CNN, R-CNN), the developed pipeline provided a strong confidence and time metrics analysis. Thus evidencing MobileNet (SSD) utility under practical object detection applications.

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

6. CONCLUSION

Object detection considered as vital technology in computer vision, with various applications in different sectors like healthcare, robotics, autonomous systems, and manufacturing. This study present a comparison of two powerful DL architectures—(YOLOv3) and (SSD)-MobileNet—evaluating their performance in offline image processing and real-time detection environments. In the proposed architecture a seven-step detection pipeline was employed For each model this pipeline includes preprocessing, model loading (using weights from the COCO dataset), feature extraction, NMS, and output evaluated with performance metrics, confidence scoring, and bounding box visualization. Experimental findings indicate important differences in performance: (SSD)-MobileNet enhance processing speeds owing to its effective depth-wise separate convolutions, rendering it more suitable for real-time applications, where (YOLOv3) shows inconsistent accuracy in the recognition of objects and object counting. The main results show that detection confidence and computational efficiency are strongly influenced by object density, being (SSD)-MobileNet preferred for high-speed low-resources conditions. These findings illustrate the speed versus accuracy trade-offs for object recognition systems, providing critical recommendations on trading off model performance to meet application specific requirements.

7. References

[1] A. A. Khan, A. A. Laghari, and S. A. Awan, "Machine Learning in Computer Vision: A Review," *EAI Endorsed Trans. Scalable Inf. Syst.*, vol. 8, no. 32, pp. 1–11, 2021.

[2] D. Patil, A. Poojari, J. Choudhary, and S. Gaglani, "CNN based Traffic Sign Detection and Recognition on Real Time Video," *Int. J. Eng. Res. Technol.*, vol. 9, no. 3, pp. 1–5, 2021.

[3] H. H. Jacobsen, G. Ringstad, Ø. K. Jørstad, M. C. Moe, T. Sandell, and P. K. Eide, "The human visual pathway communicates directly with the subarachnoid space," *Investig. Ophthalmol. Vis. Sci.*, vol. 60, no. 7, pp. 2773–2780, 2019.

[4] D. P. F. Möller, "Machine Learning and Deep Learning," *Adv. Inf. Secur.*, vol. 103, pp. 347–384, 2023.

[5] L. S. Vishnu and S. Rao, "Object Recognition and Object Counting using CNNs," no. 12376, 2017.

[6] C. Anusha and P. S., "Object Detection using Deep Learning," *Int. J. Comput. Appl.*, vol. 182, no. 32, pp. 18–22, 2018.

[7] P. Shah, P. Shah, M. Thakkar, G. By, and P. Sejal Thakkar, "Object Detection & Count in Image," *Int. Res. J. Eng. Technol.*, pp. 3844–3848, 2021.

[8] T. Nguyen, C. Pham, K. Nguyen, and M. Hoai, "Few-Shot Object Counting and Detection," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 13680 LNCS, no. 1, pp. 348–365, 2022.

[9] J. Moon, S. Lim, H. Lee, S. Yu, and K. B. Lee, "Smart Count System Based on Object Detection Using Deep Learning," *Remote Sens.*, vol. 14, no. 15, 2022.

[10] D. P. F. Möller, "Machine Learning and Deep Learning," *Adv. Inf. Secur.*, vol. 103, pp. 347–384, 2023.

[11] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," Sensors (Switzerland), vol. 18, no. 8, pp. 1–29, 2018.

[12] A. Géron, *Hands-on Machine Learning whith Scikit-Learing, Keras and Tensorfow*. 2019.

[13] J. Anderberg and and Nazdar Fathullah., "A machine learning approach to enhance the privacy of customers," *thesis, Digit. Vetenskapliga Ark. Malmö Univ. Fac. Technol. Soc. (TS).*, 2019.

[14] S. Haque and G. Loukas, "Machine Learning Based Prediction versus Human-as-a-Security-Sensor," *Int. J. Artif. Intell. Res.*, vol. 3, no. 1, pp. 11–21, 2018.

[15] X. Long, M. Du, and X. Hu, "Exploring Computer vision in Deep Learning: Object Detection and Semantic Segmentation," pp. 3317–2019, 2019.

Iraqi Journal for Computers and Informatics
Information Technology and Communications University
Vol. 51, No. 2, 2025, pp. 290-306
DOI: https://doi.org/10.25195/ijci.v51i2.631
Print ISSN: 2313-190X, Online ISSN: 2520-4912

[16] A. Chauhan, M. Verma, S. Gupta, V. Srivastava, and A. Kumar, "Object Detection Using Machine Learning," *Int. Res. J. Comput. Sci.*, vol. 07, no. 04, pp. 42–46, 2020.

[17] A. Addapa, P. R. Ramaswamy, and M. K. Kumar, "Object Detection/Recognition Using Machine Learning Techniques in AWS," no. July, 2020.

[18] V. N. Mandhala, D. Bhattacharyya, B. Vamsi, and N. Thirupathi Rao, "Object detection using machine learning for visually impaired people," *Int. J. Curr. Res. Rev.*, vol. 12, no. 20, pp. 157–167, 2020.

[19] T. Nguyen, C. Pham, K. Nguyen, and M. Hoai, "Few-Shot Object Counting and Detection," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 13680 LNCS, no. 1, pp. 348–365, 2022.

[20] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H. "MobileNets: Efficient convolutional neural networks for mobile vision applications".2027. arXiv preprint, arXiv:1704.04861. Available at: https://arxiv.org/abs/1704.04861.

[21] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C. "SSD: Single shot multibox detector'. In: Computer Vision – ECCV 2016". Cham: Springer, pp. 21–37. 2016. doi:10.1007/978-3-319-46448-0_2.

[22] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.C. "MobileNetV2: Inverted residuals and linear bottlenecks". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, pp. 4510–4520.2018.

[23] Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V. and Le, Q.V. "Searching for MobileNetV3". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Piscataway: IEEE, pp. 1314–1324. 2019.

[24] Redmon, J., & Farhadi, A. "YOLOv3: An incremental improvement". 2018. arXiv preprint arXiv:1804.02767. https://arxiv.org/abs/1804.02767.

[25] Wang, C.-Y., Liao, H.-Y. M., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., & Yeh, I.-H. "CSPNet: A new backbone that can enhance learning capability of CNN". Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 390-391. 2020. https://doi.org/10.1109/CVPRW50498.2020.00203.

[26] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. "Microsoft COCO: Common objects in context. European Conference on Computer Vision (ECCV)", 740–755.2014. https://doi.org/10.1007/978-3-319-10602-1_48.