Journal of Real-Time Image Processing (2023) 20:42
https://doi.org/10.1007/s11554-023-01299-3

ORIGINAL RESEARCH PAPER

A novel finetuned YOLOv6 transfer learning model for real-time object detection

Chhaya Gupta[1] · Nasib Singh Gill[1] · Preeti Gulia[1] · Jyotir Moy Chatterjee[2]

Received: 18 October 2022 / Accepted: 25 March 2023 / Published online: 10 April 2023

Abstract

Object detection and object recognition are the most important applications of computer vision. To pursue the task of object detection efficiently, a model with higher detection accuracy is required. Increasing the detection accuracy of the model increases the model's size and computation cost. Therefore, it becomes a challenge to use deep learning in embedded environments. To overcome this problem, the current research suggests a transfer-learning-based model for real-time object detection that enhances the YOLO algorithm's effectiveness. The model utilizes YOLOv6 as a baseline model. This study proposes a pruning and finetuning algorithm as well as a transfer learning algorithm for enhancing the proposed model's efficiency in terms of detection accuracy and inference speed. This paper also focuses on how the proposed model will be able to identify all objects (indoor as well as outdoor) in a scene and provides a voice output to warn the user about nearby and faraway objects. To receive the audio feedback, Google Text-to-Speech (gTTs) library is used. The model is trained on the MS-COCO dataset. The proposed model is compared with the Tensorflow Single Shot Detector model, Faster RCNN model, Mask RCNN model, YOLOv4, and baseline YOLOv6 model. After pruning the YOLOv6 baseline model by 30%, 40%, and 50%, the finetuned YOLOv6 framework hits 37.8% higher average precision (AP) with 1235 frames per second (FPS).

Keywords Tensorflow · SSD · Faster RCNN · Mask RCNN · gTTs · Transfer Learning · Pruning

1 Introduction

In the technical world, computers are used to categorize various objects. For this purpose, computers use object classification and recognition with deep learning and deep neural networks. Object classification and recognition are challenging concepts in the field of computer vision. Compared to image classification, object detection is more complex to perform as it counts on both, image classification as well as image localization [1]. It is a difficult task to detect and recognize each object and to estimate with which class the object belongs where different objects are placed at different positions. Various object detection algorithms like Faster RCNN, Mask RCNN, YOLOv3, and others are continuously striving to strike a balance between inference speed and accuracy.

Reducing the scale of object detection models has drawn a lot of attention in the last decade. Hence, it is a big problem for object detection models to maintain the balance between model accuracy and its inference speed. This paper focuses on this unsolved problem by providing a quick and precise object detection approach based on transfer learning to enhance the inference speed of object detection algorithms and analyze the real-time processing of images. The proposed model utilizes the baseline YOLOv6 model and helps in reducing the model size as well as improving the inference speed and accuracy. Various factors that motivate to refurnish the YOLO framework are listed below:

- Past research work paid less attention to domain-specific characteristics like label assignment and loss functions.
- Finetuning the parameters was never a thought in past research work. Finetuning hyperparameters helps in increasing the inference speed as well as the accuracy of the entire network.

✉ Jyotir Moy Chatterjee
jyotirmoy.chatterjee.cse@gmail.com

[1] Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak, India

[2] Department of IT, Lord Buddha Education Foundation(Asia Pacific University), Kathmandu, Nepal

Due to these research gaps, the study proposes a pruned and finetuned YOLOv6 object detection model with transfer learning that improves the accuracy as well as the inference speed of the model. The model works on a teacher-student network in which the student network is pruned and finetuned and then the tuned parameters are used in the teacher network for training the framework. The proposed framework achieved an average precision of 37.8% with 1235 FPS. The main objectives and contributions of the paper are summarised as follows:

- The baseline YOLOv6 model is compressed and finetuned with the help of the proposed hidden layer pruning algorithm. The hidden layer pruning algorithm helps in adjusting the network depth. The proposed algorithm also finetunes the network with a reduced learning rate to restore any precision values which are dropped during the pruning process.
- To improve the detection accuracy of the finetuned YOLOv6 model, a transfer learning algorithm has been proposed and implemented. The proposed transfer learning algorithm is slightly different from the traditional one.

The rest of the paper is organized into the following sections: Sect. 2 provides the literature survey of object detection algorithms and transfer learning, Sect. 3 describes the proposed model, Sect. 4 discusses the experiment, performance analysis, and, comparison with various other object detection models and how the proposed model can be made useful for visually impaired people and Sect. 5 concludes the paper.

2 Related work

This section presents a review of object detection, and a review of YOLO models and discusses the architecture of YOLOv6, and research work related to pruning.

a. Object Detection

Object detection is a computer vision concept in which an object in an image or video is identified as well as localized. A new 3D object detector with depth estimation has been proposed for camera-based Bird's Eye View (BEV) [2]. For better 3D scene understanding, a novel Lidar-based 3D object detection model has been proposed that works on occluded images [3]. Object detection is used for collecting real-time traffic information with the help of UAVs (Unmanned Aerial Vehicles) [4]. A new real-time small object detection (RSOD) based on YOLOv3 has been proposed for extracting features from very small images collected via UAVs [4]. Khoshboresh et al. [5] carried out the first attempt to detect ground vehicles via improved deep learning methodology. In this study, Gaussian-Bernoulli Restricted BoltzBoltzmannine was combined with a deep learning model to improve the performance of the model. An improved YOLOv4-based object detection model for handling the issue of low detection accuracy in complex machinery swarm operations has been proposed [6]. A new real-time 3D object detection convolutional neural network based on YOLOv5 has been proposed in which YOLOv5 anchor boxes have been replaced by hybrid ones [7]. A novel smartphone-based object detection architecture for portable systems based on CNN has been proposed [8]. Another viewpoint-based memory mechanism has been proposed for improving the detection accuracy of videos in real time [9]. The mechanism achieved a 20.7% object localization rate (OLR). A novel traffic sign detection based on YOLOv3 has been proposed to make an addition to the application of object detection in daily routine [10]. A YOLOv4-tiny neural network has been proposed for social distancing tracking during COVID times [11]. The network is based on Bird's eye view mechanism.

b. YOLO and YOLOv6

YOLO was proposed by Joseph Redmon in 2016 [12]. YOLO has many versions so far and it is improving continuously each time. Wei Sun et al. [4] proposed a Real-time Small Object Detection (RSOD) algorithm based on yolov3 that was trained on Visdrone-DET2018 and UAVDT datasets and achieved mAP of 43.3% and 52.7%, respectively. A. Mauri et al. [7] introduced a yolov5-based real-time 3D object detection CNN to be used in sustainable transportation applications both for road and rail contexts. Nikkhat Bushra et al. [13] proposed a system based on Yolov5 which helped in identifying weapons held by a person and the system was also capable of recognizing the faces of the suspicious user. Ruiyang Xia et al. [14] profounded a novel real-time object detector namely, Transformers Only Look Once (TOLO) which is based on Convolutional Neural Network, different Lite Transformer Heads, and Feature Fusion Neck. Masum Shah Junayed et al. [15]proposed a novel lightweight model that makes use of depth- and point-wise separable convolutions to categorize objects in images. Mais R.Kadhim et al. [16] proposed a blind assistive system based on YOLO and OpenCV python library for real-time object detection. Fahad Ashiq et al. [17] proposed a navigation system in real time with output as an automated voice that helps visually impaired people. The system is based on a deep convolutional neural network that achieved an accuracy of 83.3%. Chhaya Gupta et al. [18] proposed a hybrid approach SSDT (Smart Social Distancing Tracker) which is a combination of YOLOv4 merged with MF-SORT, Kalman Filter, and brute force feature matching technique to distinguish people from background and achieved an FPS of 24 on MOT dataset.

Chhaya Gupta et al. [19] proposed a two-stage face mask detector namely Coronamask that classifies whether a person is wearing a mask or not and achieves an accuracy of 95%. Yuxuan Cai et al. [20] proposed a pruning scheme for any kernel size which is GPU/CPU collaborative and achieved an mAP of 49.0 and FPS of 17 and is based on Yolov4. Chenglizhao Chen et al.[21] proposed a novel spatiotemporal network on real-time Video Salient Object Detection and achieves an FPS of 50 in real-time applications. Most of the work has been carried out with two-stage detectors, as two-stage detectors perform better but they have their disadvantages. A two-stage detector is not useful in real-time object detection because of its low inference speed.

YOLOv6 is a single-stage detector having an efficient design and high performance. YOLOv6 outperforms all the previous versions of YOLO in accuracy as well as inference speed. This section provides an architectural comparison between YOLO and YOLOv6 to understand what's new in YOLOv6. YOLO models have a backbone-neck architecture. Figure 1 provides the architecture of YOLO models so far (*What's New in YOLOv6?* n.d.). YOLO models take an image as input and pass it to different convolutional layers. These convolutional layers act as the backbone for YOLO models. The features extracted in the backbone layers are passed to the neck convolutional layers. These layers extract more features. The neck features are passed through three different heads namely, predict objects, class, and box regression.

YOLOv6 redesigns the YOLO backbone and neck designs and renames them as EfficientRep backbone and Rep-PAN neck. YOLOv6 uses decoupled head. Decoupled head means that the network has additional layers in the head section which helps in increasing the performance. The neck features are directly passed to the decoupled head section where tasks like objectness, classification, and regressions take place at once. This helps YOLOv6 in increasing its performance. Figure 2 presents the backbone-neck architecture of YOLOv6 and Fig. 3 provides the decoupled head architecture of YOLOv6 [23]. In the EfficientRep backbone design, the convolutional layers have been replaced with RepConv layers. The first RepConv layer transforms the data to align the channel dimensions. YOLO models till version 5 work on the CSP network but YOLOv6 works on the RepBlock network.

c. Pruning

Pruning is a technique that minimizes a network's redundancy based on the feature score. This creates a network with lower dimensionality than the baseline network, which needs less processing. Pruning is a 3-step process namely, sparsity learning, pruning, and fine-tuning. Pruning is mainly based on sparsity learning networks. In pruning, unwanted parameters are determined based on their feature scores and they are removed. This process helps in reducing the dimensionality of any neural network by reducing the number of parameters. After reducing the unwanted parameters, the remaining parameters are finetuned to retrain the network. Pruning is categorized as structured and unstructured based on the method of removing unwanted parameters. Unstructured pruning creates a sparse convolution structure by evaluating each parameter's significance separately and removing those that are unneeded. But unstructured pruning has the limitation of requiring certain libraries and hardware support. On the other hand, structured pruning judges a group of

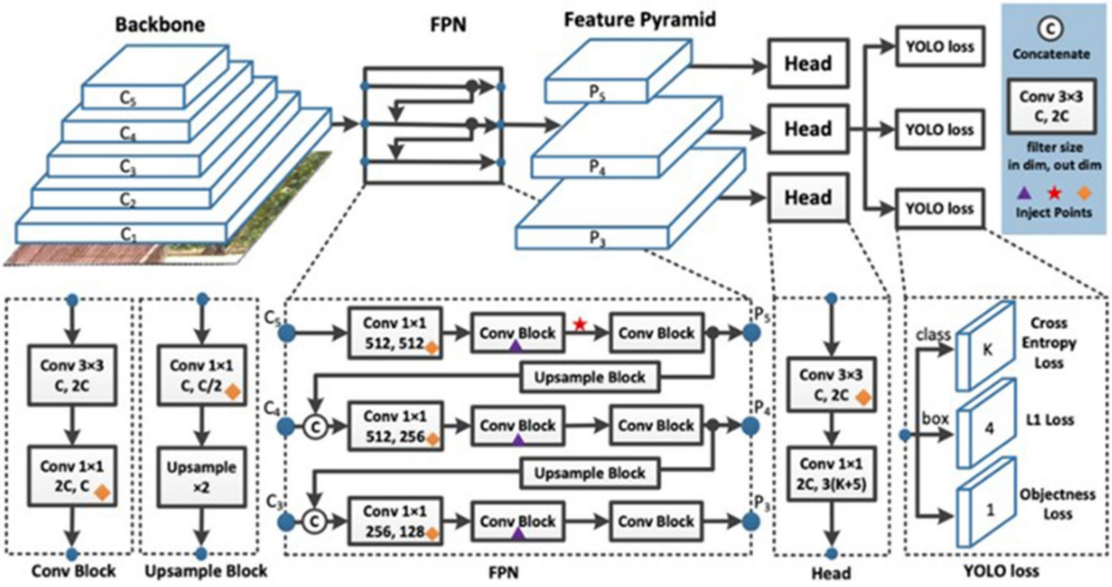


Fig. 1 YOLO architecture (*What's New in YOLOv6?* n.d.) [22]

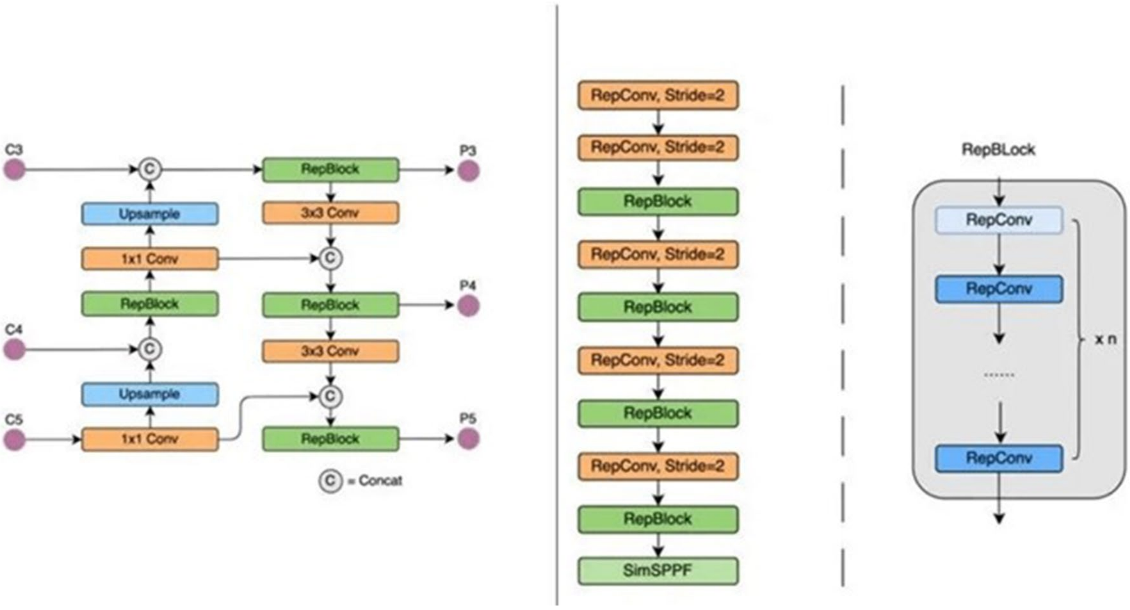


Fig. 2 Backbone-Neck architecture of YOLOv6 [23]

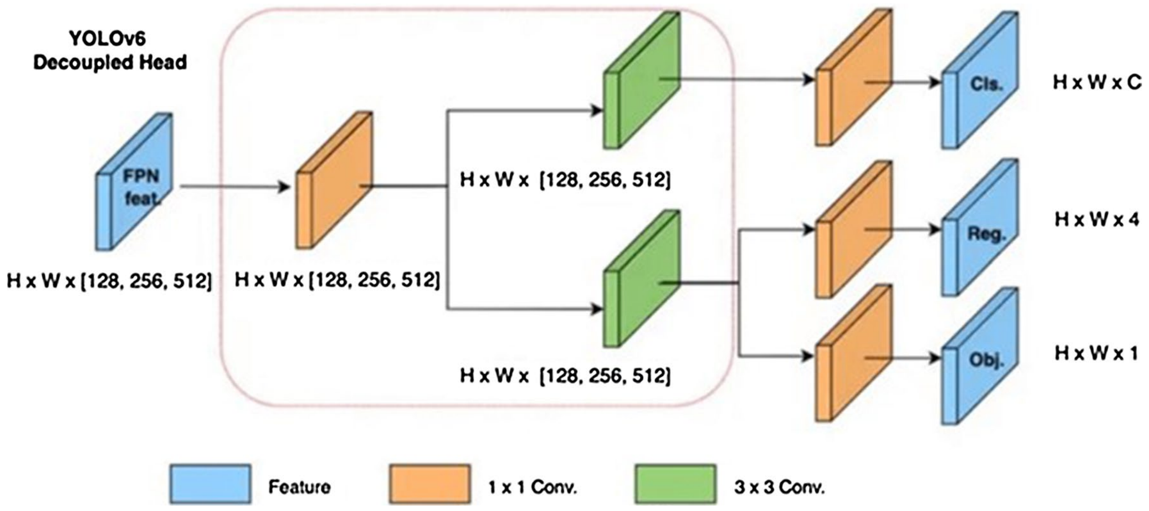


Fig. 3 Decoupled head architecture of YOLOv6 [23]

parameters and removes unnecessary parameters. Structured pruning has a high compression ratio and it reduces most of the parameters in the fully connected layer and lightens up the network. It helps in judging the importance of each parameter based on its feature score. It does not require any explicit specific library.

In this paper, a hidden layer pruning algorithm is proposed which makes YOLOv6 a light-weighted network by reducing a significant number of parameters and decreasing the network depth. After pruning the model, the detection accuracy diminishes. To enhance the detection accuracy, a transfer learning algorithm was implemented with the finetuned YOLOv6 network. Instead of using the traditional transfer learning algorithm, a new transfer learning algorithm has been proposed. In the pruning process, the number of parameters of the model and the model's depth is reduced. The complete algorithm is explained further in this paper.

A lot of work has been done in the field for real-time object detection and it has shown that object detection can be used in different applications. Despite the work done till now, there are a lot of limitations and hindrances that need to be investigated:

- Most of the work is carried out on old object detection models and pruning techniques have been just applied to them.
- The researchers have used pre-trained models via transfer learning for producing high-accuracy results and it takes a lot of effort in changing the architecture of a pre-trained model and add something new to it.

To overcome the limitations, a framework based on finetuned YOLOv6 with transfer learning has been proposed. The proposed framework works on real-time data and is being trained with the MS-COCO dataset and is tested on real-time data. In this paper:

- The Baseline YOLOv6 model has been pruned and finetuned as per the proposed hidden layer pruning algorithm to make the framework a better and more efficient one. The proposed hidden layer pruning algorithm also helps in reducing the network depth of the pruned YOLOv6 when compared with baseline YOLOv6.
- For further improving the detection accuracy, a new transfer learning algorithm is proposed and is implemented on the pruned and finetuned YOLOv6 network.
- As of now, YOLOv6 has not been used for object detection, hence in this study, the baseline YOLOv6 model has been used for real-time object detection and is also pruned to increase the detection accuracy without any precision loss.
- MS-COCO dataset has been used for training the framework. For testing purposes, real-time data has been used.
- The framework may also be used as a guide for a visually impaired person. The proposed framework works with a webcam and provides results along with voice output. A comparative study of the proposed model with other object detection models like the Single Shot Detector model, Faster RCNN model, Mask RCNN model, YOLOv4, and baseline YOLOv6 is presented in terms of precision, recall, and mAP.

3 Proposed real-time object detection model

The proposed model works on a student–teacher network. The framework is trained using the parameters that have been finetuned from the student network, and the tuned parameters are used in the teacher network. The architecture of the framework is divided into two parts, the first part is the teacher network that consists of the baseline Yolov6 model and the other part is the student network consisting of the proposed pruned and finetuned YOLOv6. The pruning is done with 30%, 40%, and 50% sparsity and it was observed that the model outperforms with the sparsity of 30%. To restore the dropped precision values, finetuning is also implemented with pruning at a very low learning rate. To improve the detection accuracy as well as inference speed of the overall framework, transfer learning has been implemented. The baseline model YOLOv6 is trained to obtain a teacher network first and afterward, the trained weights and biases are used for finetuning Yolov6 which in turn is used to train the student network. The complete architecture of the proposed framework is shown in Fig. 4. The figure shows the baseline YOLOv6 model and pruned YOLOv6 model. The baseline model has several layers which were reduced when the pruned and finetuned weights file yolov6.pt was modified and used for training the baseline YOLOv6 model. Migration learning is also used with pruned and finetuned YOLOv6 model. The baseline model is trained on the MS-COCO dataset. The proposed model achieves better detection accuracy while reducing the number of parameters and hence can be considered an efficient model for real-time object detection. The rest of this section describes all the modules used in the proposed framework.

3.1 YOLOv6 algorithm

YOLOv6 redesigns the YOLO backbone and neck designs and renames them as EfficientRep backbone and Rep-PAN neck. YOLOv6 uses decoupled head. Decoupled head means that the network has additional layers in the head section which helps in increasing the performance. The neck features are directly passed to the decoupled head section where tasks like objectness, classification, and regressions take place at once. The YOLOv6 model consists of three parts namely, backbone, Neck, and Decoupled Head. The first part is the Backbone of the model. This part impacts the efficiency and effectiveness of the model. It is a single-path network that helps in high parallelism and less memory usage which leads to higher efficiency. For this purpose, RepBlock is used during the training phase. RepBlock is converted to 3 × 3 convolutional layers with the help of the ReLU activation function. 3 × 3 convolutional layers are chosen due to their high computational density. Cross Stage Partial connections are also deployed to boost the performance of the network during training. The second part is the Neck. Unlike, Yolov5, CSP connections are removed with RepBlock. The third part is the Decoupled Head. A hybrid channel strategy is used to build the decoupled head. Baseline Yolov6 has a high computation cost and has a high number of parameters, therefore, to achieve higher accuracy and efficiency, a hidden layer pruning algorithm has been proposed.

YOLOv6 uses reparameterized VGG blocks with skip connections during the training phase. In the inference phase, the RepBlocks are converted to 3 × 3 convolutional layers known as Repconv blocks.

YOLOv6 has two loss functions namely, Varifocal Loss and Distribution Focal Loss with SIoU or GIoU. VarifThe varifocal function is used for classification purposes and Distribution Focal Loss is used for Box regression purposes.

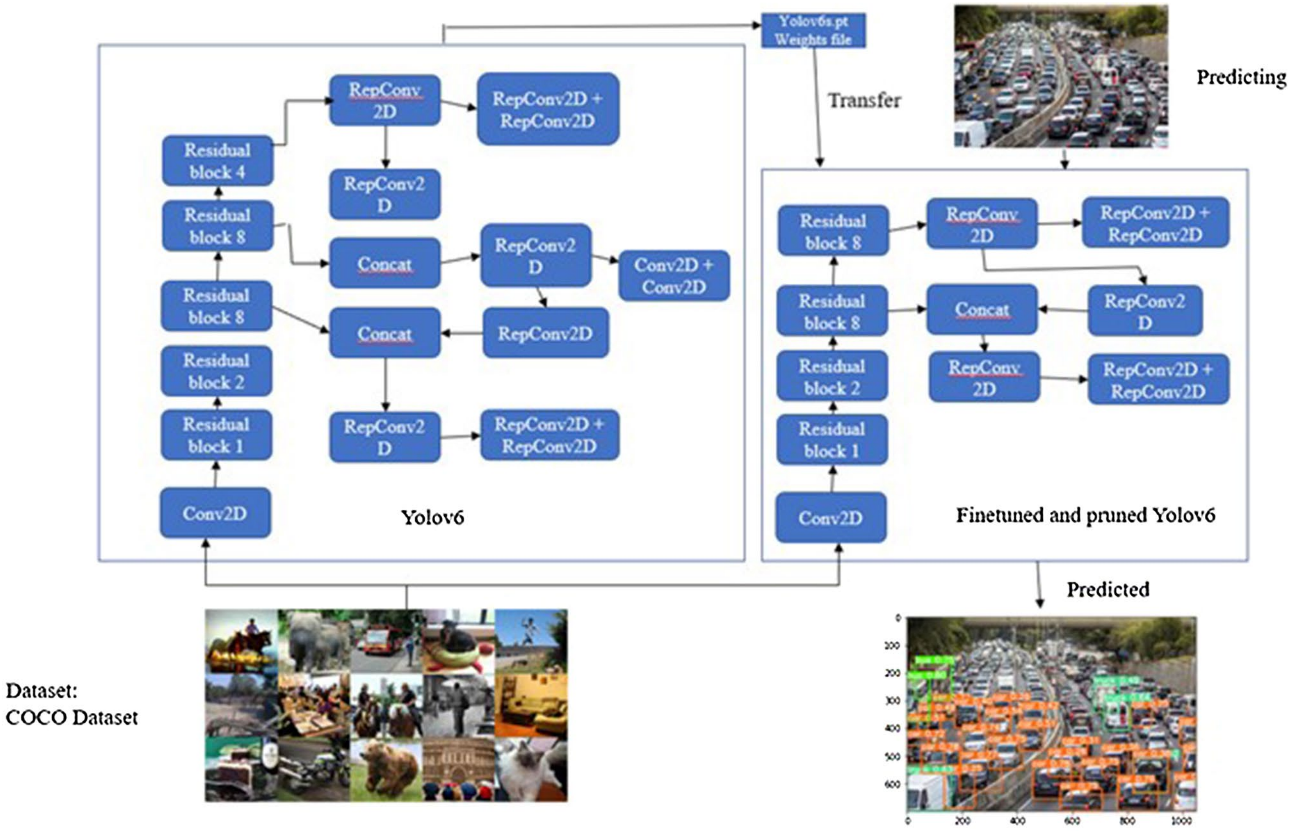


Fig. 4 Structure of overall framework based on transfer learning, the left side shows the baseline YOLOv6 model, and the right side shows the pruned and finetuned YOLOv6 model

Varifocal loss (VFL) is a forked version of Focal loss. Focal loss (FL) helps in handling class imbalance by multiplying the predicted value with the power of gamma as shown in Eq. 1. Varifocal loss uses this for negative sample loss calculation only. For a sample loss calculation, VFL uses Binary Cross Entropy (BCE) loss [24]. VFL is shown in Eq. 2.

$$\mathrm{FL}(b,c)=\begin{cases}-\alpha(1-b)^{\gamma}\log(b)\mathit{ifc}=1\\-(1-\alpha)b^{\gamma}\log(1-b)\mathit{otherwise}\end{cases}\tag{1}$$

where c ϵ $\{\pm 1\}$ is the ground truth value, b ϵ [0,1] is the predicted probability of the foreground class, $(1-b)^{\gamma}$ is the modulating factor for the foreground class and b^{γ} is a modulating factor for the background class.

$$\mathrm{VFL}(b,c)=\begin{cases}-c(c\log(b)+(1-c)\log(1-b))c>0\\-\alpha b^{\gamma}\log(1-b)c=0\end{cases}\tag{2}$$

where b is the predicted IoU-Aware Classification Score (IACS) and c is the target score. If there is a foreground point, then, c for its ground truth class is set as Intersection over Union (IoU) between generated bounding box and its ground truth. If there is a background point, the target score c is 0 for all classes.

Distribution Focal Loss (DFL) is used for calculating box regression loss. It helps in optimizing the distribution of box boundaries [25]. This method relies on the offset values from the four sides of a bounding box and sets them as regression target 'x'. The main objective of DFL is to maximize the probabilities of values around the target x as shown in Eq. 3.

$$\mathrm{DFL}\left(D_i,D_{i+1}\right)=-\left(\left(x_{i+1}-x\right)\log\left(D_i\right)+\left(x-x_i\right)\log\left(D_{i+1}\right)\right)\tag{3}$$

where D_i and D_{i+1} are probabilities of x_i and x_i respectively. For simplification purposes, D_i and D_{i+1} are used. D_i and D_{i+1} are calculated as shown in Eq. 4.

$$D_i=\frac{x_{i+1}-x}{x_{i+1}-x_i},\ D_{i+1}=\frac{x-x_i}{x_{i+1}-x_i}\tag{4}$$

3.2 Pruned and finetuned Yolov6

Pruning means when the weights of the model are reduced to some extent so that model can perform well in challenging environments as well. It hits the layer structure of a network and reduces the scale of the model[26]. The main objective of pruning is to get rid of those filters that are not important in the evaluation of the overall performance of the network. The performance of each layer is evaluated, and the one having the least effect on the model's detection accuracy is chosen and trimmed before the layer structure of the framework is deleted. This is done iteratively until the increased validation loss had passed the "post-pruned threshold". The post-pruned threshold is the ratio between pruned model's validation loss and the validation loss of the complete model. After this, the model is retrained to bring the validation loss down below the "post-retrained threshold". The post-retrained threshold is the ratio between recovered validation loss to validation loss of the complete model. The algorithm stops when pruned model's validation loss exceeds the post-pruned threshold even after re-training the model. This algorithm prunes one layer at a time and therefore, it is called single-layer pruning. To determine which layer to prune, the model's validation loss is calculated after one of its layers has been modified. Modifying a layer means the removal of unwanted features from the layer. Unwanted features are determined by calculating the Sum Absolute Weights (SAW) of each feature. The features with the lowest SAW are pruned.

The processing time is too long to reach real-time performance. Hence, to make the proposed object detection

Table 1 Network depth comparison between baseline Yolov6 and pruned Yolov6

Model	Backbone: CSP1_X			Neck: CSP2_X				
	1st	2nd	3rd	1st	2nd	3rd	4th	5th
Baseline Yolov6	CSP1_1	CSP1_3	CSP1_3	CSP2_2	CSP2_2	CSP2_2	CSP2_2	CSP2_2
Pruned Yolov6	CSP1_2	CSP1_6	CSP1_6	CSP2_1	CSP2_1	CSP2_1	CSP2_1	CSP2_1

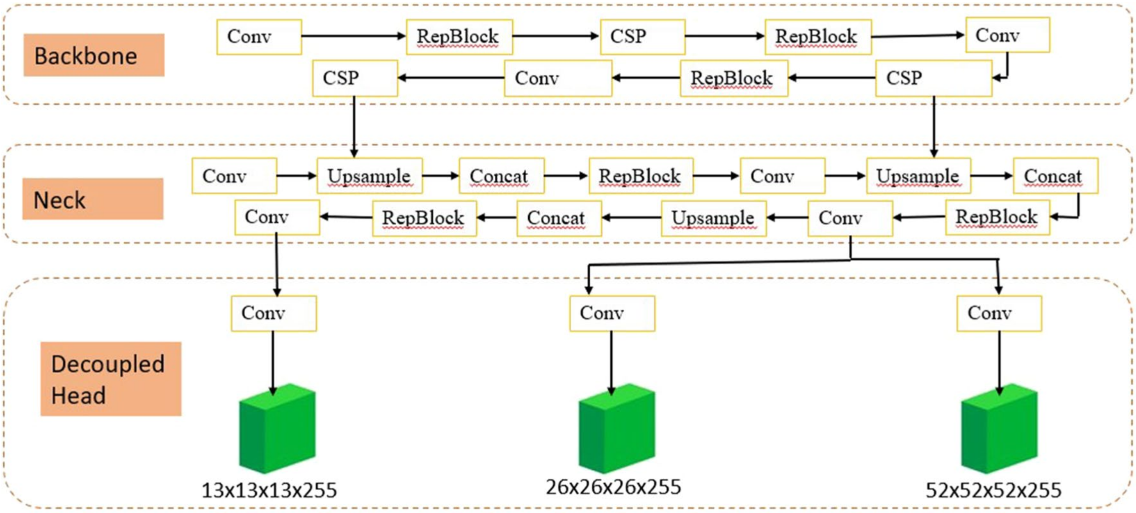


Fig. 5 Architecture of pruned Yolov6

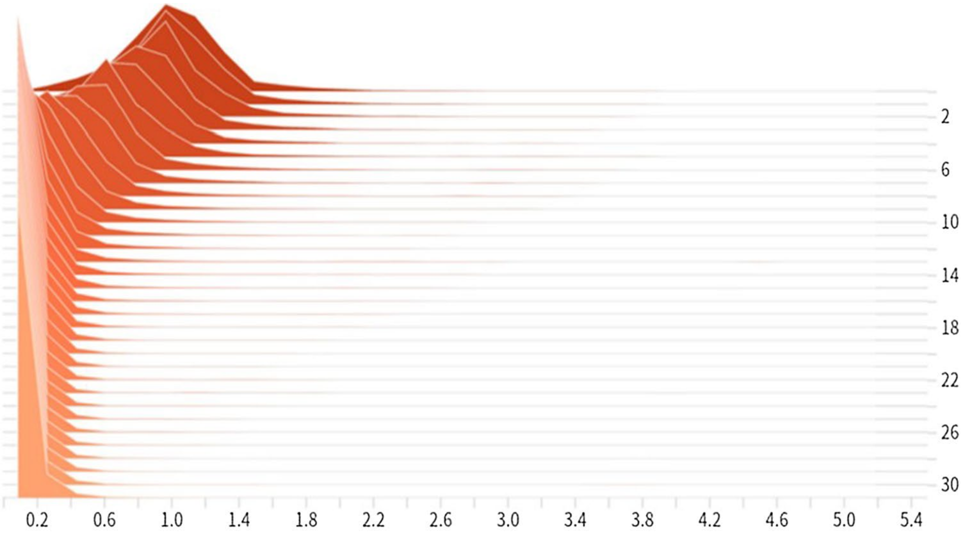


Fig. 6 Batch normalisation layer coefficient ɤ showing pruning process progress during the training phase

model more accurate and efficient, improvements are made to the baseline YOLOv6 model. To compress the model and strengthen the inference speed, YOLOv6 is pruned. On the baseline YOLOv6 model, several experiments have been conducted for the model pruning portion. After examining every step of YOLOv6, from feature value extraction to result in prediction, three values are evaluated namely, time t, parameters h, and accuracy c, and the formula is depicted in Eq. 5.

$$\text{Total}_i = t_i + h_i + c_i, i \in (1, 2, 3) \tag{5}$$

The proposed pruning and finetuning algorithms are summarised in Algorithm 1.

Algorithm 1: Pruning and finetuning the Framework.

```
Input:
$M_{in}$: height of hidden input layer
$W_{in}$: width of the hidden input layer
$M_{out}$: height of hidden output layer
$W_{out}$: width of the hidden output layer
$t_i$: time delay in the hidden layer
$h_i$: parameter in the hidden layer
$c_i$: accuracy of hidden layer
$lr_i$: learning rate parameter
optim: optimizer, value is 1 for "Adam"
$layers_i$: layers of the framework
Output:
$Total_i$: total evaluation of hidden layer
Procedure:
Obtain the baseline Yolov6 model with 13x13, 26x26, and 52x52 layers and calculate SAW for each filter in each layer. Filters with the lowest SAW are taken forward for pruning.
Train the model and calculate the accuracy $c_i$ of each model.
Evaluate the testing results.
if $M_{in} == 13$ and $W_{in} == 13$ then
    load $t_1$, $h_1$
elif $M_{in} == 26$ and $W_{in} == 26$ then
load $t_2$, $h_2$
elif $M_{in} == 52$ and $W_{in} == 52$ then
    load $t_3$, $h_3$
end if
if $M_{out} == 13$ and $W_{out} == 13$ and $layers_i == 75$ then
$Total_i = t_1+h_1+c_1$
optim = 1
$lr_i = 0.0032$
elif $M_{out} == 26$ and $W_{out} == 26$ and $layers_i == 75$ then
$Total_i = t_2+h_2+c_2$
optim = 1
$lr_i = 0.0032$
elif $M_{out} == 52$ and $W_{out} == 52$ and $layers_i == 75$ then
$Total_i = t_3+h_3+c_3$
optim = 1
$lr_i = 0.0032$
end if
return max ($Total_i$, optim, $lr_i$)
```

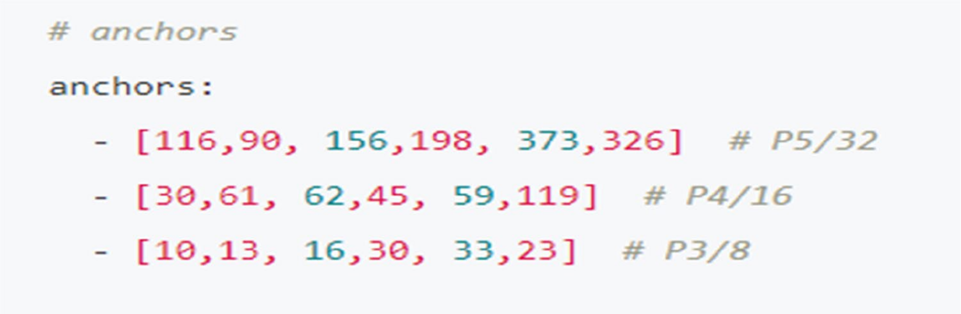
For the pruning process, the baseline YOLOv6 model with 13 × 13, 26 × 26, and 52 × 52 hidden layers is obtained. The model is trained, and accuracy is calculated. The baseline YOLOv6 is tested to evaluate the delay time in the execution of three hidden layers and parameters used. The optimizer function is changed from SGD to Adam for better results during the pruning process. The learning rate (lr) is changed from 0.01 to 0.0032 for finetuning the framework. Finetuning helps in restoring the dropped precision values during pruning. The values time delay t, parameter h, and accuracy c are obtained to decide which hidden layer to select for pruning.

The proposed pruning algorithm reduces the number of hidden 3 × 3 convolutional layers to only one layer. Pruning also helps in adjusting the network depth and network width. In this study, hidden layer pruning is proposed to control the number of residual components in the CSP module and helps in adjusting the network depth. The comparison in baseline Yolov6 network depth and pruned Yolov6 model depth is shown in Table 1. It can be seen from Table 1 that pruned Yolov6 model has different CSP modules in Backbone and Neck networks. In the backbone network of pruned Yolov6, the first CSP1 module has two residual components, and the second and third CSP1 modules have six residual components. In the neck network, the five CSP2 modules have just one residual component. Therefore, using the algorithm of hidden layer pruning, the size of the Yolov6 model can be compressed which results in a lightweight model ensuring better detection accuracy. Pruning drops the precision values, hence, to restore those values, finetuning is also proposed along with the pruning algorithm. Finetuning removes the fully connected nodes where actual class label predictions are made and replace them with newly initialized fully connected nodes. Finetuning also freezes earlier convolutional layers in the network ensuring that any robust feature learned by convolutional layers is not destroyed. After freezing the earlier convolutional layers, newly initialized fully connected layers are trained. Finally, all the frozen convolutional layers are unfrozen. The pruned Yolov6 architecture is shown in Fig. 5. The pruning process progress during the training phase is shown with the help of the Batch Normalisation coefficient ϒ (gamma) in Fig. 6 using the run logs. The gamma coefficient slowly approaches 0 but does not reach 0 completely. From the 14th epoch, the change in coefficient is the same, till the 30th epoch. To make the gamma coefficient closer to 0, an attempt was made to prune the network to 100 epochs, but the results were the same as that of the 14th epoch. For compressing the network width, convolutional kernel pruning can be used that helps to control the convolutional kernels in the convolutional layer structure. Convolutional kernel pruning is left for future work.

3.3 Customizing anchor boxes

Classification of the object is done using the anchor box[27]. The ground-truth box is unified to a given size by its fixed width and height. The anticipated box absorbs knowledge from the anchor box, keeps the weights and biases, but ultimately transforms it into the ground-truth box. The proposed framework finetunes and prunes the Yolov6 baseline model

Fig. 7 Customise anchor boxes in configuration files



```
# anchors
anchors:
  - [116,90, 156,198, 373,326]  # P5/32
  - [30,61, 62,45, 59,119]  # P4/16
  - [10,13, 16,30, 33,23]  # P3/8
```

and hence anchor boxes are required to be changed accordingly. For this purpose, the anchor boxes may be changed in the configuration file of the baseline model as shown in Fig. 7. Changing the configuration file for anchor boxes is not a problem in YOLOv6. Yolov6 automatically learns about the anchor box distributions based on the training set and hence customizes them.

3.4 Transfer learning

As the framework has been pruned and finetuned, the detection accuracy of the model must be decreased. Transfer learning technique is used for increasing the detection accuracy. In this study, a new algorithm has been proposed for transfer learning which is slightly different from the traditional one. The baseline YOLOv6 model's performance and generalizability are used to boost the detection accuracy of the pruned YOLOv6 model. Algorithm 2 summarizes the new transfer learning algorithm for the proposed model.

Algorithm 2: Modified transfer learning algorithm for the framework

```
Input:
$A_{train}$: training dataset
$W_T$: weight of baseline network
$W_S$: weight of the finetuned network
$B_T$: bias of baseline network
$B_S$: bias of finetuned network
$L_T$: count of layers in baseline network
$L_S$: count of layers in the finetuned network
$Channel_T$: layers in baseline network
$Channel_S$: layers in the finetuned network
Output:
$P_{pre}$: pre-trained parameters of the finetuned network framework
Procedure:
Obtain the parameters which have been saved while training the baseline model
Implement training for $A_{train}$ and calculate $W_T$ and $B_T$
for j = 1; j < $L_T$; j++ then
    for x = 1; x < $L_s$; x++ then
        if $Channel_S$ = = $Channel_T$ then
            $B_s = W_T$
        End if
    End for
End for
Enact the finetuned network and evaluate the predicted results $P_{pre}$
Return $P_{pre}$
```

During the transfer learning process, the student network (pruned YOLOv6) is initialized. The weights and biases of some layers of the teacher network (baseline YOLOv6) are applied to the student network. These are used as pre-trained parameters for the model. Traditionally, in transfer learning algorithms the weights of the teacher network are applied to weights of the student network and biases of the teacher network are applied to biases of the student network. But, in this proposed transfer learning algorithm, for each layer in the baseline and proposed model, the channels of the baseline and proposed model are checked. If they are the same, then, the weights of the baseline network are applied as biases to finetuned model. The weights which are applied to the finetuned model are used as pre-trained weights as shown in Eq. 6.

$$\sum_{i=1}^{N} B_s = \sum_{i=1}^{N} W_T, N = L_s \tag{6}$$

This strategy not only helps in increasing the detection accuracy of the overall model but also helps in improving the performance and efficiency of the model.

4 Experimental analysis

In this study, finetuned Yolov6 is trained using the MS-COCO training and validation datasets[28]. MS-COCO dataset consists of more than 200 k images of 80 classes. The training set consists of 82783 images and the validation set consists of 40775 images. For simplification purposes, 17 classes have been used for training various object detection models including the proposed model [29]. The proposed model is compared with different object detection models and is also assessed using a real-time environment. The proposed model is used with a webcam and is combined with gTTs (Google Text-to-Speech) to produce automated voice output and may be considered as a guide for visually impaired people in the future.

Table 2 Recall values of different models compared with the proposed framework on 17 different classes of the MS-COCO dataset

Models	Classes																
	Bike	Bird	Car	Cow	Bottle	Bus	Cat	Dog	Person	Chair	Table	Horse	Motorbike	Plant	Sofa	Bed	Cell phone
Faster R-CNN	23.3	30.1	11.23	7.8	6.5	35.5	28.8	23.3	32.2	10.4	15.3	34.4	29.9	0.6	12.3	16.7	27.5
SSD	20.1	25.6	10.6	7.9	9.5	34.6	30.2	21.2	30.2	9.8	11.3	30.4	25.4	1.7	11.6	13.3	22.5
Mask R-CNN	24.1	29.9	12.3	19.4	15.6	32.3	31.2	30.9	33.2	24.5	20.8	30.2	32.3	9.8	24.4	22.2	34.4
YOLOv4	25.6	34.4	33.4	43.2	34.5	32.3	34.7	35.6	34.9	36.5	34.6	35.7	41.4	34.2	45.4	34.6	38.6
Baseline YOLOv6	74.5	56.7	48.9	54.6	56.7	67.7	54.4	50.2	67.4	67.5	54.7	56.7	54.5	57.7	67.4	54.7	56.7
Finetuned Framework	79.8	59.8	60.8	71.9	63.3	72.3	67.7	71.1	78.6	70.2	67.6	67.8	68.9	67.5	73.4	77.6	78.9

Table 3 Precision values of different models when compared with the proposed framework on 17 classes of the MS-COCO dataset

Models	Classes																
	Bike	Bird	Car	Cow	Bottle	Bus	Cat	Dog	Person	Chair	Table	Horse	Motorbike	Plant	Sofa	Bed	Cell phone
Faster R-CNN	80.2	76.5	74.5	67.8	78.5	64.5	70.9	78.6	82.2	87.5	90.4	64.4	79.6	80.6	63.4	78.9	90.5
SSD	78.9	82.2	80.9	76.7	68.7	74.5	60.5	67.8	70.8	78.9	81.8	87.6	94.5	90.3	65.7	64.8	78.6
Mask R-CNN	95.4	78.9	65.6	86.7	86.7	84.5	61.2	70.9	83.6	94.5	60.2	70.4	82.5	90.8	64.4	72.3	84.3
YOLOv4	96.7	67.9	87.7	84.5	97.6	87.6	64.5	74.6	84.5	97.5	64.6	73.8	84.3	94.7	62.3	73.4	80.1
Baseline YOLOv6	96.8	78.6	98.7	90.1	93.4	97.6	64.4	70.2	67.4	75.4	87.6	66.7	96.7	87.6	77.5	65.4	92.1
Finetuned Framework	98.7	89.1	65.8	95.6	99.7	76.5	67.7	86.5	82.3	76.5	94.5	99.8	85.6	65.1	78.5	89.9	78.9

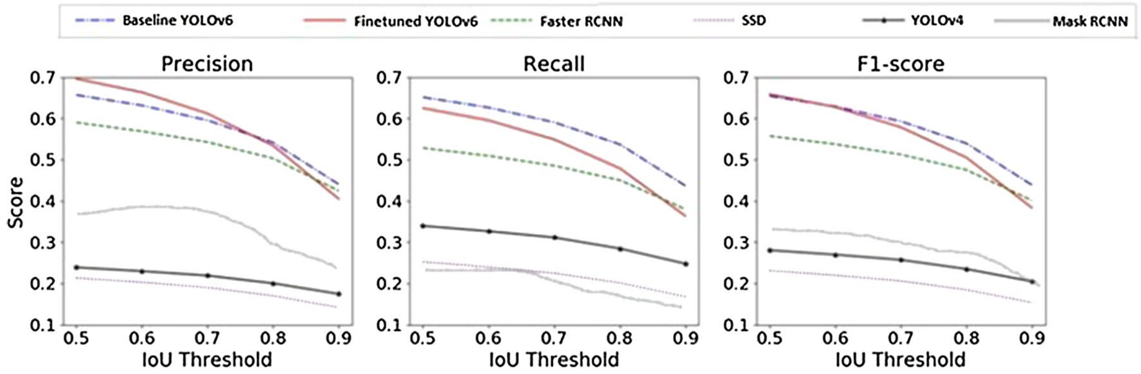


Fig. 8 Performance of different object detection models compared with the proposed framework at different IoU threshold values

Fig. 9 Recall curves for various object detection models when compared with the proposed framework

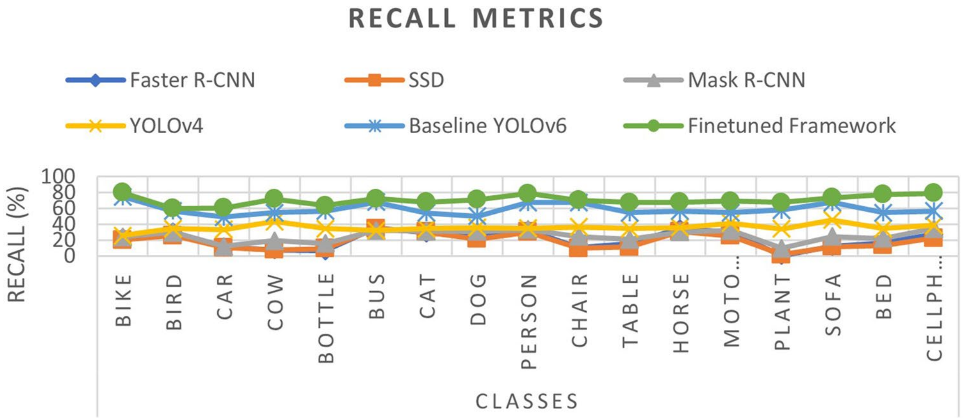
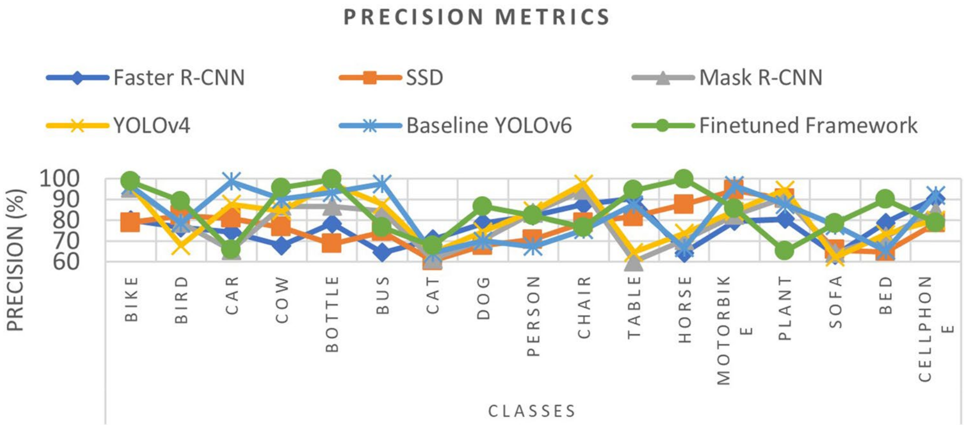


Fig. 10 Precision curve for various object detection models when compared with the proposed framework



4.1 Performance evaluation metric used

Object detection algorithms are compared in-depth during the experiment and also computed the recall and precision rates. The metrics used are:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{TN}} \tag{8}$$

$$\textit{where}\text{TP} = \text{TruePositive}$$

$$\text{TN} = \text{TrueNegative}$$

$$\text{FN} = \text{FalseNegative}$$

Table 4 Mean Average Precision values of different models when compared with the proposed model on 17 classes of the MS-COCO dataset

Models	Classes																
	Bike	Bird	Car	Cow	Bottle	Bus	Cat	Dog	Person	Chair	Table	Horse	Motorbike	Plant	Sofa	Bed	Cellphone
Faster R-CNN	27.75	26.47	25.77	23.46	27.16	22.31	24.53	27.19	28.44	30.27	31.28	22.28	27.54	27.88	21.93	27.30	31.31
SSD	27.30	28.44	27.99	26.53	23.77	25.77	20.93	23.46	24.49	27.30	28.304	30.31	32.69	31.24	22.73	22.42	27.19
Mask R-CNN	33.01	27.30	22.69	30	30	29.23	21.17	24.53	28.92	32.69	20.83	24.35	28.54	31.41	22.28	25.01	29.16
YOLOv4	33.46	23.49	30.34	29.23	33.77	30.31	22.31	25.81	29.23	33.73	22.35	25.53	29.16	32.76	21.55	25.39	27.71
Baseline YOLOv6	33.49	27.19	34.15	31.17	32.31	33.77	22.28	24.29	23.32	26.08	30.31	23.07	33.46	30.31	26.81	22.62	31.86
Finetuned Framework	34.15	30.83	24.49	33.07	34.49	26.47	23.42	29.93	28.47	26.47	32.69	34.53	29.61	22.52	27.16	33.73	27.30

Precision helps in measuring the accuracy of the predictions made by a model. It helps in measuring to what extent the predictions of a model are correct. Object detection models make predictions with the help of bounding boxes and class labels. For each bounding box, an overlap is measured between the predicted bounding box and the ground truth bounding box. This is known as Intersection over Union (IoU). Recall measures how well a model finds all the positives.

4.2 Finetuned framework evaluation

The proposed model is evaluated and analyzed to check whether it has utilized the transfer learning algorithm. It has been observed that the precision and recall rates of the proposed model are higher than the baseline Yolov6 model and are comparable to the Yolov4 model. Baseline Yolov6 and the proposed model are superior to other object detection models such as Faster RCNN, SSD, Mask RCNN, and YOLOv4 in the visualization process of the detection algorithms. The main focus was on the speed performance of all the models with the help of FPS at a batch size of 32 on 17 classes of the MS-COCO dataset. The overall recall results of different models compared in this study are shown in Table 2 and precision values are shown in Table 3. Precision and recall values are calculated based on IoU threshold values. For example, if the IoU threshold value is 0.5, and IoU for a prediction is 0.8, it is considered True Positive (TP) and if the prediction is 0.3, then it is considered False Positive (FP). In this study, IoU threshold values were considered within a range of 0.5–0.9. The proposed model when compared with other object detection models at different IoU threshold values, provided the best results of precision and recall at the threshold value of 0.7 as shown in Fig. 8. F1-score was also calculated but is not considered in this study as the results do not vary much.

Figure 9 and Fig. 10 show the recall and precision curves for all the object detection models used in the experiment on 17 classes from the MS-COCO dataset. It can be observed from the figure that the recall rate of the finetuned framework is much better than all the other object detection models used. The average precision values of all 17 classes are detected by different object detection models in this study. Average precision is finding the area under the precision-recall curve and the formula is shown in Eq. 9. This is a multi-class classification; therefore for each class, the precision-recall curve is calculated and then the average precision value is determined with n number of thresholds.

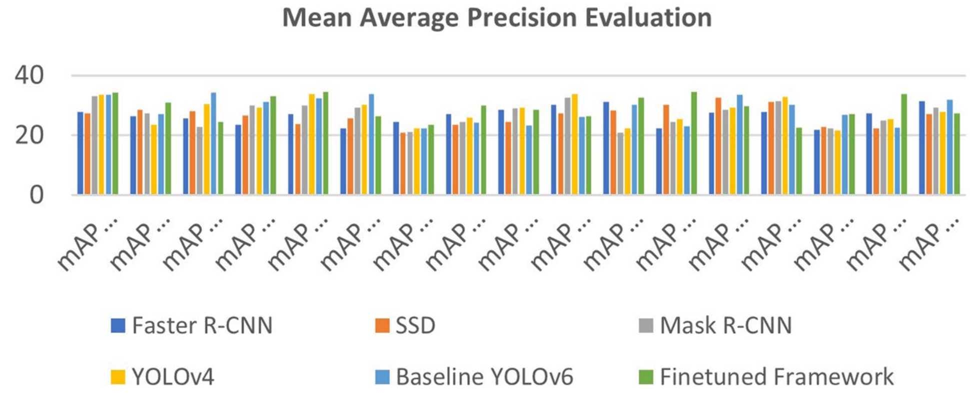


Fig. 11 Mean average precision for various object detection models when compared with the proposed framework

$$AP = \sum_{i=0}^{n-1} [\text{Recall}(i) - \text{Recall}(i+1)] * \text{Precision}(i) \tag{9}$$

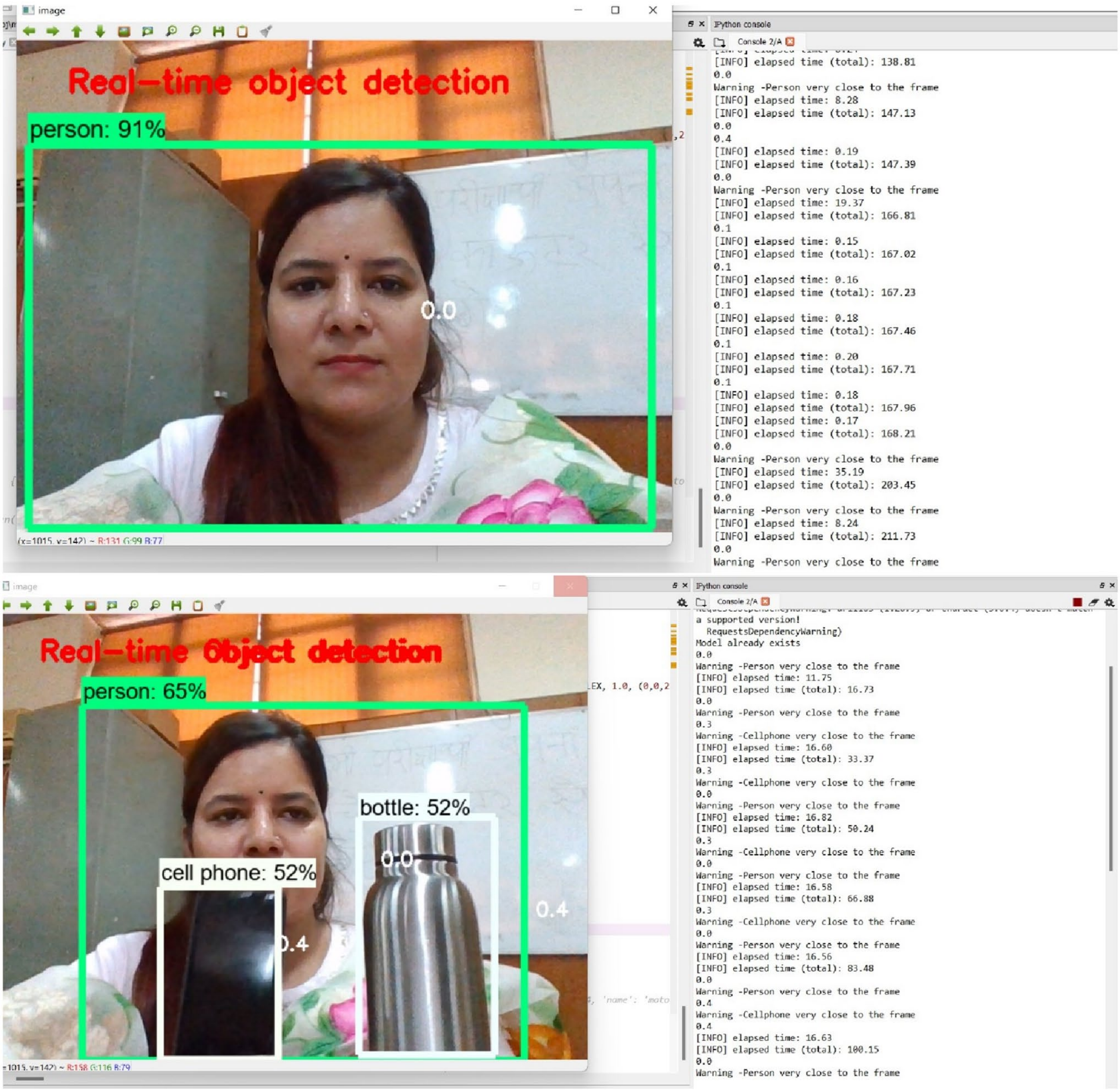
Once the average precision is calculated, the mean average precision is evaluated. Mean average precision (mAP) is the average of average precision as shown in Eq. 10 with n number of classes. The mAP values of all the 17 classes selected for the experiment are shown in Table 4 and Fig. 11 provides the graphical representation for mAP of different object detection models on different classes.

$$\text{mAP} = \frac{1}{n} \sum_{i=1}^{n} \text{AP}_i \tag{10}$$

In all the results, it can be viewed that some of the objects are not detected well by the proposed model. This is due to the lack of visibility of the object in different videos. The proposed model is not able to cope with the challenge of textured background where the object blends into the background and it becomes hard to identify those objects. This is the challenge that will be overcome in the future.

4.3 Results prediction in real time

The proposed model has achieved remarkable results in the real-time environment when implemented with a webcam on Intel(R) Core (TM) i5-1135G7 @ 2.40 GHz 2.42 GHz processor with NVIDIA GeForce × 360 graphic card. The model is combined with the gTTs (google Text-to-Speech) python library. The real-time video is converted to frames the and pytesseract python library is used to convert those images in frames to text. The gTTs library is then applied to this text and provides the automated voice output. Figure 12 provides some of the real-time object detection results with automated voice output. The images in Fig. 12 show confidence scores with each bounding box and it is observed that the model achieved good results in terms of confidence scores while detecting different objects. The inference speed of all the object detection models discussed in this paper is also computed in a real-time environment and results clearly show that the proposed finetuned YOLOv6 is much faster in identifying objects in the real world. The voice output identifies each detected object in the real world and also talks about the distance of each detected object from the webcam and instructs the user whether the object is too near or too far from them which will be helpful for the visually impaired person. The measured distance of each detected object is also visible in the images in Fig. 12. Frames per second (FPS) is also calculated for each object detection model. FPS tells how fast a model is processing any video and generating the desired results [30]. The results show that the proposed model achieved a higher inference speed at a high FPS when compared to other object detection models. The proposed model when implemented on YouTube videos provides outstanding results. Figure 13 shows results on a YouTube traffic video in which it is detecting cars and trucks with much better confidence scores and provides automated voice output for each object in the video. The proposed model is detecting different classes like the person, bottle, cell phone, car, bus, truck, chair, bed, and dog. The model achieved remarkable results when compared with other models in terms of inference speed on different real-time classes and Frames per Second (FPS). The results are shown in Table 5. The real-time images in Fig. 12 present the challenge of the proposed model. The proposed model is not able to overcome the challenge of a 'Textured Background'. The textured background is an open challenge in object detection where the object presents in a scene that blends into the background and is difficult to identify. The

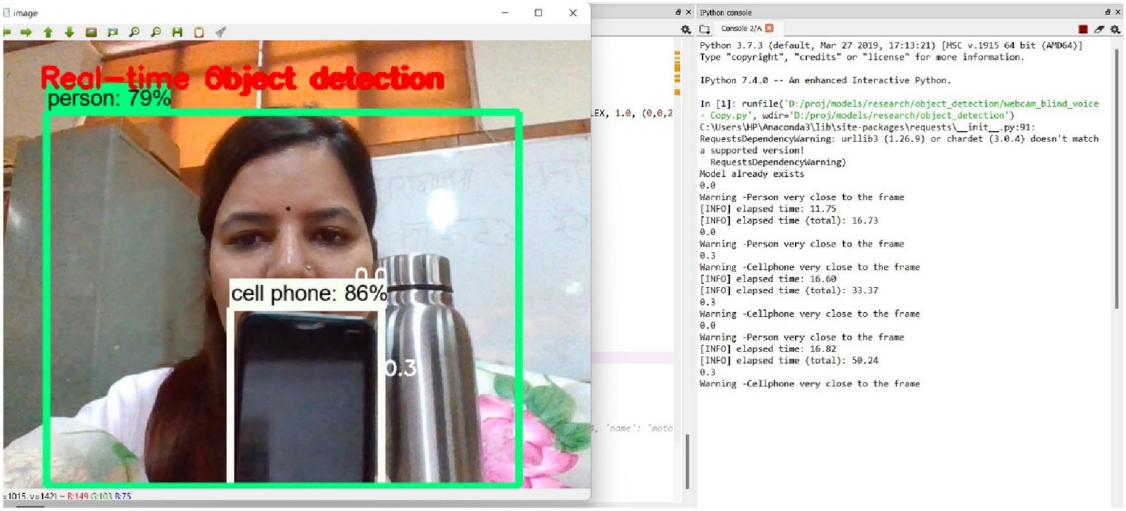


```
[INFO] elapsed time (total): 138.81
0.0
Warning -Person very close to the frame
[INFO] elapsed time: 8.28
[INFO] elapsed time (total): 147.13
0.0
0.4
[INFO] elapsed time: 0.19
[INFO] elapsed time (total): 147.39
0.0
Warning -Person very close to the frame
[INFO] elapsed time: 19.37
[INFO] elapsed time (total): 166.81
0.1
[INFO] elapsed time: 0.15
[INFO] elapsed time (total): 167.02
0.1
[INFO] elapsed time: 0.16
[INFO] elapsed time (total): 167.23
0.1
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 167.46
0.1
[INFO] elapsed time: 0.20
[INFO] elapsed time (total): 167.71
0.1
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 167.96
[INFO] elapsed time: 0.17
[INFO] elapsed time (total): 168.21
0.0
Warning -Person very close to the frame
[INFO] elapsed time: 35.19
[INFO] elapsed time (total): 203.45
0.0
Warning -Person very close to the frame
[INFO] elapsed time: 8.24
[INFO] elapsed time (total): 211.73
0.0
Warning -Person very close to the frame
```

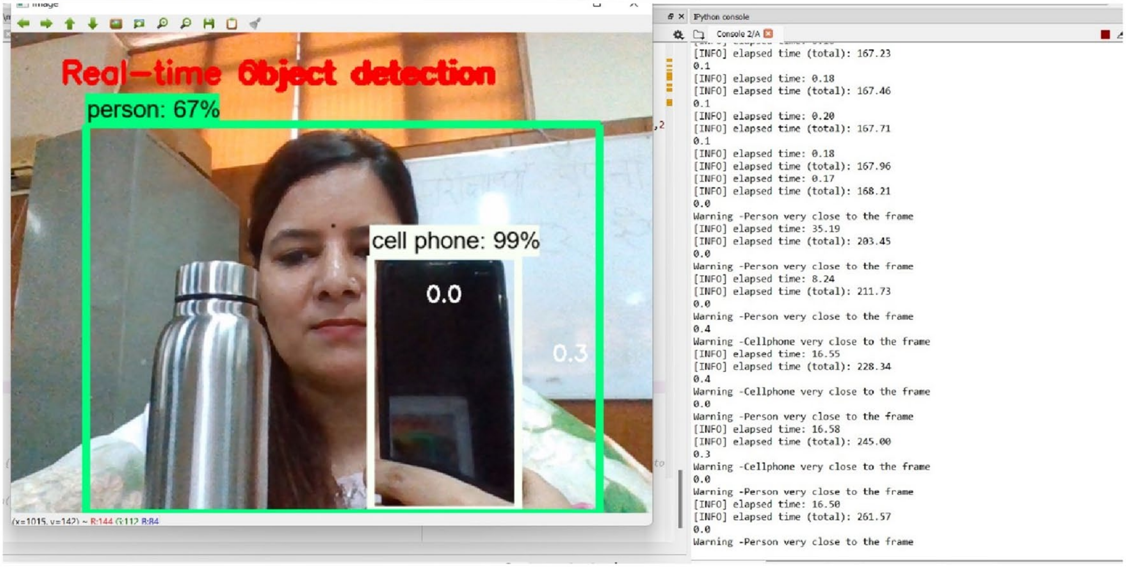
```
a supported version!
  RequestsDependencyWarning)
Model already exists
0.0
Warning -Person very close to the frame
[INFO] elapsed time: 11.75
[INFO] elapsed time (total): 16.73
0.0
Warning -Person very close to the frame
0.3
Warning -Cellphone very close to the frame
[INFO] elapsed time: 16.60
[INFO] elapsed time (total): 33.37
0.3
Warning -Cellphone very close to the frame
0.0
Warning -Person very close to the frame
[INFO] elapsed time: 16.82
[INFO] elapsed time (total): 50.24
0.3
Warning -Cellphone very close to the frame
0.0
Warning -Person very close to the frame
[INFO] elapsed time: 16.58
[INFO] elapsed time (total): 66.88
0.3
Warning -Cellphone very close to the frame
0.0
Warning -Person very close to the frame
[INFO] elapsed time: 16.56
[INFO] elapsed time (total): 83.48
0.0
Warning -Person very close to the frame
0.4
Warning -Cellphone very close to the frame
0.4
[INFO] elapsed time: 16.63
[INFO] elapsed time (total): 100.15
0.0
Warning -Person very close to the frame
```

Fig. 12 Real-time object detection results with the proposed finetuned framework with audio output

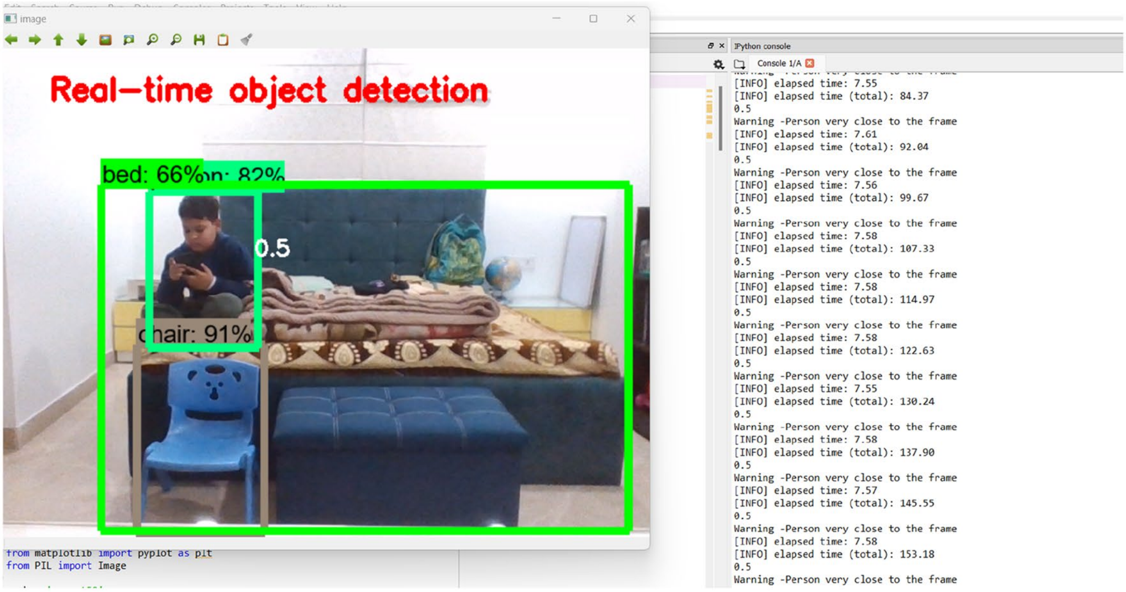
objects in the images like the 'bag', and 'couch' in front of the bed are of the same color as that of the bed, hence it becomes a challenge for the proposed model to identify these objects. The couch is identified as a chair in one of the images but it is not identified properly in any other image. This is an open challenge in the proposed model which will be worked upon in the future. It is visible from the real-time images that the proposed framework can identify occluded objects and shadows of objects in the scene like 'bed', 'person sitting on the bed', etc.



```
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.4.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/proj/models/research/object_detection/webcam_blind_voice
- Copy.py', wdir='D:/proj/models/research/object_detection')
C:\Users\HP\Anaconda3\lib\site-packages\requests\__init__.py:91:
RequestsDependencyWarning: urllib3 (1.26.9) or chardet (3.0.4) doesn't match
a supported version!
  RequestsDependencyWarning)
Model already exists
0.0
Warning -Person very close to the frame
[INFO] elapsed time: 11.75
[INFO] elapsed time (total): 16.73
0.0
Warning -Person very close to the frame
0.3
Warning -Cellphone very close to the frame
[INFO] elapsed time: 16.60
[INFO] elapsed time (total): 33.37
0.3
Warning -Cellphone very close to the frame
0.0
Warning -Person very close to the frame
[INFO] elapsed time: 16.82
[INFO] elapsed time (total): 50.24
0.3
Warning -Cellphone very close to the frame
```



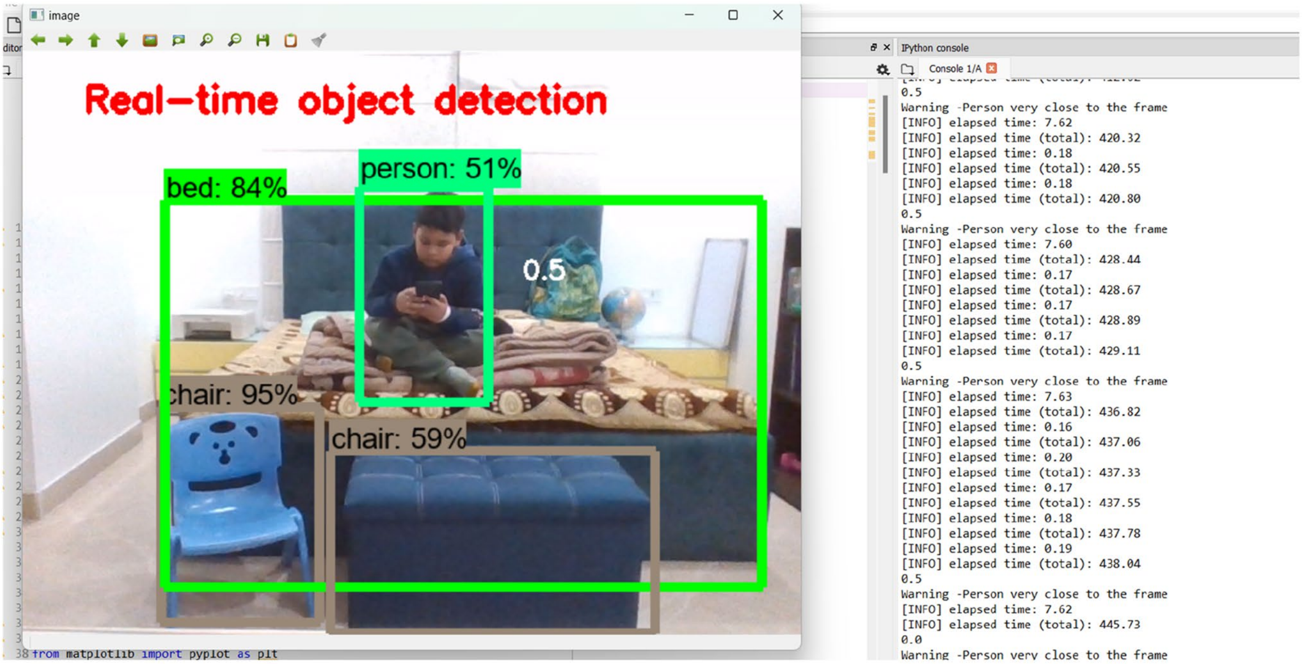
```
[INFO] elapsed time (total): 167.23
0.1
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 167.46
0.1
[INFO] elapsed time: 0.20
[INFO] elapsed time (total): 167.71
0.1
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 167.96
[INFO] elapsed time: 0.17
[INFO] elapsed time (total): 168.21
0.0
Warning -Person very close to the frame
[INFO] elapsed time: 35.19
[INFO] elapsed time (total): 203.45
0.0
Warning -Person very close to the frame
[INFO] elapsed time: 8.24
[INFO] elapsed time (total): 211.73
0.0
Warning -Person very close to the frame
0.4
Warning -Cellphone very close to the frame
[INFO] elapsed time: 16.55
[INFO] elapsed time (total): 228.34
0.4
Warning -Cellphone very close to the frame
0.0
Warning -Person very close to the frame
[INFO] elapsed time: 16.58
[INFO] elapsed time (total): 245.00
0.3
Warning -Cellphone very close to the frame
0.0
Warning -Person very close to the frame
[INFO] elapsed time: 16.50
[INFO] elapsed time (total): 261.57
0.0
Warning -Person very close to the frame
```



```
[INFO] elapsed time: 7.55
[INFO] elapsed time (total): 84.37
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.61
[INFO] elapsed time (total): 92.04
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.56
[INFO] elapsed time (total): 99.67
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.58
[INFO] elapsed time (total): 107.33
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.58
[INFO] elapsed time (total): 114.97
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.58
[INFO] elapsed time (total): 122.63
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.55
[INFO] elapsed time (total): 130.24
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.58
[INFO] elapsed time (total): 137.90
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.57
[INFO] elapsed time (total): 145.55
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.58
[INFO] elapsed time (total): 153.18
0.5
Warning -Person very close to the frame
```

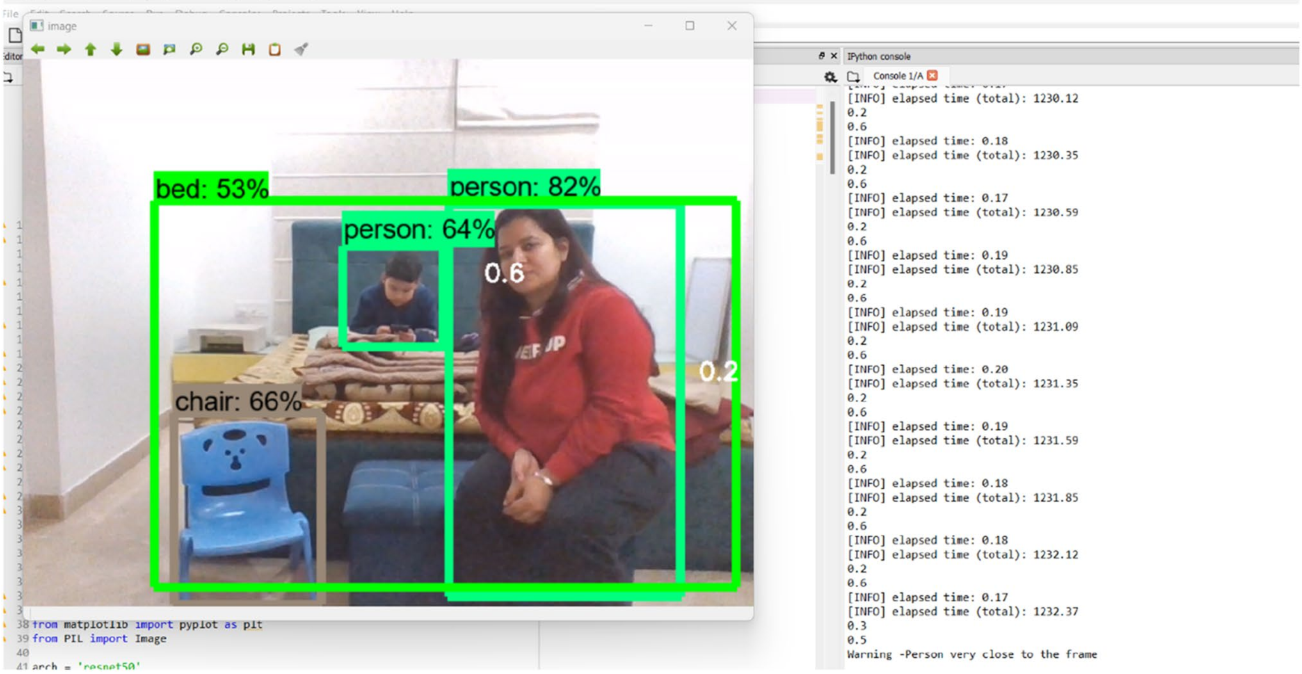
```
from matplotlib import pyplot as plt
from PIL import Image
```

Fig. 12 (continued)



```
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.62
[INFO] elapsed time (total): 420.32
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 420.55
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 420.80
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.60
[INFO] elapsed time (total): 428.44
[INFO] elapsed time: 0.17
[INFO] elapsed time (total): 428.67
[INFO] elapsed time: 0.17
[INFO] elapsed time (total): 428.89
[INFO] elapsed time: 0.17
[INFO] elapsed time (total): 429.11
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.63
[INFO] elapsed time (total): 436.82
[INFO] elapsed time: 0.16
[INFO] elapsed time (total): 437.06
[INFO] elapsed time: 0.20
[INFO] elapsed time (total): 437.33
[INFO] elapsed time: 0.17
[INFO] elapsed time (total): 437.55
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 437.78
[INFO] elapsed time: 0.19
[INFO] elapsed time (total): 438.04
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.62
[INFO] elapsed time (total): 445.73
0.0
Warning -Person very close to the frame
```

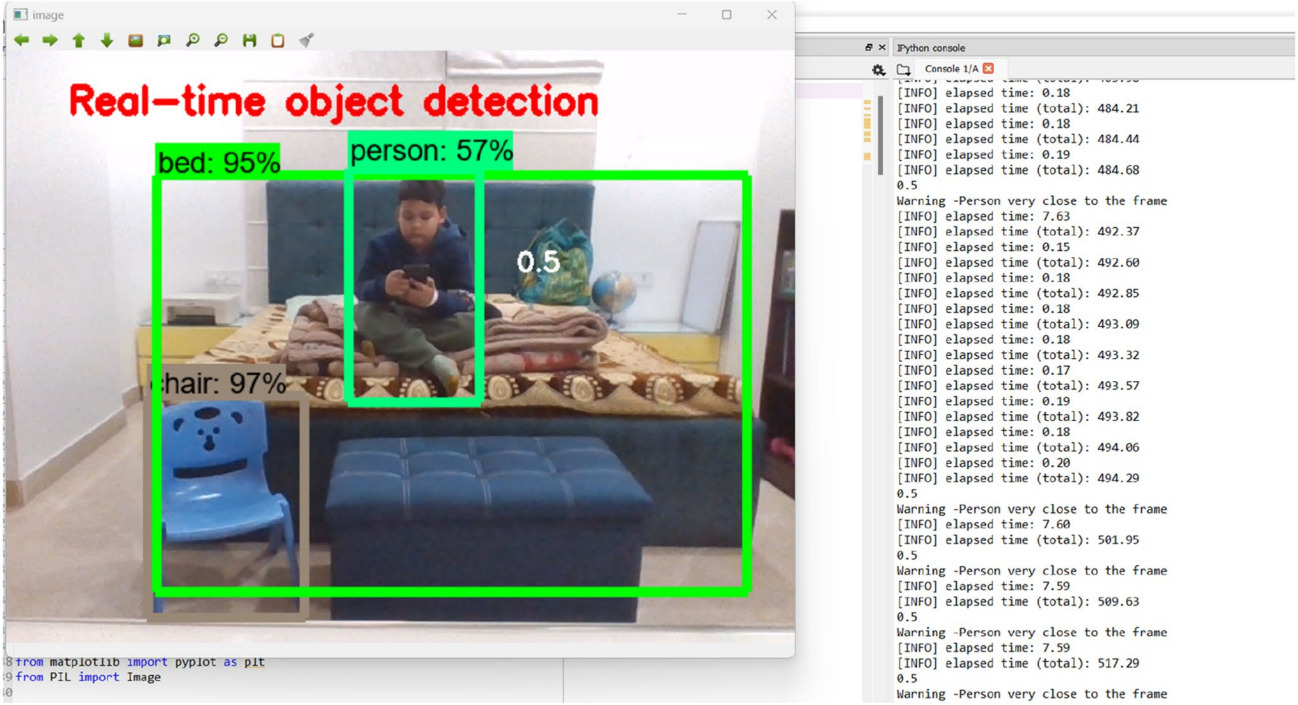
```
from matplotlib import pyplot as plt
```



```
[INFO] elapsed time (total): 1230.12
0.2
0.6
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 1230.35
0.2
0.6
[INFO] elapsed time: 0.17
[INFO] elapsed time (total): 1230.59
0.2
0.6
[INFO] elapsed time: 0.19
[INFO] elapsed time (total): 1230.85
0.2
0.6
[INFO] elapsed time: 0.19
[INFO] elapsed time (total): 1231.09
0.2
0.6
[INFO] elapsed time: 0.20
[INFO] elapsed time (total): 1231.35
0.2
0.6
[INFO] elapsed time: 0.19
[INFO] elapsed time (total): 1231.59
0.2
0.6
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 1231.85
0.2
0.6
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 1232.12
0.2
0.6
[INFO] elapsed time: 0.17
[INFO] elapsed time (total): 1232.37
0.3
0.5
Warning -Person very close to the frame
```

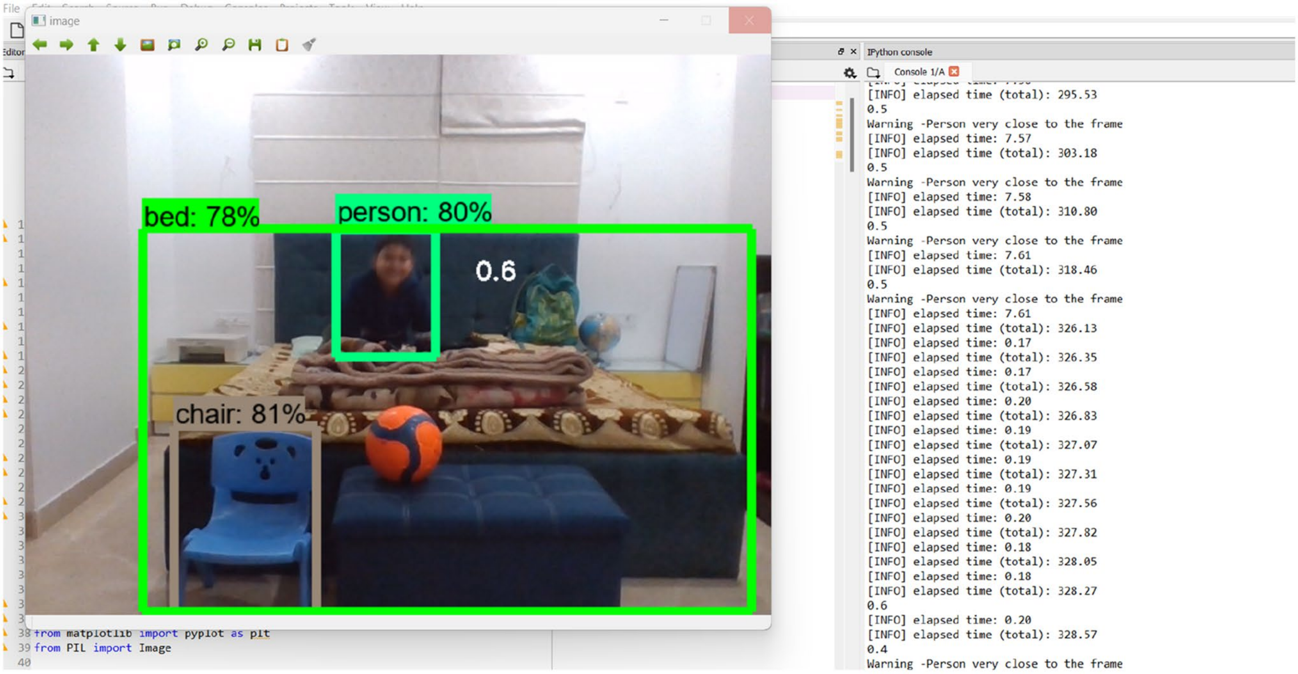
```
from matplotlib import pyplot as plt
from PIL import Image

arch = 'resnet50'
```

Fig. 12 (continued)



```
from matplotlib import pyplot as plt
from PIL import Image
```

```
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 484.21
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 484.44
[INFO] elapsed time: 0.19
[INFO] elapsed time (total): 484.68
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.63
[INFO] elapsed time (total): 492.37
[INFO] elapsed time: 0.15
[INFO] elapsed time (total): 492.60
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 492.85
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 493.09
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 493.32
[INFO] elapsed time: 0.17
[INFO] elapsed time (total): 493.57
[INFO] elapsed time: 0.19
[INFO] elapsed time (total): 493.82
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 494.06
[INFO] elapsed time: 0.20
[INFO] elapsed time (total): 494.29
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.60
[INFO] elapsed time (total): 501.95
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.59
[INFO] elapsed time (total): 509.63
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.59
[INFO] elapsed time (total): 517.29
0.5
Warning -Person very close to the frame
```



```
from matplotlib import pyplot as plt
from PIL import Image
```

```
[INFO] elapsed time (total): 295.53
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.57
[INFO] elapsed time (total): 303.18
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.58
[INFO] elapsed time (total): 310.80
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.61
[INFO] elapsed time (total): 318.46
0.5
Warning -Person very close to the frame
[INFO] elapsed time: 7.61
[INFO] elapsed time (total): 326.13
[INFO] elapsed time: 0.17
[INFO] elapsed time (total): 326.35
[INFO] elapsed time: 0.17
[INFO] elapsed time (total): 326.58
[INFO] elapsed time: 0.20
[INFO] elapsed time (total): 326.83
[INFO] elapsed time: 0.19
[INFO] elapsed time (total): 327.07
[INFO] elapsed time: 0.19
[INFO] elapsed time (total): 327.31
[INFO] elapsed time: 0.19
[INFO] elapsed time (total): 327.56
[INFO] elapsed time: 0.20
[INFO] elapsed time (total): 327.82
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 328.05
[INFO] elapsed time: 0.18
[INFO] elapsed time (total): 328.27
0.6
[INFO] elapsed time: 0.20
[INFO] elapsed time (total): 328.57
0.4
Warning -Person very close to the frame
```

Fig. 12 (continued)

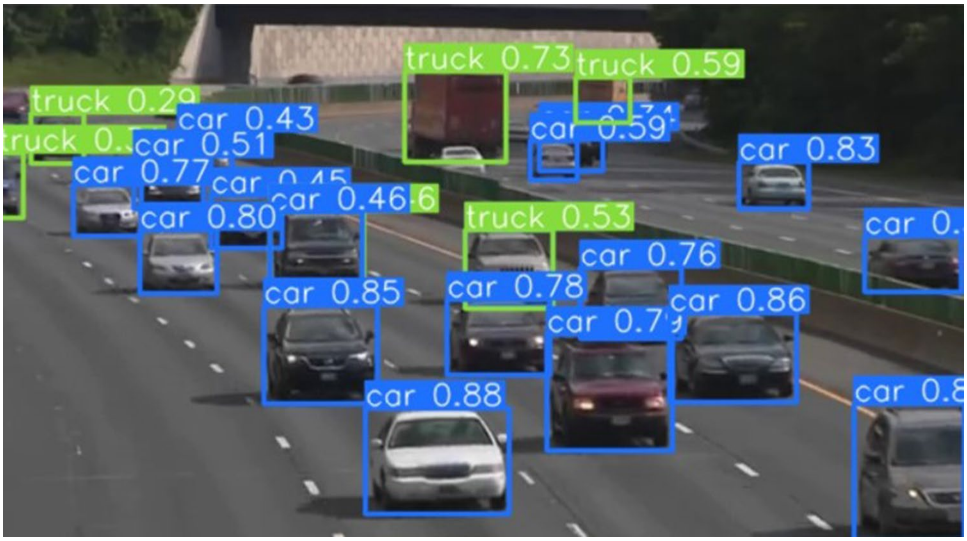


Fig. 13 Object detection results with the proposed framework with audio output on a YouTube video

Table 5 Real-time Implementation Results

Models	Inference Speed(ms) on different classes							FPS
	Person	Bottle	Cellphone	Car	Bus	Cat	Dog	
Faster RCNN	25.4	24.5	23.4	22.3	24.3	25.6	21.2	519
SSD	26.6	24.3	22.4	21.2	23.4	25.1	24.3	209
Mask RCNN	27.4	24.3	22.3	25.4	26.5	21.3	25.3	444
YOLOv4	26.5	22.3	23.5	25.6	26.7	27.8	21.7	602
Baseline YOLOv6	28.7	30.2	28.7	29.4	23.6	24.5	26.7	659
Finetuned Framework	31.2	32.3	33.5	34.3	32.1	30.7	31.1	1235

5 Conclusion and future work

The work implements transfer learning-based object detection uniquely. In this study, a finetuned Yolov6 model is presented as an improvement. To minimize the size of the model, speed up picture detection inference, and support real-time image processing, the baseline Yolov6 structure is tempered by pruning and finetuning it. For this purpose, a novel pruning and finetuning algorithm is proposed and implemented. The detection accuracy diminishes with decreasing model size. To solve this issue, the authors proposed a slightly modified transfer learning algorithm to improve the model's detection accuracy. The proposed model topology is simple, easy to set up, and has few parameters than baseline Yolov6, which enables the object detection algorithm to operate in the real world. The experiment is also carried out in a real-time environment, and it demonstrates that the proposed model, on the principle of transfer learning, shows very encouraging results. The proposed model has improved detection accuracy and inference speed and depicts a good balance between the two. In the future, the model will be investigated further and a convolutional kernel pruning algorithm will be implemented to compress the network width. The proposed model is not able to overcome the challenge of textured background (where the objects blend into the background making them hard to identify); hence in the future, the model will be enhanced to overcome this challenge. The framework can be combined with an IoT environment to create a physical device that may be helpful for visually impaired people.

Author contributions All authors have equal contribution in this manuscript.

Data availability Data will be made available on appropriate request.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Zhang, J., Wang, P., Zhao, Z., Su, F.: Pruned-YOLO: learning efficient object detector using model pruning. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect.

Notes Bioinformatics). 12894 LNCS, 34–45 (2021). https://doi.org/10.1007/978-3-030-86380-7_4/COVER/

2. Li, Y., Ge, Z., Yu, G., Yang, J., Wang, Z., Shi, Y., Sun, J., Li, Z.: BEVDepth: acquisition of reliable depth for multi-view 3D object detection. arXiv preprint. https://doi.org/10.48550/arXiv.2206.10092 (2022)
3. Xu, Q., Zhong, Y., Neumann, U.: Behind the curtain: learning occluded shapes for 3D object detection. Proc. AAAI Conf. Artif. Intell. **36**, 2893–2901 (2022). https://doi.org/10.1609/aaai.v36i3.20194
4. Sun, W., Dai, L., Zhang, X., Chang, P., He, X.: RSOD: real-time small object detection algorithm in UAV-based traffic monitoring. Appl. Intell. **52**, 8448–8463 (2022). https://doi.org/10.1007/s10489-021-02893-3
5. KhoshboreshMasouleh, M., Shah-Hosseini, R.: Development and evaluation of a deep learning model for real-time ground vehicle semantic segmentation from UAV-based thermal infrared imagery. ISPRS J. Photogramm. Remote Sens. **155**, 172–186 (2019). https://doi.org/10.1016/j.isprsjprs.2019.07.009
6. Hou, L., Chen, C., Wang, S., Wu, Y., Chen, X.: Multi-object detection method in construction machinery swarm operations based on the improved YOLOv4 model. Sensors. **22**, 1–14 (2022)
7. Mauri, A., Khemmar, R., Decoux, B., Haddad, M., Boutteau, R.: Lightweight convolutional neural network for real-time 3D object detection in road and railway environments. J. Real-Time Image Process. **19**, 499–516 (2022). https://doi.org/10.1007/s11554-022-01202-6
8. Martinez-Alpiste, I., Golcarenarenji, G., Wang, Q., Alcaraz-Calero, J.M.: Smartphone-based real-time object recognition architecture for portable and constrained systems. J. Real-Time Image Process. **19**, 103–115 (2022). https://doi.org/10.1007/s11554-021-01164-1
9. Hu, J., Wang, T., Zhu, S.: Multi-view aggregation for real-time accurate object detection of a moving camera. J. Real-Time Image Process. (2022). https://doi.org/10.1007/s11554-022-01253-9
10. Zhang, J., Ye, Z., Jin, X., Wang, J., Zhang, J.: Real-time traffic sign detection based on multiscale attention and spatial information aggregator. J. Real-Time Image Process. (2022). https://doi.org/10.1007/s11554-022-01252-w
11. Saponara, S., Elhanashi, A., Zheng, Q.: Developing a real-time social distancing detection system based on YOLOv4-tiny and bird-eye view for COVID-19. J. Real-Time Image Process. **19**, 551–563 (2022). https://doi.org/10.1007/s11554-022-01203-5
12. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 779–788 (2016)
13. Nikkath Bushra, S., Shobana, G., Uma Maheswari, K., Subramanian, N.: Smart video survillance based weapon identification using yolov5. 351–357 (2022). https://doi.org/10.1109/ICESIC53714.2022.9783499
14. Xia, R., Li, G., Huang, Z., Pang, Y., Qi, M.: Transformers only look once with nonlinear combination for real-time object detection. Neural Comput. Appl. (2022). https://doi.org/10.1007/s00521-022-07333-y
15. Junayed, M.S., Islam, M.B., Imani, H., Aydin, T.: PDS-Net: a novel point and depth-wise separable convolution for real-time object detection. Int. J. Multimed. Inf. Retr. **11**, 171–188 (2022). https://doi.org/10.1007/s13735-022-00229-6
16. Kadhim, M., Oleiwi, B.: Blind assistive system based on real time object recognition using machine learning. Eng. Technol. J. **40**, 159–165 (2022). https://doi.org/10.30684/etj.v40i1.1933
17. Ashiq, F., Asif, M., Ahmad, M.B., Zafar, S., Masood, K., Mahmood, T., Mahmood, M.T., Lee, I.H.: CNN-based object recognition and tracking system to assist visually impaired people. IEEE Access. **10**, 14819–14834 (2022). https://doi.org/10.1109/ACCESS.2022.3148036
18. Gupta, C., Gill, N.S., Gulia, P.: SSDT : distance tracking model based on deep learning. Int. J. Electr. Comput. Eng. Syst. **13**, 339–348 (2022). https://doi.org/10.32985/ijeces.13.5.2
19. Gupta, C., Gill, N.S.: Coronamask: a face mask detector for real-time data. Int. J. Adv. Trends Comput. Sci. Eng. **9**, 5624–5630 (2020). https://doi.org/10.30534/ijatcse/2020/212942020
20. Cai, Y., Yuan, G., Li, H., Niu, W., Li, Y., Tang, X., Ren, B., Wang, Y.: A compression-compilation co-design framework towards real-time object detection on mobile devices. 35th AAAI Conf. Artif. Intell. AAAI 2021. 18: 1597–1600 (2021)
21. Chen, C., Wang, G., Peng, C., Fang, Y., Zhang, D., Qin, H.: Exploring rich and efficient spatial temporal interactions for real-time video salient object detection. IEEE Trans. Image Process. **30**, 3995–4007 (2021). https://doi.org/10.1109/TIP.2021.3068644
22. What's New in YOLOv6?, https://blog.roboflow.com/yolov6/
23. Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B, 30m., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, X., Wei, X.: YOLOv6: A single-stage object detection framework for industrial applications. (2022)
24. Zhang, H., Wang, Y., Dayoub, F., Sünderhauf, N.: VarifocalNet: An IoU-aware dense object detector. Proc. IEEE Comput. Soc. Conf Comput. Vis. Pattern Recognit. (2021). https://doi.org/10.1109/CVPR46437.2021.00841
25. Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., Tang, J., Yang, J.: Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. Adv. Neural Inf. Process. Syst. **2020**, 1–11 (2020)
26. Bonnaerens, M., Freiberger, M., Dambre, J.: Anchor pruning for object detection. Comput. Vis. Image Underst. **221**, 1035 (2022). https://doi.org/10.1016/j.cviu.2022.103445
27. Zhong, Y., Wang, J., Peng, J., Zhang, L.: Anchor box optimization for object detection. Proc. - 2020 IEEE Winter Conf. Appl. Comput. Vision, WACV 2020. 1275–1283 (2020). https://doi.org/10.1109/WACV45572.2020.9093498
28. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 8693 LNCS, 740–755 (2014). https://doi.org/10.1007/978-3-319-10602-1_48/COVER/
29. COCO - Common objects in context, https://cocodataset.org/#download
30. Mehta, R., Ozturk, C.: Object detection at 200 frames per second. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 11133 LNCS, 659–675 (2019). https://doi.org/10.1007/978-3-030-11021-5_41

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.