Multimedia Tools and Applications (2024) 83:1143–1164
https://doi.org/10.1007/s11042-023-15930-9

Optimized deep learning vision system for human action recognition from drone images

Hussein Samma[1] · Ali Salem Bin Sama[2]

Received: 13 September 2021 / Revised: 3 April 2023 / Accepted: 29 May 2023 /
Published online: 2 June 2023

Abstract
There are several benefits to constructing a lightweight vision system that is implemented directly on limited hardware devices. Most deep learning-based computer vision systems, such as YOLO (You Only Look Once), use computationally expensive backbone feature extractor networks, such as ResNet and Inception network. To address the issue of network complexity, researchers created SqueezeNet, an alternative compressed and diminutive network. However, SqueezeNet was trained to recognize 1000 unique objects as a broad classification system. This work integrates a two-layer particle swarm optimizer (TLPSO) into YOLO to reduce the contribution of SqueezeNet convolutional filters that have contributed less to human action recognition. In short, this work introduces a lightweight vision system with an optimized SqueezeNet backbone feature extraction network. Secondly, it does so without sacrificing accuracy. This is because that the high-dimensional SqueezeNet convolutional filter selection is supported by the efficient TLPSO algorithm. The proposed vision system has been used to the recognition of human behaviors from drone-mounted camera images. This study focused on two separate motions, namely walking and running. As a consequence, a total of 300 pictures were taken at various places, angles, and weather conditions, with 100 shots capturing running and 200 images capturing walking. The TLPSO technique lowered SqueezeNet's convolutional filters by 52%, resulting in a sevenfold boost in detection speed. With an F1 score of 94.65% and an inference time of 0.061 milliseconds, the suggested system beat earlier vision systems in terms of human recognition from drone-based photographs. In addition, the performance assessment of TLPSO in comparison to other related optimizers found that TLPSO had a better convergence curve and achieved a higher fitness value. In statistical comparisons, TLPSO surpassed PSO and RLMPSO by a wide margin.

Keywords Deep learning · Human action recognition · YOLO · SqueezeNet · Optimization algorithms

✉ Hussein Samma
hussein.samma@yahoo.com

[1] SDAIA-KFUPM Joint Research Center for Artificial Intelligence (JRC-AI), King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia

[2] Department of Management Information Systems (MIS), College of Business Administration, King Faisal University, Al-Ahsa, Eastern Province, Kingdom of Saudi Arabia

1 Introduction

Vision systems based on deep learning demonstrated astounding performance in human activity recognition. YOLO was deemed the most popular of these systems because to its simplicity, detection speed, and precision [7, 33]. Recognition of human activity was suggested by Shinde et al. using the YOLO framework [40]. In [40], they used YOLO for activity identification from a security camera, and the findings shown that YOLO outperformed other models in terms of F-score accuracy with 88.4%. Liu et al. investigated the issue of recognition of intention and interaction [25]. In their work, they focused on the hybrid of YOLO and LSTM for identifying human intent and interaction. Initially, YOLO was utilized to identify hand-held items, and then LSTM was used to encode the temporal dependencies of the series. The problem of action localization was discussed by Hammam et al. [7]. They exploited YOLO in recent work for person localisation in a succession of video frames. Mutis et al. [25] described the use of YOLO for human occupancy forecasts and activity analysis.

Recently, YOLOv7 has been effectively implemented in a range of real-world applications, including the recognition of objects in marine cruise using photos acquired by a drone camera [19]. Their tests compare the performance of YOLOv7 to that of YOLOv3, YOLOv4, and YOLOv5. The findings indicated that YOLOv7 had the greatest accuracy and recall rate for detecting airplanes and oil tanks. This is due to the changed design of the network's backbone. Li, Yongshuai, et al. [41] studied an additional use of YOLOv7 for fault identification in transmission line insulators using UAV photos. They developed an image-based inspection system employing several YOLO models, including YOLOv5, YOLOv6, and YOLOv7-X, in [41]. Using 1593 photos of the electrical grid, the investigation determined that YOLOv5 and YOLO7-X delivered the greatest results. In [55], another application of UAV-based object detection employing the YOLOv7-sea system was examined. The suggested YOLOv7-sea was utilized to identify swimmers, boats, jet skis, buoys, and lifesaving equipment from the SeaDronesSee dataset [46]. The usage of YOLOv7 for counting ducks is shown in [12]. They recommended adding an attention mechanism to YOLOv7 in order to capture key information. 200 photos were used for testing the trained YOLOv7 model, whereas 1300 images were utilized for training. According to the performed study, the proposed attention techniques slightly improved mAP accuracy by 1.15%. Yang et al. [26] presented the YOLOv7 ship identification model using satellite images.

Despite YOLO's success, it employs computationally costly pre-trained backbone feature extraction networks, like AlexNet [17], ResNet [8], and Inception [42]. For example, the ResNet network described in [8] has 152 levels, and each layer comprises hundreds of convolutional filters. Researchers designed SqueezeNet, a compressed network, to alleviate network complexity concerns [11]. Actually, SqueezeNet is far smaller than AlexNet, with 5X less parameters, yet it was able to attain the same level of accuracy in recognition as AlexNet. Inspired by its compact size, several researchers have deployed SqueezeNet to solve a variety of computer vision issues in the real world, including vehicle identification in thermal infrared pictures [15], COVID-19 detection from CT scans [29], and industrial defect localization [51]. All of these investigations demonstrated SqueezeNet's outstanding accuracy at a reasonable computing cost. Ren et al. [34] showed the integration of SqueezeNet with YOLO for real-time people counting. Using the SqueezeNet network resulted in further gains in speed, as determined by their study.

Nevertheless, the SqueezeNet used in [15, 29, 34, 51] consists of hundreds of convolutional filters that absorb the majority of computing effort during the detection period.

Using optimization techniques like as PSO [16], TLPSO [37], and RLMPSO [36] to conduct an iterative search and pruning of SqueezeNet filters is a viable method for bridging this gap and reducing the complexity of SqueezeNet. TLPSO was the most effective of these optimizers in tackling large-scale optimization issues. In order to prune and remove less contributing SqueezeNet neural filters in human action identification, this project will include TLPSO into YOLO. Consequently, the goals of this research are threefold.

1- To reduce the complexity of SqueezeNet by eliminating less contributed convolutional filters.
2- To incorporate an efficient optimizer (i.e., TLPSO) for performing SqueezeNet filters elimination guided by YOLO detection accuracy and system complexity.
3- To investigate the optimized YOLO vision system with respect to the issue of human activity recognition (walking vs. running) using drone-captured images, as shown in Fig. 1.

The originality of this work may be summed up in two parts namely (i) It introduced a compact vision system using an optimized SqueezeNet-based feature extraction network that has been fine-tuned, and (ii) It consists of an effective TLPSO method that can function with high-dimensional SqueezeNet convolutional filter selection. The remaining part of this work is organized as follows. The related work is given in Section 2. In Section 3, the proposed optimized YOLO vision system is explained. A series of conducted experiments used to evaluate the effectiveness of the proposed vision system are given in Section 4, followed by the conclusions in Section 5.

2 Related work

Human detection and action recognition have been extensively studied in the literature. Hung et al. [10] presented a Faster-RCNN-based technique for people identification in drone-based photos. The findings suggested that the F1 measurement was accurate to a

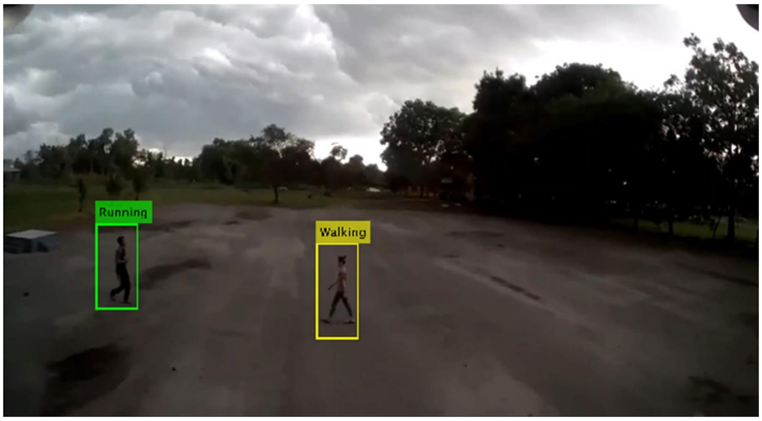


Fig. 1 Two different human actions are walking (yellow box) and running (green box)

degree of 98%. In [10], both Faster-RCNN and YOLO were built and exhibited comparable detection accuracy; however, YOLO requires shorter execution time because to the one-stage architecture's benefit [33]. CNN-based deep learning models have been described in [1, 21, 38, 54], and [14]. The merging of CNN and LSTM for activity identification was studied, for example, in [21, 54]. Maitre et al. [21] suggested a CNN-LSTM model for everyday human activity recognition. In their study, CNN was used to encode the characteristics of a 1-D radar signal, which was then given to the LSTM model for activity classification. The hybrid CNN-LSTM model described in [21] was verified using 10 subjects doing a variety of everyday activities such as drinking, sleeping, walking, etc. The results shown that CNN-LSTM classification accuracy approached 90%.

Amin Ullah et al. [45] did more research focusing on the topic of activity detection from security video streams. The system described in [21] comprises three phases: object localization using YOLO [33], feature extraction with CNN, and sequence recognition with Gated Recurrent Unit (GRU). Their technique was used to identify a variety of human activities, including sports participation, musical instrument playing, human-to-human contact, etc. According to reported statistics, their technology was able to attain an average accuracy rate of 80%. Other that human detection and action recognition, several studies addressed the issue of image classification. For instance, a comprehensive analysis that compares several features descriptors including SIFT,SURF, and ORB in image classification was conducted by [2]. In [3], the concept of transfer learning using a pre-trained VGG19 for image classification issues was presented. In [39], the approaches, problems, and opportunities in figner vien recognition are thoroughly reviewed.

Recently, YOLO has been adopted for various agriculture detection applications such as plant organs detection [44], Tea chrysanthemum detection [30], date palm tree detection from drone imagery [13], tea leaf detection [4], tomato detection [53], pinecone detection [6], fruit detection [43]. In [44], ResNet and DenseNet backbone feature extractor networks were utilised to train YOLO-v3 for plant orangutan identification. The final classification layer of YOLO-v3 generated four distinct classes, including leaves, flowers, fruits, and buds. In their trial, 4000 photos were used, and YOLO-v3 claimed a recall rate of 95.5% and a precision of 94.2%. The completed research revealed that the recall rate was boosted by 5.7% and the precision rate increased by 4.2% as a result of data augmentation. The application of YOLO for tea detection at flowering stage was investigated by Qi et al. [30]. Mainly, they have proposed modifications to the backbone component and neck of YOLO. Results showed that detection accuracy reached 92.49%. The YOLO-based face mask detection challenge was posed by Wu et al. [49]. Their system attained a 92.0% precision rate when tested on publicly available datasets at Kaggle, where they reviewed their methodology. Nevertheless, earlier reported approaches [30, 44, 49] utilised the ResNet backbone network, which has a sophisticated architecture and takes more processing effort than SqueezeNet. Intasuttisak et al. [13] recently adopted YOLO-v5 for the recognition of date palm trees in drone footage. They have accumulated 125 photos and put them into three categories: 60% for training, 20% for validation, and 20% for testing. The mean attained precision rate for test data was 92.34%. Some further applications of YOLO include road crack detection [32], semi-supervised YOLO for generic object detection [56], head detection [48], defect detection [28], YOLO-based a few-shot model [50], detecting small targets in infrared remote sensing [18], vehichle detection [5], traffic sign detection [52], colon cancer detection [24], and cattle body detection [31].

In spite of this, a small number of research, such as [22, 27] have been conducted to address the problem of human action recognition from UAVs. Mliki et al. [22] were able to concurrently find and identify individuals' behaviours with the assistance of CNN. By extending their method to UAV-based video sequences, they were able to demonstrate that CNN outperforms more traditional detection and identification techniques. Peng et al. [27] accelerated the process by employing a region-based technique [35] with 3D CNN. The literature is summarised in Table 1 for relevant investigations.

3 The proposed optimized DEEP learning vision system

Figure 2 depicts the primary architecture of the proposed optimised vision system. As noted, it employs TLPSO to identify and eleminate SqueezeNet filters with fewer contributions throughout the detection process. However, before embedding TLPSO, YOLO [33] will be trained with the SqueezeNet backbone network for human action recognition (i.e., walking vs. running); after that, it will undergo a pruning process (i.e., SqueezeNet filters selection/elimination). The main steps of the implemented process are depicted in Fig. 2, which include the SqueezeNet filters selection/elimination, fitness evaluation according to YOLO detection accuracy and percentage of filters reduction, and TLPSO particles updating phase.

Figure 3 depicts the TLPSO method, which consists of one layer for global search operations and a second layer for local search activities. Controlling the selection between these processes is a Q-learning method [47]. Our prior work [37] contains additional information regarding TLPSO. In summary, the TLPSO will begin with a random micro swarm (three particles), and each particle in the swarm is associated with two vectors, namely the velocity (V) and position (X) vectors, as shown below.

$$X_i = \left[d_i^1, d_i^2, d_i^3, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots., d_i^D\right] \tag{1}$$

$$V_i = \left[v_i^1, v_i^2, v_i^3, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots., v_i^D\right] \tag{2}$$

where D represents the dimension of the optimization problem (i.e., the total number of SqueezeNet filters), and *i* represents the particle number in the swarm. These particles are evolved, and their positions are updated according to the following equations:

$$V_i \leftarrow \omega * V_i + c_1 * \mathrm{rand}_{uniform}\left(pBest - X_i\right) + c_2 * \mathrm{rand}_{uniform}\left(gBest - X_i\right) \tag{3}$$

$$X_i \leftarrow X_i + V_i \tag{4}$$

where w is the inertia weight, c_1 is the cognitive acceleration coefficient, c_2 is the social acceleration coefficient, $\mathrm{rand}_{\mathrm{uniform}}$ is a uniformly distributed random number within [0, 1], pBest is the local best position achieved by a particular particle, and gBest is the global best position achieved by the whole population. As mentioned in [37], the local search operation is identical to the global search, but it modifies randomly chosen dimensions.

After that, the fitness of each particle will be determined according to Eq. (6). Figure 4 depicts the proposed encoding strategy used to convert particle vector to SqueezeNet filters selection/elimination format. As can be observed, each convolution filter in the convolutional layers of SqueezeNet will be associated with a bin variable F that can take the values 0 or 1. Thus, it is a binary optimisation issue where a value of one signals that the associated filter is selected and will be active throughout the YOLO features extraction procedure. A value of zero meant that the relevant SqueezeNet filter would be eliminated from SqueezeNet since it would be seen to have contributed less.

Table 1 Related work

Ref	Year	Author	Models	Benchmark	Accuracy
[21]	2021	J. Maitre et al.	CNN with LSTM	They examined fifteen distinct behaviors, including drinking, sleeping, etc.	90%
[54]	2021	Zhang et al.	CNN with LSTM	KTH, UCF50 and UCF101 dataset.	91%
[45]	2021	Amin Ullah et al.	YOLO, CNN, and GRU.	UCF-101 dataset, HMDB51 dataset, and UCF-50 dataset.	80%
[22]	2020	Mliki et al.	CNN	UCF-ARG	68%
[27]	2020	Peng et al.	Faster R-CNN with 3D CNN	UCF-ARG aerial imaging dataset	86%
[35]	2016	Saha et al.	Fast R-CNN	UCF101, J-HMDB-21, and LIRIS-HARL datasets	58%
[40]	2018	Shinde et al.	YOLO	LIRIS s dataset with ten different human activities.	88%
[20]	2020	Liu et al.	GNN, LSTM, and YOLO.	SBU dataset that contains 230 sequences of 8 classes and a BJK dataset for 15 hand-held objects.	94%
[7]	2020	Hammam et al.	YOLO	UCF-101-24 and J-HMDB-21 datasets.	72%
[25]	2020	Mutis et al.	YOLO	NADA action dataset contains six different actions.	83%
[34]	2020	Ren et al.	YOLO with SqueezeNet	PASCAL VOC dataset.	72%
[30]	2022	Qi et al.	YOLO	It was tested using 300 field images	92%
[38]	2022	Sha et al.	CNN-based approach	CityPersons and the ETH dataset	–
[13]	2022	Intasuttisak et al.	Yolo-V5	125 images captured by a drone-based camera	92%
[49]	2022	Wu et al.	YOLO with Im-Res2Net-101	Public dataset from Kaggle	92%
[1]	2022	Amudhan et al.	Hypermetropic CNN approach	VEDAI and Visdrone dataset	61%
[14]	2021	Junos et al.	Lightweight CNN model	VEDAI dataset	47%
[44]	2022	Triki et al.	Improved YOLO-V3	Specimen dataset with 4000 images	95%
[10]	2020	Hung et al.	Faster R-CNN	They used UAV123 public benchmark for pedestrian identification and drone-captured photos.	98%
[4]	2023	Bao et al.	YOLOv5	They employed 350 UAVs, each of which took a JPEG photograph 8000 by 6000 pixels in size.	71.5%
[53]	2023	Zeng et al.	Improved YOLOv5	They employed a homegrown tomato dataset. 932 pictures were captured in total.	95%
[6]	2023	Cui et al.	YOLOv4-Tiny	A total of 1200 photos of pinecones were obtained from a forest farm in the Chinese province of Heilongjiang.	95.3%
[43]	2023	Tang et al.	improved YOLOv4-tiny	1600 images for training and 400 for evaluation	92%
[32]	2023	Qiu et al.	YOLOv4-tiny	A dataset of 1000 photographs was randomly used as training, while the remaining 200 images were collected as test data.	94.54%

Table 1 (continued)

Ref	Year	Author	Models	Benchmark	Accuracy
[56]	2023	Zhou et al.	Semi-supervised YOLOv5	It was evaluated using PascalVOC, Clipart1k, Cityscapes, and Foggy Cityscapes as public benchmarks.	56%
[48]	2023	Wei et al.	HD-YOLO	Thier approach was evaluated utilising PHDF, SEU-fisheye, HABBOF, and CEPDOF datasets.	98%
[28]	2023	Vu et al.	YOLO	In total, 400 photographs were taken: 200 of broken boxes and 200 of intact ones.	78.6%
[50]	2023	Xia et al.	BC-YOLO	PASCAL VOC 2007 and MS COCO 2014 datasets were used	43.9%
[18]	2023	Li and Shen	YOLOSR-IST	Dataset from infrared image sequences (IRIS) and single-frame Infrared small target (SIRST)	99.2%
[5]	2023	Bie et al.	YOLOv5n-L	The BDD100K dataset was utilised.	67.8%
[52]	2023	Yao et al.	YOLOv4-Tiny	The CSUST Chinese Traffic Sign Detection Benchmark contains 15,734 pictures, including approximately 40,000 traffic signs of varying sizes.	86.8%
[24]	2023	Murugesan et al.	YOLOv3 Multi-Scale Framework (YOLOv3-MSF)	They employed CVC colonDB database	94.02%
[31]	2023	Qiau et al.	YOLOv5-ASF	A total of 300 photos of cows	94.1%

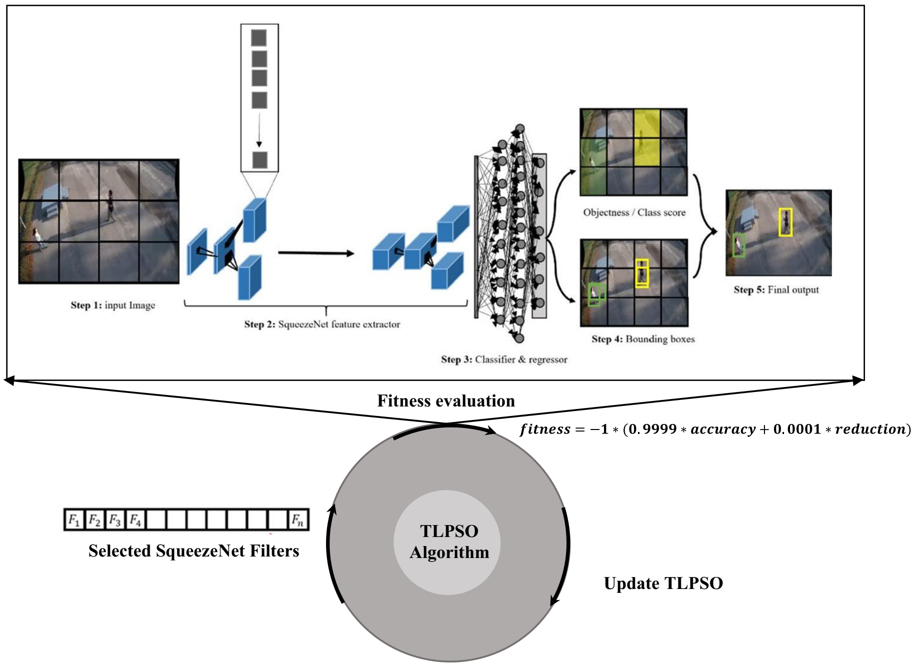

Step 1: input Image
Step 2: SqueezeNet feature extractor
Step 3: Classifier & regressor
Objectness / Class score
Step 4: Bounding boxes
Step 5: Final output
Fitness evaluation
fitness = −1 ∗ (0.9999 ∗ accuracy + 0.0001 ∗ reduction)
F1 F2 F3 F4 Fn
Selected SqueezeNet Filters
TLPSO Algorithm
Update TLPSO

Fig. 2 The proposed optimized vision system

Therefore, the optimizer iteratively performs a selection/elimination process of SqueezeNet filters guided by a fitness function defined as follows.

$$fitness = -1 * (0.9999 * F - score + 0.0001 * reduction) \tag{5}$$

where $F - score$ measure represents the accuracy rate of YOLO on training data. *reduction* variable is related to the percentage of eliminated filters from SqueezeNet (i.e. number of activated filters divided by the total number of SqueezeNet filters). It should be noted that higher weightage (i.e. 0.9999) is given to accuracy over reduction due to the importance

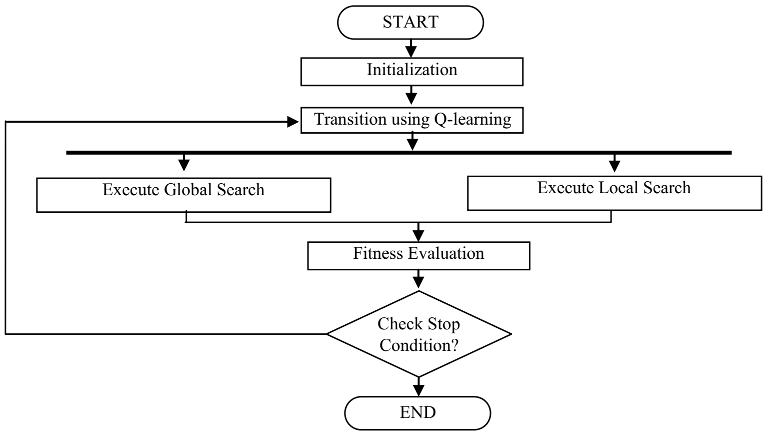


Fig. 3 TLPSO [37]

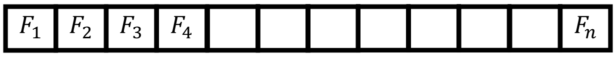


Fig. 4 Encoding scheme of SqueezeNet filters selection

of accuracy. The optimisation process will end once the maximum number of iterations has been reached, at which point the best possible solution will be returned. Once the final SqueezeNet has been pruned, it will be added back into YOLO and retrained so that the last layer of YOLO may be fine-tuned (regressor and classifier).

3.1 YOLO steps

As can be seen in Fig. 2 that the main steps of the YOLO system are (1) input image division, (2) feature extraction, (3) cell classification and regression, (4) bounding boxes generation, and (5) final output prediction. These steps are explained as follows:

Step 1: Divide the input image into cells.

At this step, the input image is divided into S x S cells, and each cell is responsible for predicting several anchor boxes, as given in Fig. 5. In this work, the input image will be divided into 7 × 7 cells with three anchor boxes as suggested in [40].

3.1.1 Feature extraction using SqueezeNet

At this stage, the entire image input is sent to SqueezeNet for feature extraction. The fundamental SqueezeNet architecture is depicted in Fig. 6 and Table 2 [11]. Evidently, SqueezeNet employs a number of filters to recognise visual characteristics such as edges, dots, etc. Even though, SqueezeNet convolutional filters consumed the most computational time, as each filter was convolved with the entire input image. Table 2 illustrates that, when SqueezeNet is explored further, the complexity of the network has increased and more convolutional filters will be required. Consequently, the objective of this research is to lower the number of convolutional filters in order to accelerate the feature extraction process of the suggested vision system.

3.1.2 Cell classification/regression

This stage is responsible for providing the output of YOLO classification and regressor layer. As stated previously, YOLO will be trained to recognize two human activities,

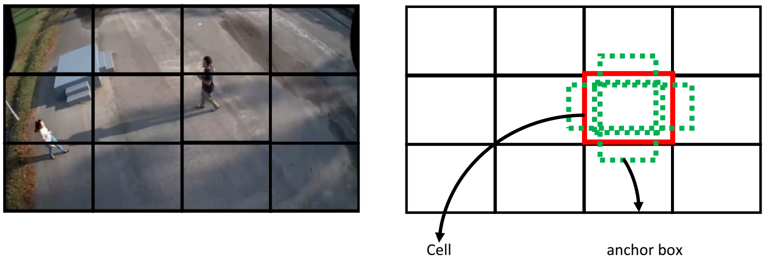


Fig. 5 Input image division

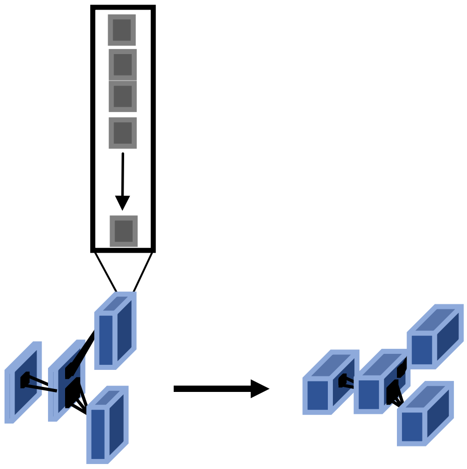


Fig. 6 Convolutional filters of SqueezeNet

namely walking and running. Thus, throughout the training process of YOLO, both the classification and regression layers will be tweaked and updated based on the loss function defined in the following equation: (1).

$$\begin{aligned} loss = {} & \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 \right] \\ & + \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{w_j}\right)^2 + \left(\sqrt{h_i} - \sqrt{h_j}\right)^2 \right] \\ & + \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(c_i - c_j\right) + \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \left(c_i - c_j\right) \\ & + \lambda_{coord} \sum_{i=0}^{s^2} 1_{i}^{noobj} \sum_{c\ \epsilon\ class} \left(p\left(c_j\right) - p\left(c_j\right)\right)^2 \end{aligned} \tag{6}$$

Table 2 SqueezeNet layers

No.	Layer name	Abbreviation	No. of filters	Filter size
1	Input	'data'		
2	Convolution	'conv1'	64	3×3
3	ReLU	'relu_conv1'		
4	Max Pooling	'pool1'		
5	Convolution	'fire2-squeeze1x1'	64	1×1
6	ReLU	'fire2-relu_squeeze1x1'		
7	Convolution	'fire2-expand1x1'	64	1×1
8	ReLU	'fire2-relu_expand1x1'		
9	Convolution	'fire2-expand3x3'		
10	ReLU	'fire2-relu_expand3×3'	64	3×3
11	Depth concatenation	'fire2-concat'		
64	Convolution	'conv10'	512	1×1
65	ReLU	'relu_conv10'		
66	Global Average Pooling	'pool10'		

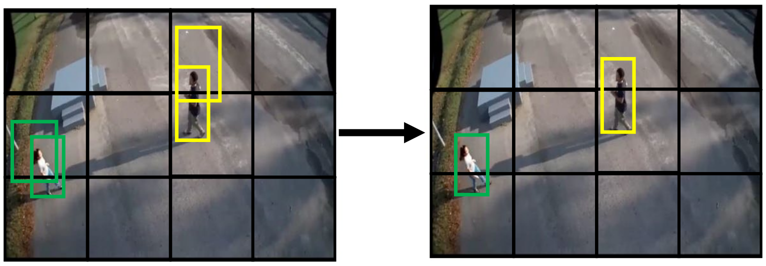


Fig. 7 YOLO final output

where x_i, y_i are the center of the bounding box. w_i, *and* h_i are the width and the height of the bounding box. C_i is the objectness measure that identifies the confidence level of whether the cell contains an object or not. $p_i(c)$ is related to class probability score. Variables S *and* B are related to the implemented number of cells and anchor boxes. It should be noted that the value 1_{ij}^{obj} will be 1 when a cell contains an object; otherwise, it will be set to zero. However, the value 1_{ij}^{noobj} will be 1 when there is no object in that cell and zero elsewhere.

3.1.3 Bounding boxes generation

This activity is responsible for generating the outcomes of both the classification and regeneration layers. According to the predicted location of the regressor, as shown in Fig. 7, many boxes were drawn. In addition, yellow boxes indicate running action, whereas green boxes indicate walking action.

3.1.4 YOLO final output prediction

This is the final phase of YOLO, which generates the final bounding boxes. First, only those bounding boxes with a confidence level more than 0.5 will be kept, while others will be discarded as false alarm boxes. Then, overlapping boxes will be combined, as seen in Fig. 7. The green box signifies a class of running motion, while the yellow box shows a class of walking movement.

Fig. 8 S-Series S30W drone

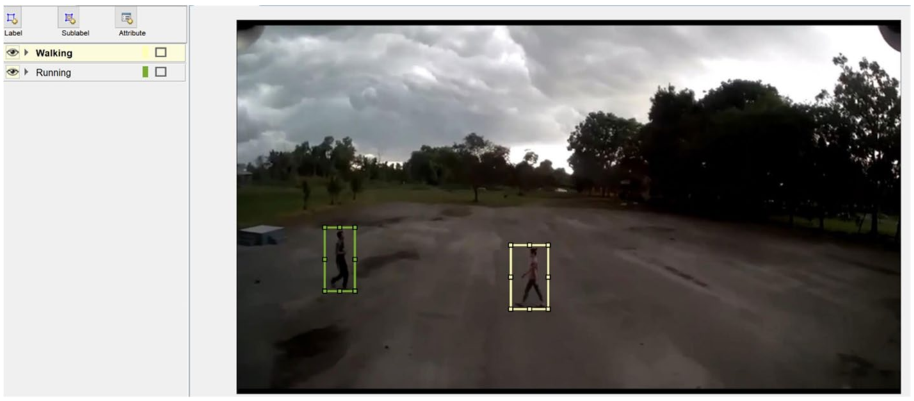


Fig. 9 Snapshot of MATLAB image labeler tool

4 Experimental results

4.1 Data collection

Figure 8 depicts the S30W drone used to acquire numerous photographs for this inquiry. The S30W drone has a control range of 400 m and a WIFI transmission range of 50 m. On the drone, a rotating 720P HD camera was fitted. Two volunteers were filmed as they walked and ran at different speeds and distances (measured in seconds). These films were filmed around the Engineering campus of Universiti Sains Malaysia from a number of view points and in a variety of weather conditions (USM). We selected 300 still images from these films to show the different stages of motion: 200 for walking and 100 for running.

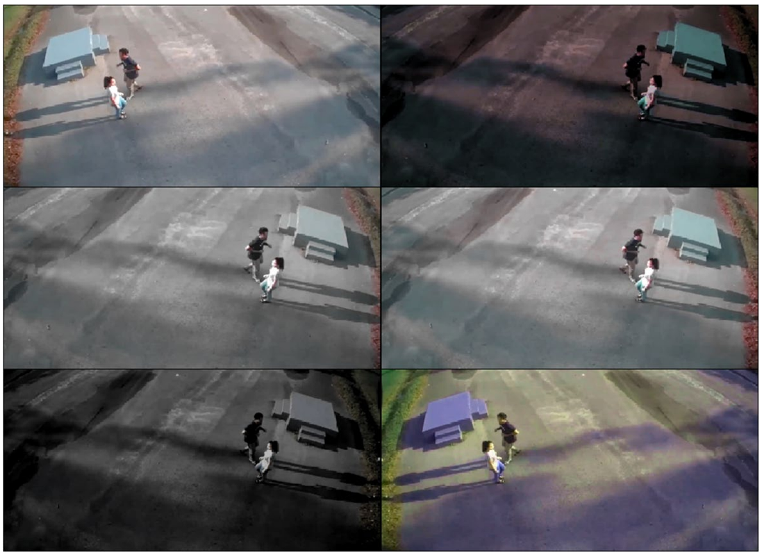


Fig. 10 Data augmentation

Table 3 Experimental settings

Data / Algorithm	Settings
Train / Test	3-fold cross-validation where two folds are used for training and one fold for testing.
No. of TLPSO runs	Ten times for each fold
Augmentation	Each image is randomly rotated, translated, cropped, etc. during training time.

After data collection, the labelling method was executed using the imageLabeler MATLAB tool. Figure 9 depicts an example of this tool.

4.2 Data augmentation

This study use data augmentation to boost the quantity of training data. This technique generates new images by executing various image processing operations, such as translation, rotation, cropping, and random noise addition. Figure 10 contains a collection of some photos for illustrative purposes. Throughout YOLO's training period, the augmentation operation was automatically executed. Consequently, random procedures were used to build an augmented image for each image in the training data. After augmentation, the size of training images will be increased from 200 to 400 images.

4.3 Performance measures

For the purpose of evaluating the optimised YOLO vision system, random data partitioning based on three-fold cross-validation was used. In this method, the experiment is performed three times, with one repetition utilised for testing and the other two for training. In addition, the experiment was performed ten times for each fold to determine the efficacy of

Table 4 Results of YOLO and optimized YOLO

		Action	Precision	Recall	F-score
Fold 1	YOLO	Running	91.34%	87.41%	89.33%
		Walking	93.02%	87.41%	90.13%
	Optimized YOLO	Running	89.42%	88.83%	89.12%
		Walking	90.48%	90.70%	90.59%
Fold 2	YOLO	Running	92.55%	83.87%	88.00%
		Walking	93.42%	91.03%	92.21%
	Optimized YOLO	Running	91.44%	86.40%	88.85%
		Walking	92.11%	92.86%	92.48%
Fold 3	YOLO	Running	91.51%	85.65%	88.48%
		Walking	93.14%	87.96%	90.48%
	Optimized YOLO	Running	89.76%	87.52%	88.63%
		Walking	91.18%	91.00%	91.09%

the TLPSO algorithm. Table 3 provides a summary of the analysis's conditions. In each fold, YOLO must first be trained on training data with augmentation, and then the trained SqueezeNet is subjected to a filters selection/elimination process using TLPSO [15], as previously mentioned. The optimised YOLO will then be retrained in order to achieve additional tuning and performance enhancement. The final evaluation of the fine-tuned/ optimized YOLO will be conducted on the test set. The performance of the proposed YOLO-based vision system was evaluated using the following three metrics.

$$f - scoe = 2 * \frac{precision * recall}{precision + recall} \tag{7}$$

$$recall = \frac{TP}{TP + FP} \tag{8}$$

$$precision = \frac{TP}{TP + FN} \tag{9}$$

where TP is the true positive rate, FP is the false positive rate, and FN is the false-negative rate. These measures have been widely used to assess YOLO performances [40].

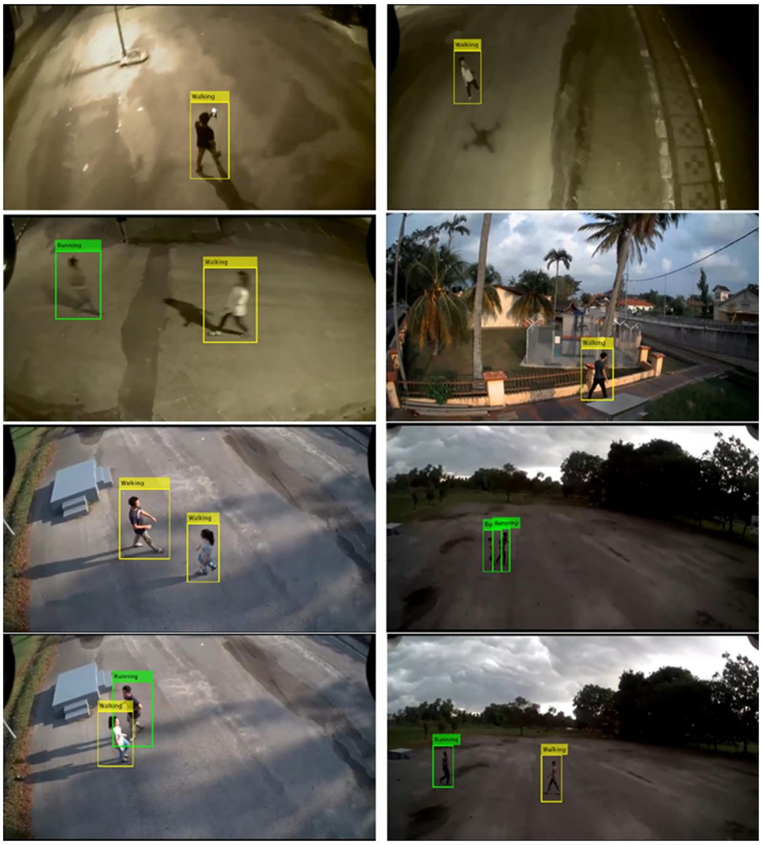
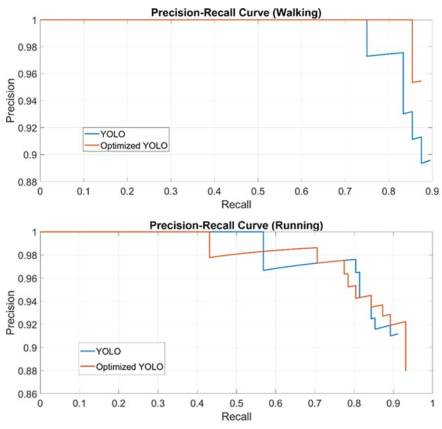


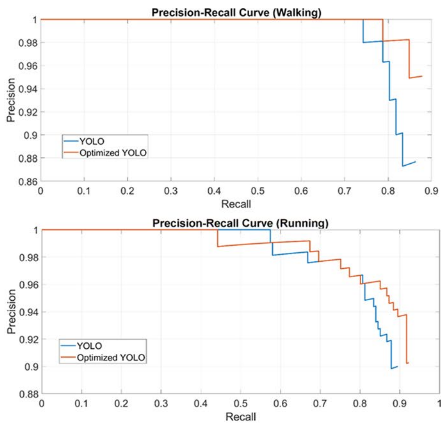
Fig. 11 Successfully recognized cases

4.4 Performance analysis

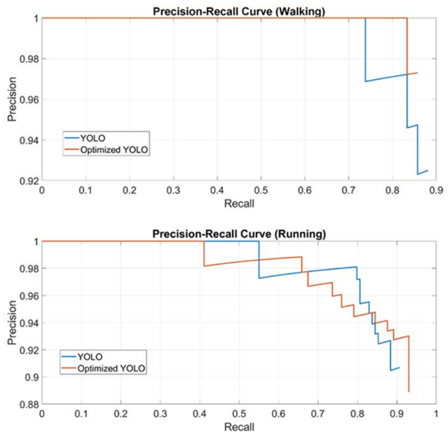
Table 4 contains the outcomes of both the traditional YOLO system and the proposed improved YOLO. The optimised YOLO was able to attain more accuracy in terms of F-score for both actions, i.e., walking and running, with the exception of fold 1. These enhancements are a result of the generalisation of the pruned/optimized SqueezeNet, which possesses fewer convolutional filters than the regular YOLO. Table 4 demonstrates that the proposed optimised YOLO system was able to achieve a balance between precision and recall rate. In fold 1 walking motion, for instance, the conventional YOLO recorded a precision of 94.02% and a recall rate of 87.41%. However, the optimised YOLO was able to balance these metrics with precision and recall scores of 90.48 and 90.70%, respectively. It should be noticed that the mean F-score value for walking action was greater than that for running activity due to the greater number of training samples for walking action. Figure 11 depicts photographs of successfully detected situations as examples.



(a) Fold-1 running and walking actions



(b) Fold-2 running and walking actions



(c) Fold-3 running and walking actions

Fig. 12 The precision-recall curve in three folds

Table 5 Results of filters reduction

	Percentage of filters reduction	Detection time (sec)
YOLO	–	0.417
Optimized YOLO	52%	0.061

Figure 12 illustrates a comparison between the precision-recall curves for walking and running motions. Due to the enhanced SqueezeNet feature extractor's capacity for generalisation, the optimised YOLO creates a better curve than YOLO. Figure 12a demonstrates that the optimised YOLO provided a 100% precision rate in walking with an 85% recall rate, whereas YOLO reported only a 95% precision rate at the same point of recall rate. As seen in Fig. 12, it is important to note that both vision systems reported poorer performance during running (a-c). This is owing to the difficulty of the addressed recognition challenge, as the shape and textural characteristics of these motions are extremely similar. In addition, there are numerous training samples for the walking action.

4.5 Time and complexity analysis

Additional investigation was undertaken by incorporating the developed TLPSO method and computing the speedup analysis [37]. In Table 5, the detection time in seconds and the filter reduction percentage have been calculated and tabulated. Note that both systems were implemented in MATLAB and tested on the same PC with Windows 10, an i7–8700 processor running at 3.2 GHz, and 32 GB of RAM. According to Table 5, the embedded optimizer was able to reduce around 52% of SqueezeNet filters. Importantly, the enhanced YOLO algorithm described required only 0.061 seconds per image. In other words, the optimizer was able to accelerate the YOLO vision system by a factor of seven. This is due to the advantage of removing less contributed SqueezeNet filters that incur additional computational time costs during the feature extraction procedure. Notably, the training/optimization time of TLPSO is not addressed here because it is only required during the construction of the YOLO system.

Table 6 Optimizers parameter settings

Optimizer	Settings
PSO [16]	Population size 30, c1 = 1.4 and c2 = 1.4, w:0.9–0.4
RLMPSO [36]	Population size 3 c1 = 2.5 in exploration mode, c1 = 0.5 in exploitation mode, c2 = 0.5 in exploration mode, c2 = 2.5 in exploitation mode, and w = 0.9 in exploration mode, w-0.4 in exploitation mode.
TLPSO [37]	Population size 3 c1 = 2.5 in exploration mode, c1 = 0.5 in exploitation mode, c2 = 0.5 in exploration mode, c2 = 2.5 in exploitation mode, and w = 0.9 in exploration mode, w-0.4 in exploitation mode.

Table 7 Results of optimizers comparison

	Fitness	Percentage of filters reduction
RLMPSO [36]	−0.0020	48%
PSO [16]	−0.0032	50%
TLPSO [37]	−0.0059	52%

4.6 Compare with other optimizers

Comparing the outcomes of TLPSO against those of two comparable optimizers, namely PSO [16] and RLMPSO [36], yielded more insights. The parameters for these algorithms are listed in Table 6. All algorithms were run ten times with a maximum of 100 iterations, and Table 7 displays the results produced by each strategy in terms of the mean fitness function and the percentage of filter reduction. In terms of the mean fitness value, it is evident that the suggested TLPSO optimizer outperformed other optimizers. As described in [37], this is owing to the advantages of a small population size (3 particles) and a local search layer. TLPSO states that the height reduction reached 52% as measured by the percentage of reduction. Due to the large cost of fine-tuning operations spent by the local search optimizer, RLMPSO achieved the worst results [36]. Boxplots were used to undertake additional analysis, as shown in Fig. 13. It can be shown that the TLPSO achieves significantly better outcomes than other optimizers based on the range of reported fitness values in each run.

The TLPSO convergence curve was compared to those of PSO and RLMPSO in Fig. 14. This curve illustrates TLPSO's behaviour during operation. It depicts the base-10 logarithmic mean values generated by ten iterations of each algorithm for each fold. TLPSO has a better convergence curve in all folds, as can be observed. This is owing to TLPSO's capacity to manage large-scale problems, as described in [37].

The Wilcoxon statistical test [9] has been used to examine the efficiency of TLPSO from a statistical standpoint. The essential concept of this test is that the null hypothesis H_0 implies that all optimizers have comparable performance and that their means are selected from the same distribution. Yet, according to the alternative hypothesis

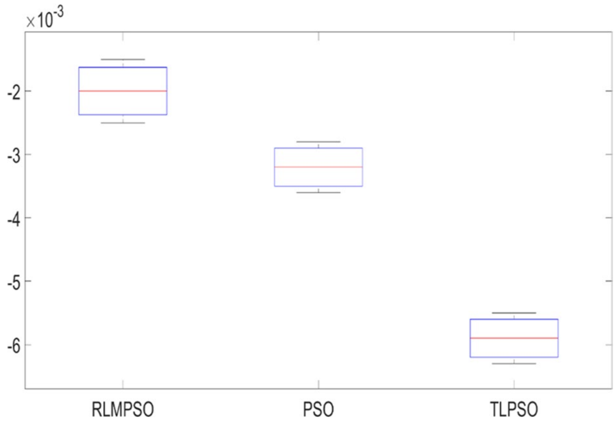
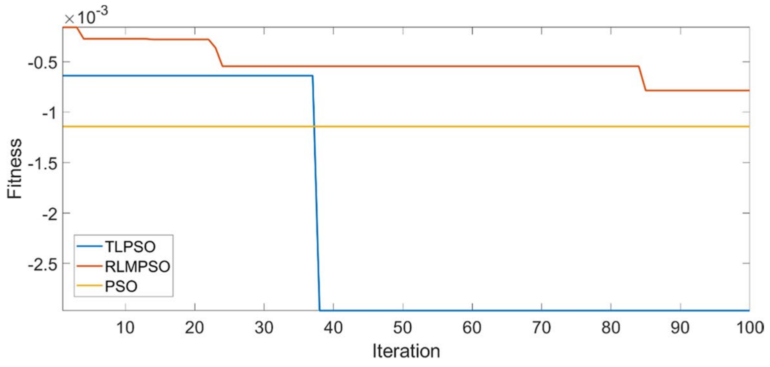
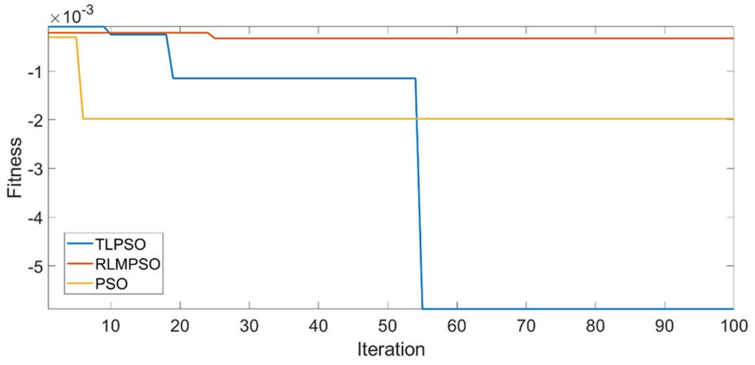


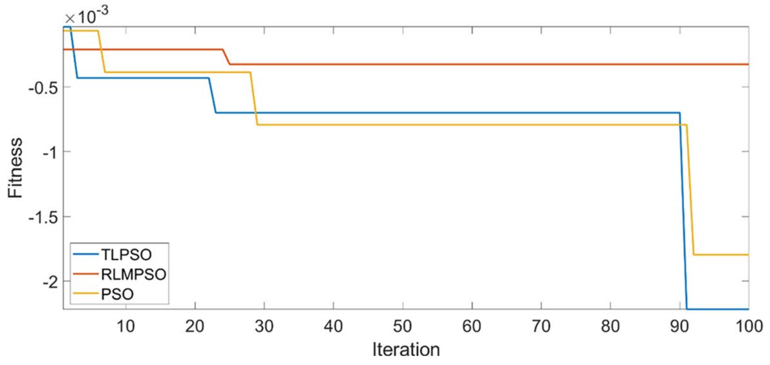
Fig. 13 Box plot of fitness value



Fold #1



Fold #2



Fold #3

Fig. 14 The convergence curve analysis of TLPSO, PSO, and RLMPSO

H_1, they have a different distribution. The *p value* is set to 0.05, therefore the alternative hypothesis H_1 is accepted if the p value is less than 0.005 (95% confidence level). The results of the statistical study are presented in Table 8, where it is evident that TLPSO statistically outperforms other algorithms with a *p value* less than 0.005. This is due to the advantages of a limited population size and the capacity to handle large-scale optimisation issues [37].

Table 8 Statistical test analysis

	RLMPSO [36]	PSO [16]
p value	0.0021	0.0302

4.7 Evaluation using public dataset

In this section, the improved YOLO vision system is applied to a publicly available benchmark dataset for identifying people in drone-captured images. The UAV123 [23] dataset is utilised, which includes 3000 photographs. The findings of the improved YOLO algorithm suggested are compared with those of other methods published in [10]. These results are displayed in Table 9. In terms of precision, recall, and F-score, it is obvious that the modified YOLO method proposed produced the best results. Due to the lower network complexity of SqueezeNet, its generalisation performance is superior to that of YOLO and Faster R-CNN.

5 Conclusion, limitation and future directions

In this study, we present a YOLO-based deep learning vision system that employs the efficient TLPSO algorithm to remove ineffective SqueezeNet filters. Improved YOLO vision technology has been incorporated into drones, allowing them to detect human motions like walking and running. The presented results confirmed the superior performance gains achieved by the proposed upgraded YOLO vision system. With TLPSO, we were able to get a 7X improvement in detection speed over the baseline unoptimized YOLO vision system, and we were also able to get rid of 52% of SqueezeNet filters. With a drop in detection time from 0.417 seconds to 0.061 seconds, the benefits of the integrated TLPSO algorithm are verified. TLPSO also achieves a higher mean fitness value than PSO and RLMSPO, and it does so at a considerably faster rate of convergence. The proposed approach's lightweight architecture allows it to surpass previously published human recognition findings from drone photographs (i.e. the UAV123 dataset).

These are some of the key restrictions that might be placed on this study including; (i) the optimization task is a time-consuming process that requires sufficient hardware support (ii) the formulated fitness function is complicated where it combines the accuracy with the complexity, and (iii) SqueezeNet is not efficient as compared to other pre-trained CNN such as Mobilenet. Extending TLPSO into a multiobjective method, optimizing additional pre-trained networks, and implementing parallel that might speed up the optimization effort are just a few of the numerous concepts that could be utilized to further improve this work. Other real-world computer vision issues that the suggested improved YOLO vision system might tackle include vehicle identification, face recognition, and industrial flaw detection.

Table 9 Results of the UAV123 dataset [23]

	YOLO [10]	Faster R-CNN [10]	Optimized YOLO
Precision	97.12%	97.70%	98.02%
Recall	90.34%	90.50%	91.50%
F-score	93.61%	93.96%	94.65%

Acknowledgements The authors would like to acknowledge the support provided by Saudi Data & AI Authority (SDAIA) and King Fahd University of Petroleum & Minerals (KFUPM) under SDAIA-KFUPM Joint Research Center for Artificial Intelligence (JRC-AI) grant No. JRCAI-RG-07.

Declarations

Conflict of interest Hussein Samma and Ali Salem Bin Sama declare that they have no conflict of interest.

References

1. Amudhan AN, Sudheer AP (2022) Lightweight and computationally faster Hypermetropic Convolutional Neural Network for small size object detection. Image Vis Comput 119:104396
2. Bansal M, Kumar M, Kumar M (2021) 2D object recognition: a comparative analysis of SIFT, SURF and ORB feature descriptors. Multimed Tools Appl 80:18839–18857
3. Bansal M, Kumar M, Sachdeva M, Mittal A (2021) Transfer learning for image classification using VGG19: Caltech-101 image data set. J Ambient Intell Humaniz Comput:1–12
4. Bao W, Zhu Z, Hu G, Zhou X, Zhang D, Yang X (2023) UAV remote sensing detection of tea leaf blight based on DDMA-YOLO. Comput Electron Agric 205:107637
5. Bie M, Liu Y, Li G, Hong J, Li J (2023) Real-time vehicle detection algorithm based on a lightweight You-Only-Look-Once (YOLOv5n-L) approach. Expert Syst Appl 213:119108
6. Cui M, Lou Y, Ge Y, Wang K (2023) LES-YOLO: A lightweight pinecone detection algorithm based on improved YOLOv4-Tiny network. Comput Electron Agric 205:107613
7. Hammam AA, Soliman MM, Hassanien AE (2020) Real-time multiple spatiotemporal action localization and prediction approach using deep learning. Neural Netw 128:331–344
8. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
9. Hettmansperger TP, McKean JW (2010) Robust nonparametric statistical methods. CRC Press
10. Hung GL, Bin Sahimi MS, Samma H, Almohamad TA, Lahasan B (2020) Faster R-CNN deep learning model for pedestrian detection from drone images. SN Comput Sci 1(2):1–9
11. Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016) SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv Prepr. arXiv1602.07360
12. Jiang K et al (2022) An attention mechanism-improved YOLOv7 object detection algorithm for Hemp Duck Count Estimation. Agriculture 12(10):1659
13. Jintasuttisak T, Edirisinghe E, Elbattay A (2022) Deep neural network based date palm tree detection in drone imagery. Comput Electron Agric 192:106560
14. Junos MH, Khairuddin ASM, Dahari M (2022) Automated object detection on aerial images for limited capacity embedded device using a lightweight CNN model. Alex Eng. J. 61(8):6023–6041
15. Kang Q, Zhao H, Yang D, Ahmed HS, Ma J (2020) Lightweight convolutional neural network for vehicle recognition in thermal infrared images. Infrared Phys Technol 104:103120
16. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, vol 4, pp 1942–1948
17. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 25:1097–1105
18. Li R, Shen Y (2023) YOLOSR-IST: a deep learning method for small target detection in infrared remote sensing Images based on super-resolution and YOLO. Signal Process:108962
19. Li Y, Yuan H, Wang Y, Xiao C (2022) GGT-YOLO: a novel object detection algorithm for drone-based maritime cruising. Drones 6(11):335
20. Liu C, Li X, Li Q, Xue Y, Liu H, Gao Y (2020) Robot recognizing humans intention and interacting with humans based on a multi-task model combining ST-GCN-LSTM model and YOLO model. Neurocomputing
21. Maitre J, Bouchard K, Bertuglia C, Gaboury S (2021) Recognizing activities of daily living from UWB radars and deep learning. Expert Syst Appl 164:113994
22. Mliki H, Bouhlel F, Hammami M (2020) Human activity recognition from UAV-captured video sequences. Pattern Recognit 100:107140
23. Mueller M, Smith N, Ghanem B (2016) A benchmark and simulator for uav tracking. In: European conference on computer vision, pp 445–461

24. Murugesan M, Arieth RM, Balraj S, Nirmala R (2023) Colon cancer stage detection in colonoscopy images using YOLOv3 MSF deep learning architecture. Biomed Signal Process Control 80:104283
25. Mutis I, Ambekar A, Joshi V (2020) Real-time space occupancy sensing and human motion analysis using deep learning for indoor air quality control. Autom Constr 116:103237
26. Patel K, Bhatt C, Mazzeo PL (2022) Improved ship detection algorithm from satellite images using YOLOv7 and graph neural network. Algorithms 15(12):473
27. Peng H, Razi A (2020) Fully autonomous UAV-based action recognition system using aerial imagery. In: International symposium on visual computing, pp 276–290
28. Pham D-L, Chang T-W et al (2023) A YOLO-based real-time packaging defect detection system. Procedia Comput Sci 217:886–894
29. Polsinelli M, Cinque L, Placidi G (2020) A light cnn for detecting covid-19 from ct scans of the chest. Pattern Recognit Lett 140:95–100
30. Qi C, Gao J, Pearson S, Harman H, Chen K, Shu L (2022) Tea chrysanthemum detection under unstructured environments using the TC-YOLO model. Expert Syst Appl 193:116473
31. Qiao Y, Guo Y, He D (2023) Cattle body detection based on YOLOv5-ASFF for precision livestock farming. Comput Electron Agric 204:107579
32. Qiu Q, Lau D (2023) Real-time detection of cracks in tiled sidewalks using YOLO-based method applied to unmanned aerial vehicle (UAV) images. Autom Constr 147:104745
33. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788
34. Ren P, Wang L, Fang W, Song S, Djahel S (2020) A novel squeeze YOLO-based real-time people counting approach. Int J Bio-Inspired Comput 16(2):94–101
35. Saha S, Singh G, Sapienza M, Torr PHS, Cuzzolin F (2016) Deep learning for detecting multiple space-time action tubes in videos. arXiv Prepr. arXiv1608.01529
36. Samma H, Lim CP, Mohamad Saleh J (2016) A new reinforcement learning-based memetic particle swarm optimizer. Appl Soft Comput J 43:276–297. https://doi.org/10.1016/j.asoc.2016.01.006
37. Samma H, Suandi SA, Mohamad-Saleh J (2020) Two-layers particle swarm optimizer. In: 2020 IEEE international conference on automatic control and intelligent systems (I2CACIS), pp 165–169
38. Sha M, Boukerche A (2022) Performance evaluation of CNN-based pedestrian detectors for autonomous vehicles. Ad Hoc Networks 128:102784
39. Shaheed K, Mao A, Qureshi I, Kumar M, Hussain S, Zhang X (2022) Recent advancements in finger vein recognition technology: methodology, challenges and opportunities. Inf Fusion 79:84–109
40. Shinde S, Kothari A, Gupta V (2018) YOLO based human action recognition and localization. Procedia Comput Sci 133:831–838
41. Souza BJ, Stefenon SF, Singh G, Freire RZ (2023) Hybrid-YOLO for classification of insulators defects in transmission lines based on UAV. Int J Electr Power Energy Syst 148:108982
42. Szegedy C, Ioffe S, Vanhoucke V, Alemi A (2017) Inception-v4, inception-resnet and the impact of residual connections on learning. In: Proceedings of the AAAI conference on artificial intelligence, vol 31
43. Tang Y, Zhou H, Wang H, Zhang Y (2023) Fruit detection and positioning technology for a Camellia oleifera C. Abel orchard based on improved YOLOv4-tiny model and binocular stereo vision. Expert Syst Appl 211:118573
44. Triki A, Bouaziz B, Mahdi W (2022) A deep learning-based approach for detecting plant organs from digitized herbarium specimen images. Ecol Inform 69:101590
45. Ullah A, Muhammad K, Ding W, Palade V, Haq IU, Baik SW Efficient activity recognition using lightweight CNN and DS-GRU network for surveillance applications. Appl Soft Comput:107102
46. Varga LA, Kiefer B, Messmer M, Zell A (2022) Seadronessee: A maritime benchmark for detecting humans in open water. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision, pp 2260–2270
47. Watkins CJCH, Dayan P (1992) Q-learning. Mach Learn 8(3–4):279–292
48. Wei X, Wei Y, Lu X (2023) HD-YOLO: Using radius-aware loss function for head detection in top-view fisheye images. J Vis Commun Image Represent 90:103715
49. Wu P, Li H, Zeng N, Li F (2022) FMD-Yolo: an efficient face mask detection method for COVID-19 prevention and control in public. Image Vis Comput 117:104341
50. Xia R, Li G, Huang Z, Meng H, Pang Y (2023) Bi-path combination YOLO for real-time few-shot object detection. Pattern Recognit Lett 165:91–97
51. Yang Y et al (2020) A lightweight deep learning algorithm for inspection of laser welding defects on safety vent of power battery. Comput Ind 123:103306
52. Yao Y, Han L, Du C, Xu X, Jiang X (2022) Traffic sign detection algorithm based on improved YOLOv4-Tiny. Signal Process Image Commun 107:116783

53. Zeng T, Li S, Song Q, Zhong F, Wei X (2023) Lightweight tomato real-time detection method based on improved YOLO and mobile deployment. Comput Electron Agric 205:107625
54. Zhang L, Lim CP, Yu Y (2021) Intelligent human action recognition using an ensemble model of evolving deep networks with swarm-based optimization. Knowl-Based Syst. 220:106918
55. Zhao H, Zhang H, Zhao Y (2023) Yolov7-sea: Object detection of maritime uav images based on improved yolov7. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision, pp 233–238
56. Zhou H, Jiang F, Lu H (2023) SSDA-YOLO: Semi-supervised domain adaptive YOLO for cross-domain object detection. Comput Vis Image Underst:103649