Applied Intelligence (2023) 53:15603–15620
https://doi.org/10.1007/s10489-022-04299-1

Hyper-parameter optimization of deep learning architectures using artificial bee colony (ABC) algorithm for high performance real-time automatic colorectal cancer (CRC) polyp detection

Ahmet Karaman[1] · Dervis Karaboga[2,3] · Ishak Pacal[3,4] · Bahriye Akay[2,3] · Alper Basturk[2,3] · Ufuk Nalbantoglu[2,3] · Seymanur Coskun[1] · Omur Sahin[2,3]

Accepted: 27 October 2022 / Published online: 23 November 2022

Abstract

Colorectal cancer (CRC) is one of the most common and malignant types of cancer worldwide. Colonoscopy, considered the gold standard for CRC screening, allows immediate removal of polyps, which are precursors to CRC. Many computer-aided diagnosis systems (CADs) have been proposed for automatic polyp detection. Most of these systems are based on traditional machine learning algorithms and their generalization ability, sensitivity and specificity are limited. On the other hand, with the widespread use of deep learning algorithms in medical image analysis and the successful results in the analysis of colonoscopy images, especially in the early and accurate detection of polyps, these problems are eliminated in recent years. In short, deep learning algorithms and applications have gained a critical role in CAD systems for real-time autonomous polyp detection. Here, we make significant improvements to object detection algorithms to improve the performance of CAD-based real-time polyp detection systems. We integrate the artificial bee colony algorithm (ABC) into the YOLO algorithm to optimize the hyper-parameters of YOLO-based algorithms. The proposed method can be easily integrated into all YOLO algorithms such as YOLOv3, YOLOv4, Scaled-YOLOv4, YOLOv5, YOLOR and YOLOv7. The proposed method improves the performance of the Scaled-YOLOv4 algorithm with an average of more than 3% increase in mAP and a more than 2% improvement in F1 value. In addition, the most comprehensive study is conducted by evaluating the performance of all existing models in the Scaled-YOLOv4 algorithm (YOLOv4s, YOLOv4m, YOLOV4-CSP, YOLOv4-P5, YOLOV4-P6 and YOLOv4-P7) on the novel SUN and PICCOLO polyp datasets. The proposed method is the first study for the optimization of YOLO-based algorithms in the literature and makes a significant contribution to the detection accuracy.

Keywords Colorectal cancer · Deep learning · Hyper-parameter optimization · Artificial bee colony · Real-time polyp detection · YOLOv4

1 Introduction

Colorectal cancer (CRC) is the second leading cause of cancer death in the world after lung cancer, and is the third most frequently diagnosed cancer in the world [1]. Two years ago, CRC was the third mortal cancer in the world and the fourth most commonly diagnosed cancer [2].

Approximately 2 million new cases and approximately 1 million deaths were expected in 2020 [1]. This shows that CRC is one of the most important and fatal cancers with an increasing mortality and frequency. In the light of this information, it can be said that the diagnosis and treatment of CRC is of critical importance. Polyps that are precursors to CRC begin as glandular tissue in the inner lining of the colon or

✉ Ishak Pacal
Ishakpacal@gmail.com

1 Department of Gastroenterology, Acıbadem Hospital, Kayseri, Turkey

2 Department of Computer Engineering, Engineering Faculty, Erciyes University, Kayseri, Turkey

3 Artificial Intelligence and Big Data Application and Research Center, Erciyes University, Kayseri, Turkey

4 Department of Computer Engineering, Engineering Faculty, Igdir University, Igdir, Turkey

rectum. Early detection of polyps can significantly prevent the development of CRC. Colonoscopy is considered as the gold standard in detecting CRC and allows immediate removal of polyps. However, the success of polyp detection in colonoscopy screening varies greatly depending on the expertise of the endoscopist and the techniques they apply. High-quality colonoscopy and the required adenoma detection rate (the proportion of procedures in which at least one or more adenomas are detected) are the cornerstones of CRC prevention. Studies have been presented showing that for every 1.0% increase in Adenoma Detection rate (ADR), there is at least a 6.0% reduction in the risk of CRC [3, 4].

Computer aided diagnosis (CAD) systems have been developed to detect polyps missed during colonoscopy screening and to increase ADR. CAD systems developed for polyp detection generally used classical machine learning algorithms. Since these algorithms lack generalization, they could not provide the desired success because colorectal polyps show great differences in size, direction, color and texture. On the other hand, the use of deep learning in the field of medical image analysis has started and attracted great attention recently, as deep learning algorithms and architectures have started to offer superior performances in object detection and image classification [5]. Deep learning is a subfield of machine learning and artificial intelligence that focuses on learning high-level abstractions aimed at discovering new features in data using hierarchical structures. Deep learning has attracted great interest in many popular areas such as health, defense industry, object recognition, self-driving cars and speech recognition [6–9]. The reason behind the high performance of deep learning belongs to CNN architectures. CNNs contain operators called convolutions and these operators try to optimize feature extraction by using filters of different structures. Almost all the CAD-based approaches proposed for polyp detection consist of CNN-based object detection algorithms [10]. These systems offer the highest performance for real-time polyp detection.

Deep learning methods are among the most popular methods used both in the diagnosis of CRC and in the diagnosis of other cancers [10–18]. The main reason why deep learning methods are popular in cancer diagnosis today is the use of large amounts of data in hospitals and research centers with permission. Data labeled by a few experts can be successfully applied to deep learning environments. CNN-based object detection algorithms, one of the deep learning methods, have achieved more successful results in automatic polyp detection compared to other methods. Especially, YOLO [19] and R-CNN-based [20] algorithms have created an intense research era for polyp detection. Adaptation of deep learning methods to polyp detection in real time is more convenient for CAD systems. The polyp detection system that can be integrated into the endoscopy device must provide real-time and high-level accuracy. Therefore, YOLO-based, SSD-based [21], CenterNet-based [22] or custom models, which are single-stage approaches, are more successful in this sense than two-stage approaches because it is seen that studies based on YOLO object detection algorithm have been studied more. YOLO-based algorithms have become even more popular with the release of new versions. With the accelerated increase in real-time performance and detection accuracy, it has earned its place in many areas.

The paper is organized as follows. Section 2 illustrates the works related to CRC polyp detection and hyper-parameter optimization of deep learning algorithms. Section 3 explains the details of the proposed method and datasets. Section 4 presents the experiments, which contains the setup, experimental results and comparisons. Finally, some remarks and conclusions are given in Section 5.

1.1 Contributions

The main purpose of this study is to strengthen the performance of CNN-based commonly used object detection algorithms through ABC, a simple and powerful swarm-based global optimization algorithm for automatic polyp detection. The main contributions of this study are as follows.

A comprehensive comparison of the performance of the Scaled-YOLOv4 algorithm on the novel SUN and PICCOLO polyp datasets is presented. For this, all existing models in the Scaled-YOLOv4 algorithm are trained and evaluated. These models are respectively; YOLOv4s, YOLOv4m, YOLOv4l (YOLOv4-CSP), YOLOv4-P5, YOLOv4-P6, and YOLOv4-P7. Thus, the effect of scalability in object detection algorithms on polyp detection was investigated. Such a comprehensive study has the distinction of being the first study in the literature according to our knowledge.

We utilize the ABC algorithm, a swarm-based global optimization algorithm, for automatic hyper-parameter optimization of YOLO object detection algorithms. Therefore, we significantly increase the polyp detection performance of YOLO algorithms. The ABC algorithm integrated on the Scaled-YOLOv4 algorithm could be quickly adapted to all YOLO-based algorithms such as YOLOv3, YOLOv4, YOLOv5, YOLOR and YOLOv7. This is the first study in the literature for hyper-parameter optimization of YOLO-based algorithms.

The proposed method is performed on the novel SUN and PICCOLO polyp datasets. Within both datasets, polyp detection accuracy (mAP, F1) is improved by at least 3%. It offers data-specific optimization automatically according to the images in the data, by presenting the most ideal hyper-parameters for each dataset. In addition, it provides significant time and cost savings.

This study demonstrates the importance of hyper-parameter optimization in polyp detection or other object detection fields and encourages researchers for new research.

2 Related works

2.1 Deep learning for polyp detection

Deep learning methods based on CNNs are the most widely used approaches for object detection as well as automatic polyp detection due to their excellent feature representation capacity and data-hungry characteristics. Colonic or colorectal polyp detection can be considered as object detection. Object detection, on the other hand, is a computer vision technique for finding instances of objects in images or videos. It is seen that all studies on polyp detection are based on object detection algorithms, which is a part of deep learning architectures. YOLO-based algorithms, single stage approaches, are the most popular part of object detection algorithms. YOLO algorithms are quickly separated from their counterparts in terms of both the high accuracy and the real-time performance.

In recent studies, it is seen that new mechanisms have been added to object detection algorithms to increase performance in real-time automatic polyp detection, and deep learning models have been integrated into CAD systems for this manner. Some recent studies can be summarized as follows. Liew et al. [23] proposed a deep CNN-based system for the classification of colonic polyps. This system includes a new combination of deep residual convolutional neural networks modified with PCA and ensemble learning approach to increase the performance of polyp classification. Thus, while the classification performance of polyps increased, the calculation time of the model was also improved. Younas et al. [24] compared six existing Convolutional Neural Networks in addition to transfer learning for classification of polyp types, and then applied ensemble learning to select the optimal performing architecture, presenting a more efficient method of polyp classification. Lee et al. [25] proposed a real-time polyp detection system based on YOLOv4. In this system, a multi-scale mesh was used to detect small polyps. In addition, the performance of polyp detection has been increased by using advanced data augmentation techniques and different activation functions. Nogueira-Rodriguez et al. [26] presented a deep learning model capable of automatic polyp detection based on YOLOv3. The proposed method is enhanced by an object tracking step that aims to reduce false positives. The performance of the model has been increased by training the proposed system with a special dataset containing a large amount of polyp images.

Hoang et al. [27] presented a capsule endoscope system for small bowel and colon applications with 5D position sensing as well as real-time automatic polyp detection. This system uses a YOLOv3-based algorithm to detect real-time polyps, and in this way, the system performs with an average precision of 85%. Wan et al. [28] proposed the YOLOv5-based model for a real-time polyp detection based on the self-attention mechanism. With the proposed method, while the beneficial features are strengthened, the weak features are weakened, and the performance of polyp detection is increased. Pacal and Karaboga [29] presented a robust system that can be integrated into endoscopy devices. This system is based on scaling the YOLOv4 algorithm and deploying models to increase inference speed and performance. This system outperforms previous studies on polyp detection accuracy on Etis-Larib and CVC-ColonDB datasets. Pacal et al. [30] evaluated SUN and PICCOLO datasets, which are novel datasets, using the Scaled-YOLOv4 algorithm in a large experimental study. Experimental studies show that SUN and PICCOLO datasets are very successful in detecting polyps, while the Scaled-YOLOv4 algorithm is one of the most ideal object detection algorithms for large-scale datasets.

Souaidi et al. [31] proposed a hybrid SSDNet-based method for polyp detection. This method aimed to model the visual appearance of small polyp areas. Additionally, instead of using simple convolution layers, it employed modified initial-A modules to augment intermediate feature maps. In the experiments, the polyp detection performance of the proposed method was verified with public datasets. Durak et al. [32] trained the state-of-the-art object detection algorithms, such as YOLOv4 [33], CenterNet, EfficientDet [34], YOLOv3 [35], SSD, and Faster R-CNN [36] for automatic gastric polyp detection. In the experimental results, YOLOv4 algorithm provided highest performance compared to other methods and could be used effectively in CAD systems for the automatic polyp detection purpose. Qian et al. [37] proposed a method that combines GAN architectures and YOLOv4 object detection algorithm for robust polyp detection. Experimental results of this study were evaluated on 3 publicly available datasets. Experimental results showed that the proposed method outperforms U-Net and can synthesize more realistic polyp images. In addition, the polyp detection performance has been significantly improved with this method.

2.2 Hyper-parameter optimization of deep learning algorithms

In deep learning-based CAD systems, method and dataset are the two most important factors that affect the performance of the system. The effective use of these two fundamental factors is of particular importance. While most studies on polyp detection generally use the latest up-to-date methods and larger datasets, their effective use, in other words fine-tuning or hyper-parameter optimization, has not been considered. One of the main reasons for this is that hyper-parameter optimization requires high processing power. However, performance is of greater importance, especially in CAD systems used in cancer diagnosis. Recently, many studies have been presented for hyper-parameter optimization in different fields, and it has been noted that performance has been improved in most studies. A good review article has been recently published that delves into this topic [38]. However, very simple hyper-parameters of CNN were tried to be optimized by population-based metaheuristic algorithms in the studies.

Artificial bee colony (ABC) algorithm [39], which is one of the well-known population-based algorithms, has been used in several recent studies to optimize deep learning architectures: Erkan et al. [40] proposed a hyperparameter optimization method for the identification of plant species from leaf images, using the ABC algorithm based on the optimization of CNN architectures. In the proposed method, the ABC algorithm found the most ideal values in some simple structures of the CNN architecture, making the classification more successful. Banharnsakun et al. [41] employed ABC algorithm to optimize the performance of CNN. In this study, ABC algorithm was used to minimize classification errors by optimally initializing the weights of the CNN classifier. Ozcan et al. [42] performed hyper-parameter optimization for AlexNet [43], which is one of the basic CNN architectures, by using the ABC algorithm. In this study, sign language numbers were used, and AlexNet with ABC showed higher performance. Badem et al. [44] presented a study combining the ABC and L-BFGS method for tuning the parameters of a DNN. The DNN architecture used in this study aimed to increase the performance of the system by including one or more Autoencoder (AE) layers cascaded into a softmax layer. Hyper-parameter optimization in object detection algorithms has been applied for the first time using genetic algorithm (GA) [45] in the YOLOv3-SPP algorithm to optimize 20 hyper-parameters [46]. In the YOLOv4 algorithm, it is aimed to find the most optimal hyper parameters using GA. However, in these studies, there is no clear information about how much the GA increases the performance. Pacal and Karaboğa [29] employed GA for hyper-parameter optimization together with the Scaled-YOLOv4 [47] algorithm to increase the performance in real-time automatic polyp detection. In the study, the performance of polyp detection was increased by optimizing 22 hyper-parameters and the model reached convergence in a shorter time.

3 Methods

3.1 Scaled-YOLOv4 algorithm

YOLO object detection algorithms are frequently used, especially in automatic polyp detection. Model scaling technique is very important in the design of an effective real-time automatic polyp detection system, because the object detection detector must include vital features such as integration into the system, real-time operation and high accuracy. The YOLOv4 algorithm is one of the most up-to-date object detection algorithms that includes improvements in both accuracy and inference speed with the addition of some features such as Cross Stage Partial Network (CSPNet) [48] architecture, mish activation function and mosaic data augmentation to the YOLOv3 algorithm. Scaled-YOLOv4 is created by redesigning the YOLOv4 algorithm. Changing the depth and width of the backbone is one of the most common scaling techniques in object detection algorithms. Thus, the number of convolutional layers in the CNN and the number of convolutional filters in a convolutional layer are changed according to different hardware requirements. The Scaled-YOLOv4 algorithm is obtained by redesigning the YOLOv4 algorithm according to these points. By rescaling YOLOv4, the YOLOv4-tiny model (suitable for low-end GPUs), YOLOv4-CSP model (suitable for general GPUs) and YOLOv4 large model (suitable for high-end GPUs or cloud computing) are systematically obtained and are shown in Fig. 1.

The main purpose of the scaling technique in Scaled-YOLOv4 is to obtain improvements in cost along with the accuracy of the model. Therefore, quantitative costs were analyzed according to changes in image size, number of layers and number of channels in YOLOv4 scaling technique. CNN architectures such as ResNet [49], ResNext and DarkNet were used for these analyses. In experiments on ResLayer, ResXlayer and Darklayer, it has been observed that scaling size, depth and width cause an increase in computational costs. The authors examined other techniques to effectively scale the YOLOv4 network. A significant reduction in computational cost has been achieved by using CSPNet architecture in the YOLOv4 algorithm. CSPNet reduces the amount of computation and inference time, while increasing the accuracy and learning ability of CNN. It is a partial inter-stage network that integrates feature maps from the beginning to the end of a network stage to account for changes in gradients. CSPNet tries to offer a rich combination of gradients to improve learning.

The aim of the CSPNet architecture is to divide the base layer's feature map into two branches, one of which passes through several layers for further development while the other skips these layers. Thus, the gradient information is differentiated, and a richer combination is obtained. As a result, an effective reduction in memory cost and inference time is achieved, while the accuracy performance of the object detection model is increased. The CSPNet structure for ResNet is shown in Fig. 2. According to the Scaled-YOLOv4 algorithm, it was observed that after the CNNs were converted to CSPNet, the new architecture effectively reduced the number of computations (FLOPs) on ResNet, ResNeXt and Darknet by 23.5%, 46.7% and 50.0%. Thus, the YOLOv4 algorithm can be scaled effectively with the CSPNet architecture. In the Scaled-YOLOv4 algorithm, kernel > 1 is required for the CSPDarknet stage to have a better computational advantage over the DarkNet stage. Therefore, unlike YOLOv4, the first

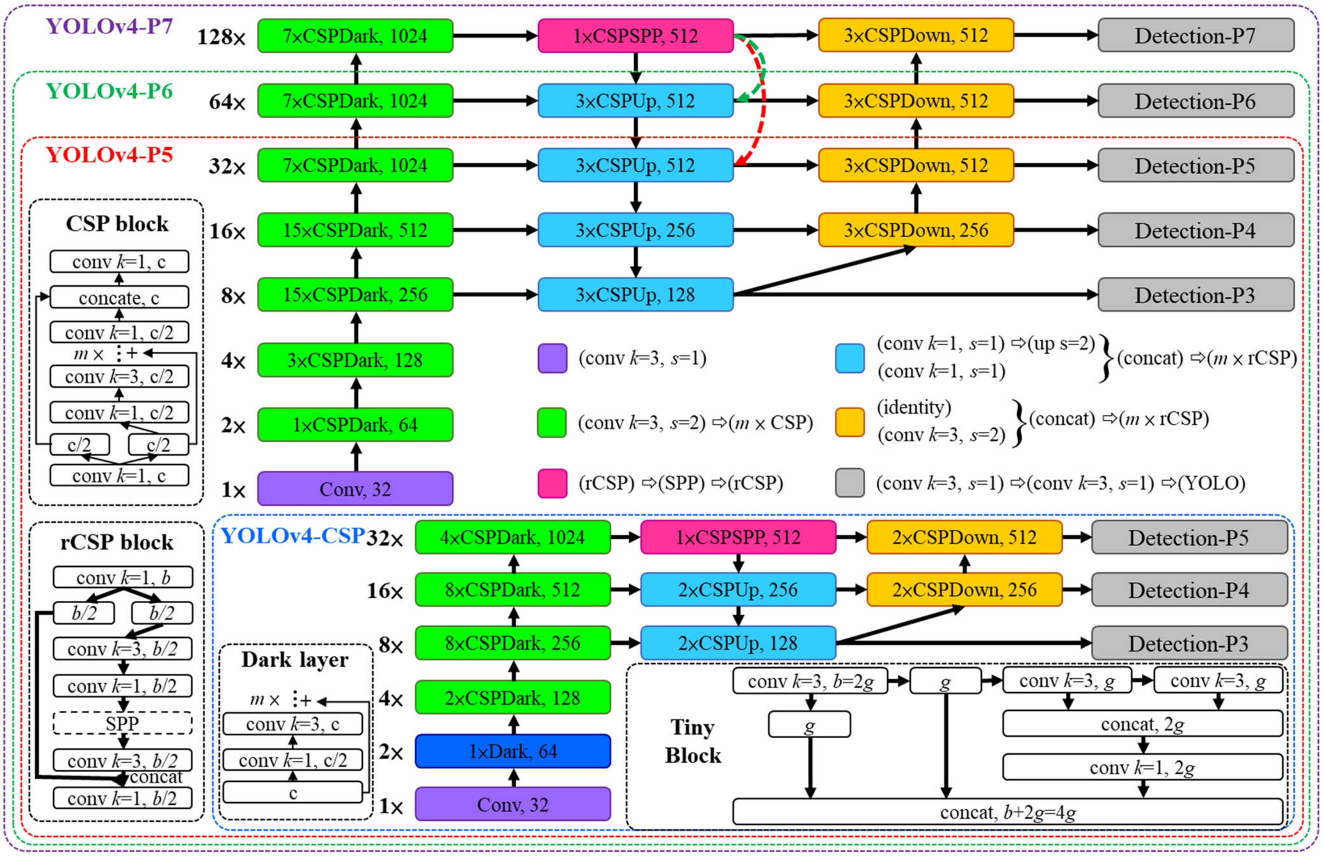

Fig. 1 Architecture of Scaled-YOLOv4 object detection algorithm

block in the backbone is transformed into a DarkNet residual layer, and all the remaining blocks (as seen in Figs. 1 and 2) are used together with the CSPNet and DarkNet residual blocks. Thus, the layers are now used in the following order:

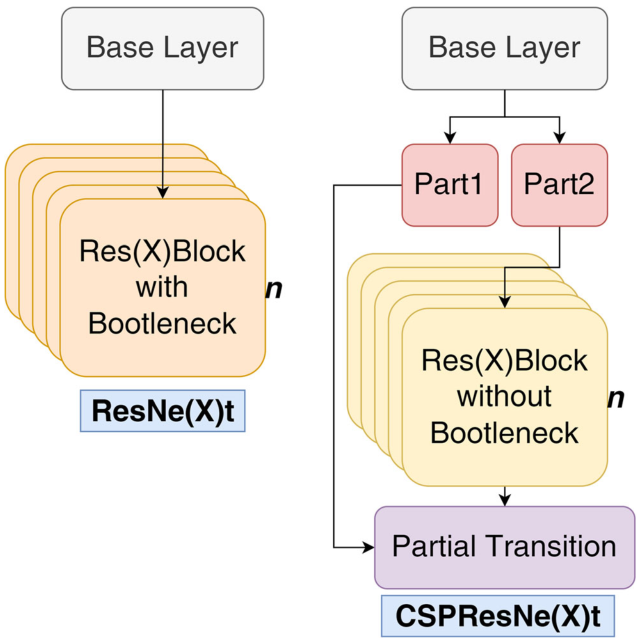

Fig. 2 Applying CSPNet to ResNe(X)t

1xDark, 2xCSPDark, 8xCSPDark, 8CSPDark, 4xCSPDark. Thus, a better balance of speed/accuracy is achieved in the spine section. In the neck part, the PAN architecture in YOLOv4 is placed in the CSP, resulting in significant savings in computational cost. In this way, it effectively reduces 40% of the computational cost. The SPP module, which is one of the other important parts in Scaled-YOLOv4, was used together with the CSPNet structure and placed in the middle of the first calculation list group of CSPPAN. The Scaled-YOLOv4 object detection algorithm, which is the combination of all these units, is one of the most ideal algorithms for real-time polyp detection. This algorithm can be flexibly integrated into endoscopy devices with the model deployment technique.

3.2 Artificial bee colony (ABC) algorithm

ABC algorithm is one of the population-based heuristic algorithms presented by Karaboga in 2005 [39, 50]. The ABC algorithm is modeled by inspiring the foraging behavior of real honeybees in nature. The main goal in the algorithm is that each food source coincides with a possible solution to the problem, so that the best food source gives the best result. In the ABC algorithm, foraging bees are divided according to three tasks in divisions of work: employed bees, onlooker bees, and scout bees. While the employed bees search around the sources they keep in their memory, onlooker bees search potentially better food source areas according to the information they receive from the employed bees. On the other hand, scout bees are responsible for making a global search by making random searches in new food areas. The basic steps of the ABC algorithm are given in Algorithm 1.

Algorithm 1 The Main Steps of the ABC Algorithm

```
1  Initialization of food locations
2  Evaluation phase
3  REPEAT
4  |  Employed bee phase
5  |  Onlooker bee phase
6  |  Scout bee phase
7  UNTIL (A termination criteria)
```

In the ABC algorithm, the control parameter values are determined in the first step, and then the initial locations of the food sources are defined randomly. The initial random population is generated with respect to the lower and upper bounds of the variables and is determined as in Eq. 1.

$$\mathrm{x_{ij}} = \mathrm{x_j^{min}} + \mathrm{rand}\ (0,\ 1)\left(\mathrm{x_j^{min}} - \mathrm{x_j^{min}}\right) \tag{1}$$

where x_j^{max} is the upper bound, x_j^{min} is the lower bound, x_{ij} is the solution vector $(j = 1,2,...,D)$ and $i = 1,2,...,SN$. Here, D is the number of the parameters and SN is the number of solutions. The employed bee, onlooker and scout bee phases are repeated until a stopping criterion is met, keeping the best solution in memory. The employed bee phase produces new solutions by investigating the neighborhood of the existing solutions and uses the formula in Eq. 2.

$$\mathrm{x'_{ij}} = \mathrm{x_{ij}} + \phi_{\mathrm{ij}}\left(\mathrm{x_{ij}} - \mathrm{x_{kj}}\right) \tag{2}$$

where x' is the new candidate solution vector, ϕ_{ij} is a random value between $[-1, 1]$ and x_k corresponds to a randomly chosen neighbor solution vector. According to the Equation 2, after a new solution is produced, the better solution is chosen among the new and existing solutions through the greedy selection mechanism. If the new solution has a better fitness value than the current solution, the new solution replaces the existing solution, otherwise the current solution is kept and the corresponding counter is incremented by one. The purpose of this counter is to decide whether the food source is exhausted or not. In the onlooker bee stage, a new solution is produced by using Equation 1 as in the employed bee stage. In order to determine the food source of an onlooker bee, a probabilistic selection method is used. This ensures that better solutions with higher probability are selected. and allows local search around these solutions, while reducing the chance to search around low-quality solutions. Probability calculations of individuals are made by Eq. 3.

$$P_i = \frac{Fitness_i}{\sum_{i=1}^{NP} Fitness_i} \tag{3}$$

where $p_{i,}$ is the probability of choosing i solution and its fitness corresponds to the quality of the solution. As in the employed bee phase, the greedy selection mechanism is applied here, and the relevant counter (trial) values of the solutions that do not improve are increased by one. As a result of the bees collecting nectar, food sources decrease over time and eventually run out. This corresponds to the fact that the solution can no longer be improved. The determination of whether the resource is exhausted is determined by the control parameter called "limit". When the trial counter of a solution, x_i, exceeds the "limit", the relevant resource is abandoned. After that, the bee of that source conducts random search as a scout bee. ABC algorithm, which is one of the algorithms based on swarm intelligence, has few control parameters in its nature, allowing ABC algorithm to be simple and very flexible to use compared to other algorithms. The number of food sources, the maximum number of cycles and the limit value are the basic control parameters of ABC. The basic flowchart of the ABC algorithm is shown in Fig. 3.

3.3 Proposed method

YOLO algorithms are a set of end-to-end deep learning models designed for real-time and fast object detection. As with YOLO and other deep learning algorithms, there are model-specific hyper-parameters that must be set before starting the model training phase. Training the model with these parameters is very effective in the behavior and performance of the network. Object detection algorithms usually use initial hyper-parameters for MSCOCO [51], but for different tasks such as polyp detection or lung nodule detection, these hyper-parameters may not be optimal. Often users initialize these hyper-parameters from their experience, but still may not achieve the best result. In brute-force hyper-parameter optimization, when the search space is high-dimensional, it takes a lot of effort and more time to reach the best hyper-parameters. For such reasons, heuristic hyperparameter optimization is inevitable to minimize computational cost and user involvement. Here, we present the implementation of the ABC algorithm for optimizing of hyper-parameters in YOLO-based algorithms, which is illustrated in Fig. 4.

As seen in Fig. 4, the proposed method consists of the dataset unit and the ABC-based YOLO algorithm unit. The dataset unit includes several operations. Colonoscopy images are collected from hospital settings with the help of gastroenterologists.

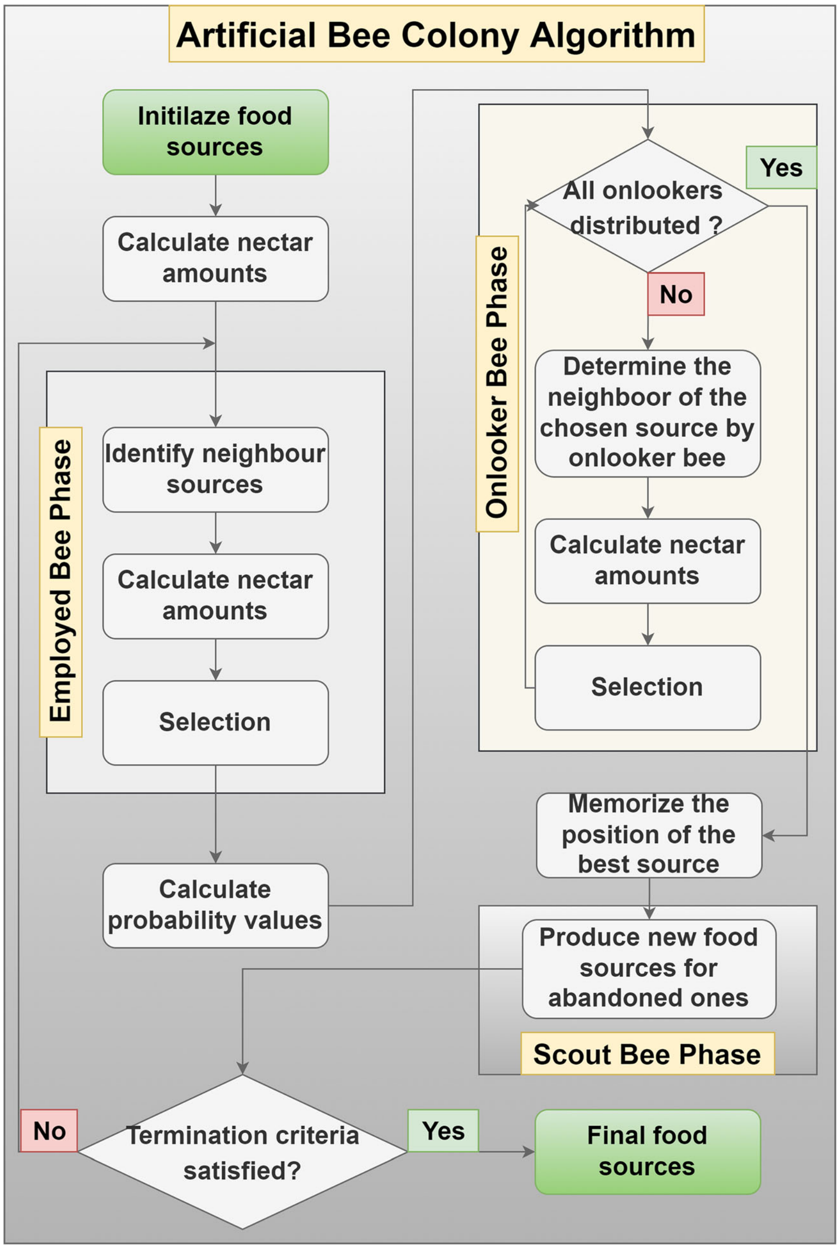
Artificial Bee Colony Algorithm
Initilaze food sources
Calculate nectar amounts
Employed Bee Phase
Identify neighbour sources
Calculate nectar amounts
Selection
Calculate probability values
Onlooker Bee Phase
All onlookers distributed ?
Yes
No
Determine the neighboor of the chosen source by onlooker bee
Calculate nectar amounts
Selection
Memorize the position of the best source
Produce new food sources for abandoned ones
Scout Bee Phase
No
Termination criteria satisfied?
Yes
Final food sources

Fig. 3 Basic flowchart of the ABC algorithm

Gastroenterologists mark the frames of the polyp areas with the help of a computer program. Thus, frames with polyps belonging to each patient are obtained. In the next process, images are made suitable for deep learning algorithms by applying various image processing methods. For example, images with a desired number of polyps are extracted from the frames of a patient and the ground truth or coordinates of these polyps are extracted. Then the labels are prepared as txt, xml, csv or ground truth. In the image preprocessing part, many similar processes such as removal of black borders, specular highlights, interlacing effects, ghost colors and lighting normalization are applied for each polyp image. The obtained images are divided into 3 different folders as test, validation and test within certain rules, and thus the dataset is created.

An object detection algorithm and an optimization algorithm are included in the ABC-based YOLO unit structure. Since the proposed model is flexible, any YOLO-based algorithms can be used, from YOLOv3 to the latest version YOLOv7. In this study, we chose ABC for the optimization algorithm and Scaled-YOLOv4 for the YOLO based algorithm. Since the Scaled-YOLOv4 algorithm contains many models, a wider experimental study can be performed compared to other versions. The basic operation here is that the ABC algorithm tunes the YOLO algorithm to match the hyper-parameter in the YOLO algorithm for the best food source inherent in it. The aim is to present the most ideal hyper-parameters for each dataset. This dataset can be a medical dataset or an object detection dataset, so it is not just for polyp detection. It is the optimization of only hyper-parameters without any change in the structure and training process of YOLO-based algorithms. Therefore, the proposed method performs the training, validation and testing processes in the same way as in the MS-COCO dataset. Algorithm 2 shows more details how the ABC algorithm is employed for hyper-parameter optimization in YOLO algorithms.

Algorithm 2 Proposed Hyper-parameter Optimization

```
1   Load dataset
2   Split dataset as train, validation and test
3   Modify fitness function as mAP
4   Load pre-trained COCO weights
5   Initialize default hyper-parameters of YOLO
6   Train YOLO-based algorithm (small model for
    fast training)
7   Save best model
8   Initialize the parameters of the ABC algorithm
9   Load best model
10  REPEAT
11      Tune best model for a few epochs with ABC
        algorithm
12  UNTIL (A termination criteria)
13  Get optimized hyper-parameter values
14  Re-train or fine-tune YOLO with new hyper-
    parameters
15  Evaluate the performance of models
```

As shown in Algorithm 2, the first steps of the proposed method are to load the dataset into the system. The loaded dataset is then separated as train, validation and test data. Here, only train and validation data are used in training processes. After the training is completed, the generalization ability of the model is measured on the test data. In other words, the training and testing strategies are the same as in the MS-COCO dataset. The section up to this point includes the parts related to the dataset. The next sections continue with the default Sclaed-YOLOv4 parameter values. The fitness function is used as mAP, as in YOLO, because the mAP metric is the best evaluation metric for object detection or polyp detection. The training process is started for the SUN and PICCOLO datasets with the pre-trained weights of the MS-COCO dataset and the YOLOv4s model, which is the smallest model of the default Scaled-YOLOv4 algorithm, and the parameters of this model. We chose the YOLOv4s model here because hyperparameter optimization is a time-consuming process and training small-scale models saves time compared to other large models.

The test data is never used in the training process, only the train and validation data are used. After the training process is completed for the selected number of epochs, the weight with the highest mAP value is recorded and the other weights are not recorded. In the training process, large epoch values are selected and in case there is no increase in the mAP metric after 50 epochs, unnecessary costs are avoided with early stopping. Thus, a more effective training process can be completed. After the training process is completed, the process in which the ABC algorithm is included begins. The parameters of the ABC algorithm are initialized (such as population size, limit value, trial value and total number of evaluations). For the ABC algorithm to optimize the hyper parameters of the YOLOv4 algorithm, lower and upper values are defined for these parameters. Then the best weight of the YOLOv4s model is loaded. The ABC algorithm tries to fine-tune the best model of the YOLOv4 algorithm with the specified number of epochs. This process is done according to the working principle of the ABC algorithm. That is, the ABC algorithm aims to find the best food source. The ABC algorithm sends YOLO the best food source as a parameter, which corresponds to the best solution. This process continues within a certain cycle. Thus, the best solution, that is, the best hyperparameters, is found for each polyp dataset. The fitness function employed by ABC algorithm is defined depending on mAP value generated by the object detection algorithm and given by Eq. 4.

$$fitness_i = mAP_i \tag{4}$$

where i is the number of the solution vector.

Each solution vector consists of the possible hyper-parameter values which is listed in the Table 1 and is represented with a string given in the Eq. 5.

$$A\ solution\ = [x_1, x_2, x_3, \ldots x_i, \ldots x_N] \tag{5}$$

where x_i is a hyper-parameter (to be optimized in the YOLO-based algorithms) and N is the total number of hyper-parameters.

Hyper-parameter values in a possible solution are trained (fine-tuned) for only a few epochs with the best model obtained with YOLOv4s. This cycle continues until a termination criterion is met. At the end of the cycle, the best food source, that is, the best hyper-parameters, is produced based on the ABC algorithm. Thus, the success of the best hyper-parameters is evaluated. The best hyperparameters obtained are stored in a file. Then the training process can be done in two ways; either train the Scaled-YOLOv4 algorithm with new hyper-parameters from scratch or fine-tune the Scaled-YOLOv4 algorithm on the saved model. In both processes, polyp detection of the algorithm is increased.

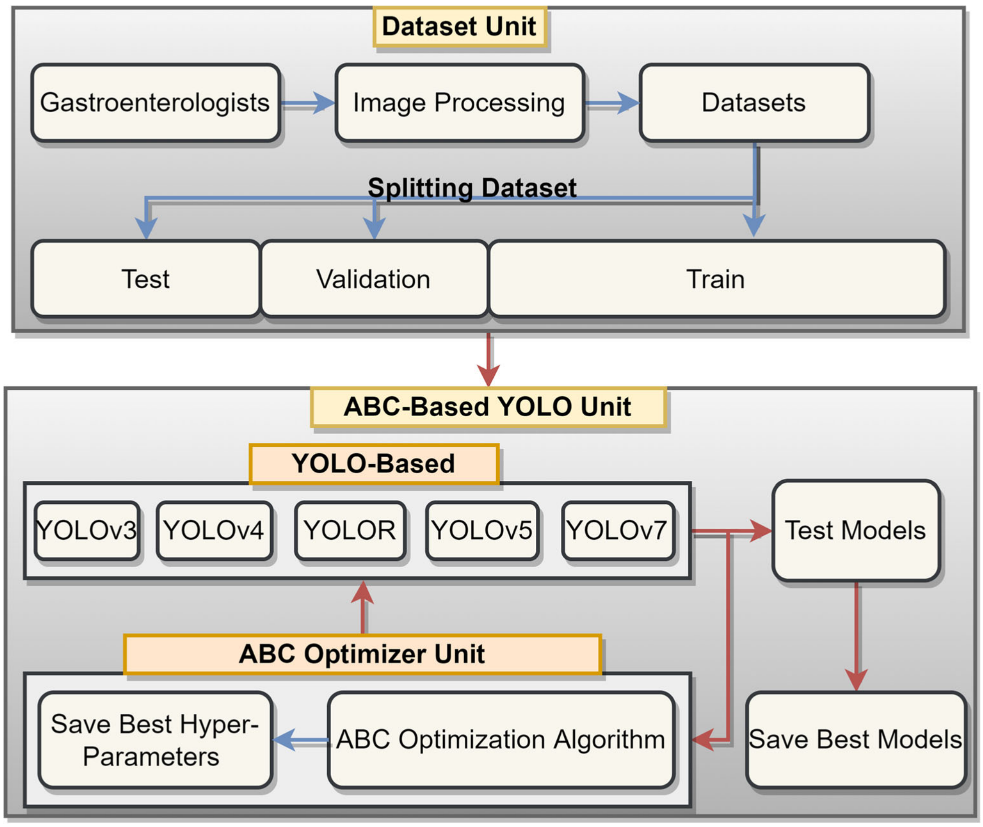

Fig. 4 Structure of the proposed ABC-Based YOLO algorithm

In the last part, the performance of the model obtained with the original Scaled-YOLOv4 and the performance of the YOLOv4 model obtained with ABC are compared on test data. Previously, operations were performed according to the validation data, but only the performance on the test dataset is considered here. In fact, the YOLO algorithm trained with ABC gives higher improvement on both validation data and test data. Here the same operations can be performed using any YOLO-based algorithm such as YOLOv7, YOLOv5, YOLOR and YOLOv3.

4 Experiments

4.1 Experimental setup

In this study, the following computer and frameworks were used to train and test deep learning models. The computer which has the Ubuntu (version 20.04) operating system; consists of Intel Core i9 9900 × (10 cores 3.50 GHz, 19.25 MB Intel® Smart Cache) processor, 64 GB DDR4 RAM and single RTX 8000 (48 GB GDDR6 with 4,608 CUDA cores and

Table 1 Hyper-parameter values of the Scaled-YOLOv4 object detection algorithm

Hyper-parameters name (basic hyper-parameters)	Lower limit	Upper limit	Hyper-parameters name (image-related hiper-parameters)	Lower limit	Upper limit
Initial learning rate	1e-5	1e-1	Image HSV-Hue (fraction)	0.0	0.1
Momentum	0.6	0.98	Image HSV-Saturation (fraction)	0.0	0.9
Optimizer weight decay	0.0	0.001	Image HSV-Value (fraction)	0.0	0.9
GIoU loss gain	0.02	0.2	Image rotation (+/- deg)	0.0	45.0
Classification loss gain	0.2	4.0	Image translation (+/- fraction)	0.0	0.9
Classification BCELoss	0.5	2.0	Image scale (+/- gain)	0.0	0.9
Objectness loss gain	0.2	4.0	Image shear (+/- deg)	0.0	10.0
Objectness BCELoss	0.5	2.0	Image perspective (+/- fraction)	0.0	0.001
IoU training threshold	0.1	0.7	Image flip up-down (probability)	0.0	1.0
Anchor-multiple threshold	2.0	8.0	Image flip left-right (probability)	0.0	1.0
Focal loss gamma	0.0	2.0	Image mix-up (probability)	0.0	1.0

576 NVIDIA Tensor Cores) graphics card hardware. In addition, experiments were carried out using Python 3.8, NVIDIA CUDA Toolkit 11.1, NVIDIA GPU-Accelerated Library (cuDNN) 8.1 and the latest stable version of PyTorch and DarkNet frameworks.

4.2 Datasets

Our study was performed on the SUN and PICCOLO polyp dataset, which are one of the largest publicly available novel datasets for more precise execution of polyp detection performance. The SUN Colonoscopy Video Database is a colonoscopy-based dataset containing high resolution images labeled by many endoscopists. Consisting of 100 unique polyps, the SUN polyp dataset contains 49,136 polyp images. In addition to these polyp images, it includes 109,554 polyp-free background colonoscopy images. Some images of the SUN polyp dataset are shown in Fig. 5.

The SUN polyp dataset is not divided into different folders such as train and test set like other datasets. There are 100 folders in total, each corresponding to a unique polyp. In order to conduct a more independent study, the first 80 (files between 1 and 80) folders, that is, 40,707 polyp images in total, were reserved for the training process (train and validation), while the remaining 20 (files between 81 and 100) folders, 8429 polyp images, were used as the test set. We followed the same dataset rules as Pacal et al. [33], who used the SUN dataset for the first time in automatic polyp detection.

The PICCOLO Widefield dataset consists of a total of 3433 clinical colonoscopy images from 40 patients containing 76 different lesions. The 62 lesions contained white light (WL) and narrowband imaging (NBI), a total of 2131 images, while the remaining 14 lesions contained a total of 1302 NBI images. Polyp segmentation and detection can be performed as there are images of ground truth label against each image in the PICCOLO dataset. Some polyp images of the training set of the PICCOLO dataset are shown in Fig. 6. In the PICCOLO dataset, there are three main folders: train, validation and test. The train folder consists of 2203 images, the validation folder consists of 897 images, and the test folder consists of 333 images. The experiments in this study were carried out considering the rules set by the PICCOLO dataset.

4.3 Evaluation metrics

There are some key metrics to measure the performance of deep learning algorithms in object detection area. Rather than these metrics, we focus on the common assessment metrics used in polyp detection for a fairer comparison. These metrics can be listed as precision, recall, F1-score and mAP, respectively. The object detection algorithm aims to predict rectangular bounding boxes containing polyps defined by four coordinates (x, y, w, h). Thus, all metrics are calculated based on the predicted bounding boxes and the actual bounding boxes (ground truth). The main links in this context are as follows. If the predicted bounding box falls on the ground truth of the polyp it is True Positive (TP), otherwise it is False positive (FP) wedged outside the ground truth. If a predicted bounding box is not produced and there is actually at least one polyp frame, this indicates that the polyp was missed, then False Negative (FN) occurs. Finally, True Negative (TN) indicates no polyps are detected in polyps-free images. Precision gives the positive predictive value, while recall, also known as sensitivity, provides details of the restriction of FNs. The F1-score indicates the harmonic mean of precision and recall metrics. The mAP refers to the area under the precision-

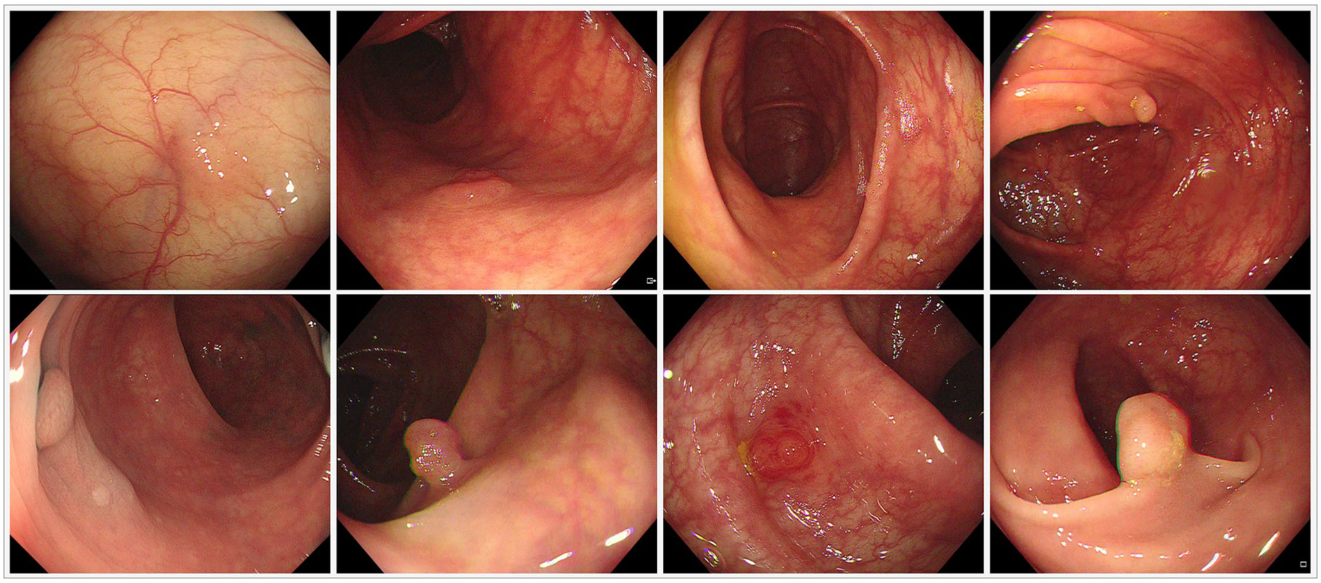
Fig. 5 Some sample polyp images from the SUN polyp dataset

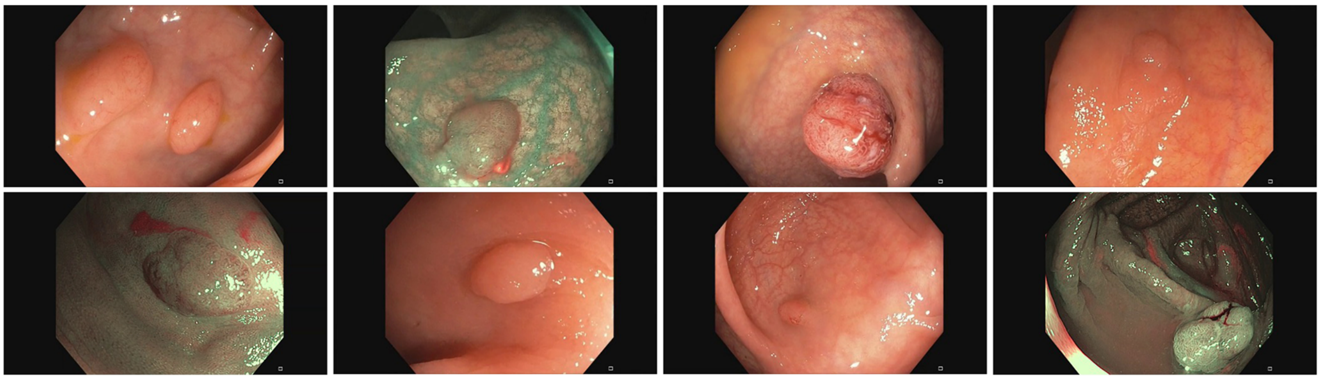

Fig. 6 Some sample polyp images from the PICCOLO polyp dataset

recall curve. FPS (Frame Per Second) describes how fast the object detection model processes images and is the metric used for real-time applications. These metrics are expressed mathematically by Eqs. 6–9 as follows.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = Sensitivity = \frac{TP}{TP + FN} \tag{7}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{8}$$

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{9}$$

5 Results and discussion

This section includes the experimental results of the performances of the pure Scaled-YOLOv4 and the ABC-based Scaled-YOLOv4 algorithm, along with a comparative discussion of these results. Experimental studies primarily show the performance of YOLOv4s, YOLOv4m, YOLOv4-CSP, YOLOv4-p5, YOLOv4-p6 models of the pure Scaled-YOLOv4 algorithm on SUN and PICCOLO datasets. That is, the model and hyper-parameters were trained and evaluated as in the repository [52, 53]. Scratch (MS-COCO) hyper-parameters were used to train these models and fine-tune hyper-parameters were used for fine-tuning (MS-COCO).

Here, YOLOv4s model was tried to be optimized with initial hyper-parameters with ABC optimization algorithm because hyper-parameter optimization took quite a long time in other large models. Since the hyper-parameters obtained at the end of the optimization with YOLOv4s will be the same for other larger-scale models, using this model saves time and cost. There are two main groups of hyper-parameters in YOLO algorithms. The first group includes basic hyper-parameters such as learning rate, momentum, weight reduction, while hyper-parameters such as scale, shear, and translation are hyper-parameters related to data augmentation techniques. Table 1 shows the hyper-parameters used in the Scaled-YOLOv4 algorithm.

The hyper-parameter numbers to be optimized in YOLO algorithms are approximately close to each other. For example, while the number of hyper-parameters to be optimized in the YOLOv5 algorithm is 29, it is 28 in the YOLOv4 algorithm, 28 in the YOLOR algorithm, and 22 in the Scaled-YOLOv4 algorithm. All hyper-parameters of any YOLO algorithm can be optimized with the ABC optimization algorithm. In object detection algorithms and YOLO algorithms, two different hyper-parameter files are generally used for the MSCOCO dataset. For example, in YOLO algorithms, scratch hyper-parameter file is used in MSCOCO training from scratch, while optimized fine-tune file is used for MSCOCO fine-tuning.

In this study, we first need to train the SUN and PICCOLO dataset with the scratch hyper-parameter file. Next, the same steps need to be performed with the fine-tune hyper-parameter file. These hyper-parameters are already defined for the MS-COCO dataset in the Scaled-YOLOv4 repository [52]. Finally, we give the hyper-parameters in Table 1 as input to the ABC algorithm. ABC algorithm considers the lower and upper limit values as a food source and tries to find the best food. The best food source inherent in the ABC algorithm corresponds to the most performance-enhancing hyper-parameters for YOLO algorithms. Thus, the most ideal hyper-parameter values for each dataset are obtained with the ABC algorithm. The hyper-parameter values of these files are shown in Table 2. These hyper-parameters are important for training and evaluating of models.

As seen in Table 2, ABC optimization algorithm increases the accuracy of polyp detection by trying to optimize the hyper parameters between the lower and upper values. In our study, some parameter values used by the ABC algorithm to better optimize the YOLO algorithm were chosen as follows.

Table 2 Hyper-parameter values of the Scaled-YOLOv4 employed in this paper (Scratch, Fine-tune and Fine-tuned with ABC algorithm for SUN and PICCOLO polyp datasets)

Hyper-parameters	Scratch (MS-COCO)	Fine-tune (MS-COCO)	ABC (SUN)	ABC (PICCOLO)	Hyper-parameters	Scratch (MS-COCO)	Fine-tune (MS-COCO)	ABC (SUN)	ABC (PICCOLO)
Initial learning rate	0.01	0.01	0.095	0.09	Image HSV-Hue	0.015	0.015	0.0165	0.0137
Momentum	0.937	0.937	0.930	0.923	Image HSV-Saturation	0.7	0.7	0.721	0.62
Optimizer weight decay	0.0005	0.0005	0.00054	0.00055	Image HSV-Value	0.4	0.4	0.446	0.497
GIoU loss gain	0.05	0.05	0.052	0.0479	Image rotation	0.0	0.0	0.15	0.125
Classification loss gain	0.5	0.5	0.55	0.514	Image translation	0.5	0.5	0.35	0.197
Classification BCELoss	1.0	1.0	1.12	0.9	Image scale	0.5	0.8	0.62	0.48
Objectness loss gain	1.0	1.0	0.9172	1.1	Image shear	0.0	0.0	0.15	0.2
Objectness BCELoss	1.0	1.0	0.9	1.07	Image perspective	0.0	0.0	0.0	0.0
IoU training threshold	0.2	0.20	0.2	0.2	Image flip up-down	0.0	0.0	0.35	0.42
Anchor-multiple threshold	4.0	4.0	3.5	2.87	Image flip left-right	0.5	0.5	0.4	0.48
Focal loss gamma	0.0	0.0	0.0	0.0	Image mix-up	0.0	0.2	0.31	0.11

The population size was taken as 50, the limit value was 60, and the total number of evaluations was 5000. As summarized in Algorithm 2, firstly, the smallest model of the Scaled-YOLOv4 algorithm, YOLOv4s, is trained with the SUN dataset with enough epoch values and default hyper-parameters. The best model obtained here is fine-tuned with a small epoch value depending on the upper and lower hyper-parameter values with the ABC algorithm.

As a result, the ABC algorithm improved its performance by fine-tuning the weight of YOLOv4s with the best model and trained hyper-parameters corresponding to the best food source it obtained. In this study, we chose the mAP metric as the fitness function. Therefore, the ABC algorithm tries to optimize the mAP metric. Any metric can be selected as a fitness function, the reason for choosing mAP is that it reveals the performance more clearly in object detection algorithms and polyp detection than other metrics. As can be seen in Table 3, metrics such as precision, recall and F1, which are commonly used in polyp detection, have been added. The results of the Scaled-YOLOv4 algorithm trained with scratch, fine-tune and ABC hyper-parameters are given in Table 3.

Table 3 gives the total number of layers, parameters and FPS value of each model of the Scaled-YOLOv4 algorithm. The YOLOv4s model, which is one of the smallest members of the Scaled-YOLOv4 model, is the ideal model for small-scale datasets and less hardware needs. Considering Table 3, as the number of layers in the model increases, the number of parameters increases proportionally, but the FPS ratio increases in the opposite direction, that is, a decrease. On the other hand, as the number of layers of the model increases, the performance of the model increases at a correct rate, if there is sufficient hardware and in large-scale datasets. In large-scale datasets such as SUN and PICCOLO, as the model grows, the detection performance of the model increases proportionally, while the real-time speed of the model decreases with the growth of the model. As a result, the most ideal model should be chosen considering the size of the available dataset and the hardware.

The training time of each model varies according to the dataset and the size of the model. For the SUN dataset, the duration of each epoch corresponds to 2 min, while for the PIICOLO dataset, this time is even shorter. The number of Bath-size, Image-size and GPU cores and other similar elements are important in training time. With the growth of the model, model training becomes very slow due to the parameters of the model and the number of layers. If YOLOv4-P7 is selected instead of the YOLOv4s model that completes training in a few hours, this time can be treated as days. Since it is of vital importance in medical image analysis, especially in cancer detection or classification, the accuracy of the model should be increased by using the most powerful models possible.

Table 3 Some characteristics of the models of the Scaled-YOLOv4 algorithm (FPS value is calculated based on NVIDIA RTX 8000 GPU card.)

Model	Number of layers	Params (Million)	FPS	Model	Number of layers	Params (Million)	FPS
YOLOv4s	226	9.4	100	YOLOv4-P5	476	70.9	53
YOLOv4m	262	24.7	87	YOLOv4-P6	599	127.6	42
YOLOv4-CSP	334	52.9	71	YOLOv4-P7	722	287.6	35

Tables 4 and 5 show the performances of the Scaled-YOLOv4 algorithm according to files containing 3 different hyper-parameters for SUN and PICCOLO polyp datasets. Here, the img-size (pixels) shows the dimensions of the images used in training and testing. In the experiments, we tried to choose the image size with 416 × 416 resolution, but we selected different resolutions for the large models Scaled-YOLOv4-P6 and Scaled-YOLOv4-P7 and models. The main reason is that we updated the value closest to 416 to 448, since the maximum stride of the Scaled-YOLOv4-P6 model is a multiple of 64 and we updated the Scaled-YOLOv4-P7 model to 512, since the maximum stride of the Scaled-YOLOv4-P7 model is a multiple of 128. P represents precision, R represents recall, F1 represents F1-score, and mAP represents mean average precision. Many parameters such as IOU: 0.65, batch-size 1, confidence threshold: 0.001 were selected by default in the Scaled-YOLOv4 repository [53]. The main purpose here is to examine the effect of hyper-parameters on performance and to demonstrate the success of the ABC algorithm in optimizing hyper-parameters. Therefore, each model was trained 10 times for a more stable experimental study. Then, the average of the metrics belonging to each model was taken and the evaluation was made.

Considering Tables 4 and 5, the most striking point here is that the Scaled-YOLOv4 models trained with the hyper-parameters of the ABC optimization algorithm shows a significant increase in performance compared to the models trained with other hyper-parameters. The following increases were observed in Scaled-YOLOv4 models fine-tuned with ABC hyper-parameters. For the SUN dataset, there was an increase of more than 3% in the mAP value, while for the PICCOLO dataset, there was an increase in the mAP value of over 2%. In the same way, an increase of more than 2% was achieved in the F1 value for both datasets. That is, the ABC optimization algorithm can successfully optimize the hyper-parameters for any dataset, increasing the performance. Another important point is that with the growth of the model (YOLOv4s->YOLOv4-P7), the performance has been increased significantly. The main reason for this is that if the deep learning architectures are trained with a sufficient dataset, the performance of the models increases with the deepening of the architecture, such as the number of layers of the model and the number of filters.

The YOLOv4-P7 model showed the best detection accuracy in polyp detection and the best performance of this model is achieved with the hyper-parameters produced by the ABC algorithm. YOLOv4s, which is the smallest model of the Scaled-YOLOv4 algorithm, has the lowest detection accuracy compared to other models. Despite all this, YOLOv4s is the best model in terms of real-time performance, that is, according to FPS values, while YOLOv4-P7 is the model with the lowest FPS value. When the F1 and mAP metrics of the models are examined, it is seen that the models trained with the hyper-parameters of ABC seem to increase the performance in these metrics continuously, but also have a more stable and regular improvement. Models with both detection accuracy and real-time performance can move to the next stage, the clinical application of the polyp detection system.

Table 4 Results on SUN polyp dataset (Default parameters of Scaled-YOLOv4, IOU: 0.65, batch-size 1, confidence threshold: 0.001, device: 1xRTX 8000)

Scaled-YOLOv4		Results using scratch				Results using fine-tune				Results using ABC			
Model	Img-size	P	R	F1	mAP	P	R	F1	mAP	P	R	F1	mAP
YOLOv4s	416	76.32	76.11	76.21	81.13	79.14	80.22	79.86	83.08	76.14	81.4	78.68	**84.14**
YOLOv4m	416	83.28	74.34	78.56	82.12	74.86	81.65	78.11	84.27	83.43	83.68	83.55	**88.82**
YOLOv4-CSP	416	80.90	78.61	79.74	84.92	81.30	85.11	83.16	87.43	86.23	84.06	85.13	**89.39**
YOLOv4-P5	416	79.18	80.96	80.06	85.24	82.46	84.14	83.29	87.78	83.76	87.51	85.28	**90.79**
YOLOv4-P6	448	78.68	83.96	81.24	87.63	82.77	85.94	84.32	88.18	83.36	87.71	85.47	**90.98**
YOLOv4-P7	512	80.36	84.92	82.57	87.89	83.49	85.87	84.66	88.75	84.42	89.33	86.80	**92.93**

Bold font indicates the best performance

Table 5 Results on PICCOLO polyp dataset (Default parameters of Scaled-YOLOv4, IOU: 0.65, batch-size 1, confidence threshold: 0.001, device: 1xRTX 8000)

Scaled-YOLOv4		Results using scratch				Results using fine-tune				Results using ABC			
Model	Img-size	P	R	F1	mAP	P	R	F1	mAP	P	R	F1	mAP
YOLOv4s	416	60.75	76.67	65.33	71.41	64.01	72.47	67.98	72.81	63.74	76.28	69.45	**75.14**
YOLOv4m	416	61.89	72.96	66.97	73.98	70.76	72.47	71.60	74.54	68.79	75.26	71.88	**75.68**
YOLOv4-CSP	416	84.42	67.35	74.92	74.33	77.74	72.70	75.14	74.96	81.35	78.57	79.93	**78.11**
YOLOv4-P5	416	76.75	72.96	74.81	74.65	74.52	79.08	76.73	76.69	81.60	78.79	80.17	**79.67**
YOLOv4-P6	448	80.24	74.59	77.31	78.81	75.15	78.06	76.58	78.98	82.14	80.65	81.39	**82.27**
YOLOv4-P7	512	78.85	77.55	78.20	79.56	82.39	77.55	79.90	79.85	82.69	81.85	82.26	**83.18**

Bold font indicates the best performance

For a model to be counted in real time, its FPS value must be at least 30. If the model is to be integrated into an endoscopy device, the FPS value must be even higher than 30, otherwise there will be delays in real applications, which limits the applicability of the model. For non-real-time situations, the best model is the YOLOv4-P7, largest model. On the other hand, YOLOv4s, YOLOv4m, YOLOv4-CSP appeared to be the sufficient for real-time application. If Scaled-YOLOv4 models are deployed, YOLOv4-P7 may become more usable for real-time applications. Furthermore, cloud GPUs or high-end GPUS should be used for larger models such as YOLOv4-P7 and YOLOv4-P6.

Figure 7 shows an example of the detection performance of the Scaled-YOLOv4 algorithm (YOLOv4s vs. YOLOv4-P7) on test images of the SUN dataset. As seen in Tables 4 and 5, the detection performance of the YOLOv4s model is lower than that of the YOLOv4-P7 model. The ABC algorithm, on the other hand, is seen to significantly increase the detection performance of the models. While the YOLOv4s model cannot detect 3 randomly selected images from the test data in the SUN dataset, in short, 3 FN values occur, while in the YOLOv4-P7 model there is no FN value, that is, detects all randomly selected images. As a result, it is obvious that there is a significant increase in the accuracy of polyp detection as the model grows in the Scaled-YOLOv4 object detection algorithm. Figure 8 shows an example of the detection performance of the Scaled-YOLOv4 algorithm (YOLOv4s and YOLOv4-P7) on test images of the PICCOLO dataset. As seen in Tables 4 and 5, the detection performance of the YOLOv4s model is lower than that of the YOLOv4-P7 model. Considering Table 4, it is seen that the ABC algorithm significantly increases the detection performance of the models. As a result, it is seen that there is a significant increase in polyp detection accuracy as the model grows in the Scaled-YOLOv4

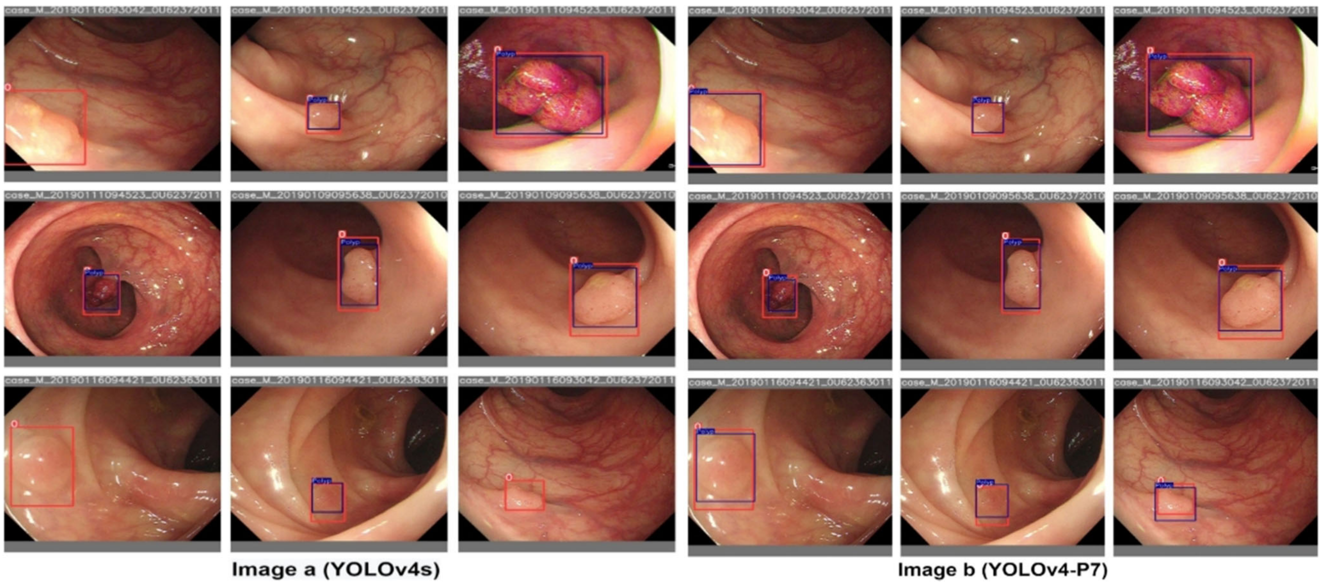

Fig. 7 Detection performance of the Scaled-YOLOv4 on SUN dataset (The red bounding boxes show ground truth. The image a (left) shows the results of the YOLOv4s model, on the other hand, the image b (right) shows the results of the YOLOv4-P7 model)

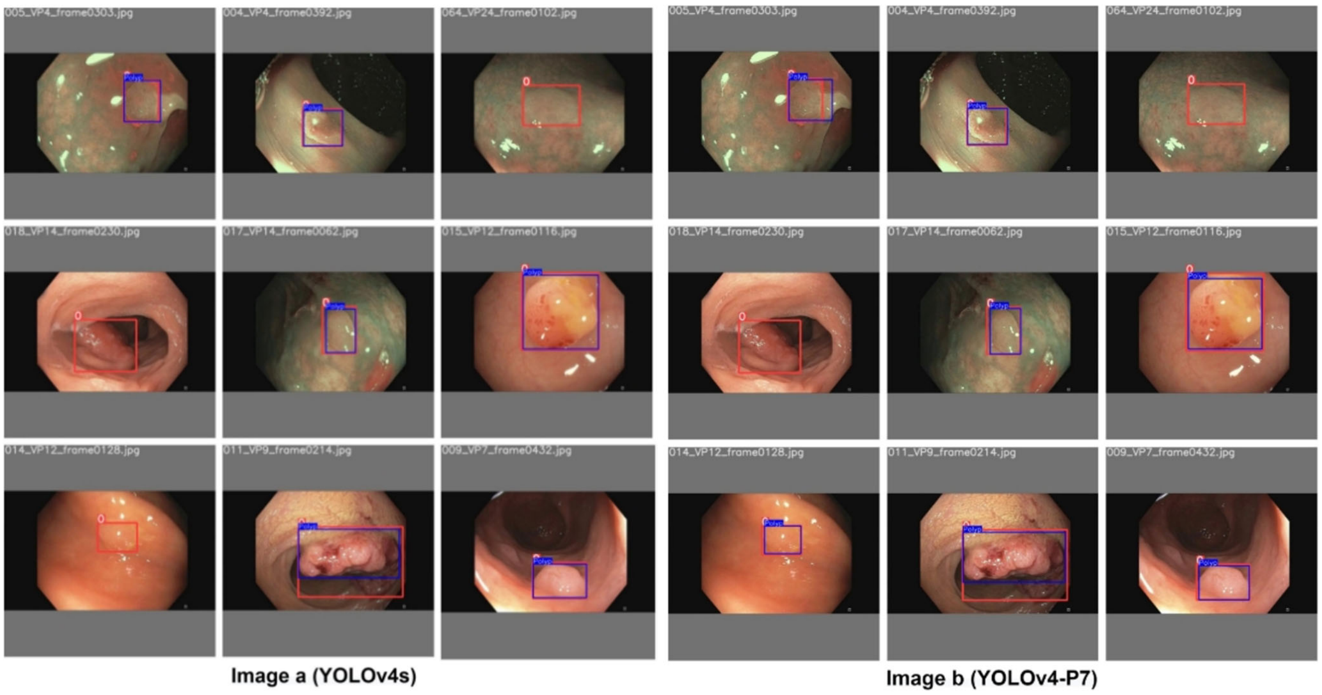

Fig. 8 Detection performance of the Scaled-YOLOv4 on PICCOLO dataset (The red bounding boxes show ground truth. The image **a** (left) shows the results of the YOLOv4s model, on the other hand, the image **b** (right) shows the results of the YOLOv4-P7 model)

object detection algorithm. The YOLOv4s model lags far behind the YOLOv4-P7 model in terms of detection accuracy. In terms of FPS, the smaller model YOLOv4s is more successful. YOLOv4-P6 and YOLOv4-P7 models will be much more successful in FPS if the model is deployed (NVIDIA TensorRT). As mentioned earlier, large models such as YOLOv4-P7 offer higher detection accuracy in large-scale datasets such as SUN and PICOLO.

5.1 The performance of the proposed method

Considering the experimental results, the superiority of the proposed method is as follows. It has been observed that all Scaled-YOLOv4 models (YOLOv4s, YOLOv4m, YOLOv4-CSP, YOLOv4-P5, YOLOv4-P6, YOLOv4-P7) trained with hyper-parameters of the ABC optimization algorithm provide significant improvements compared to models trained with other hyper-parameters such as MS- COCO. In this sense, it is seen that there is a 3% increase in F1 and mAP values, as well as similar notable increases in other metrics. The proposed method is also superior in completing the training successfully and faster. For example, when the YOLOv4s model is trained with an NVIDIA RTX 8000 card for 400 epochs on the SUN dataset, each epoch takes approximately 2 min and a total of more than 800 min (approximately 14 h). If trying to manually optimize the 22 hyper-parameters of the Scaled-YOLOv4 algorithm in a controlled experiment, a total of 14 × 22 (approximately 13 days) hours is required. On the other hand, ABC optimization algorithm can optimize all these hyper-parameters at once and optimally in about two days. This is vital in terms of both time savings and cost savings. Another advantage of the proposed method is that it can be applied to any dataset. For this, it is necessary to fine-tune the best weight of the model with the ABC optimization algorithm to find the most ideal hyper-parameters. The hyperparameters obtained for any given dataset may not be ideal for a different dataset. For example, in the SUN dataset, polyps are generally high resolution and located in the middle of the image. In the PICCOLO dataset, this situation is different. It can be said that each dataset is unique for itself, due to many factors such as the location of the polyp in the dataset, its resolution, the angle of the colonoscopy camera, the light, and the degree of crystallinity of the polyps.

This method offers optimal hyper-parameters of datasets for all domains such as lung nodule, breast mammography or any object detection period. In addition, the proposed method can be easily adapted to any YOLO-based object detection algorithm. It can adapt to the latest YOLOv7 algorithm and quickly adapt to older YOLO versions such as YOLOv3.

5.2 Comparison with state-of-the-art methods

In this section, we compare our proposed method with similar studies. Although there are many studies on automatic polyp detection, almost most of the studies do not use a common experimental setup and evaluation criteria. In other words, the studies do not use a common training and test data in the experiments. Most studies, on their own initiative, divide

Table 6 Comparison of the proposed method with state-of-the-art methods

Author, Year	Method	Training dataset	Test dataset	Precision	Recall	F1	mAP
Rodriguez et al., 2022 [54]	YOLOv3	SUN	SUN	0.83	0.78	0.81	0.81
Pure Scaled-YOLOv4	YOLOv4-CSP (scratch)	SUN	SUN	0.80	0.78	0.79	0.84
Pure Scaled-YOLOv4	YOLOv4-CSP (fine-tune)	SUN	SUN	0.81	**0.85**	0.83	0.87
Proposed	YOLOv4-CSP+ABC	SUN	SUN	**0.86**	0.84	**0.85**	**0.89**
Rodriguez et al., 2022 [54]	YOLOv3	PICCOLO	PICCOLO	0.76	0.60	0.67	0.63
Pure Scaled-YOLOv4	YOLOv4-CSP (scratch)	PICCOLO	PICCOLO	**0.84**	0.67	0.74	0.74
Pure Scaled-YOLOv4	YOLOv4-CSP (fine-tune)	PICCOLO	PICCOLO	0.77	0.72	0.75	0.74
Proposed	YOLOv4-CSP+ABC	PICCOLO	PICCOLO	0.81	**0.78**	**0.79**	**0.78**

Bold font indicates the best performance

datasets into training, validation and testing. In such cases, the objectivity of the studies decreases as well as the possibility of comparison. Since the SUN and PICCOLO datasets are the most up-to-date polyp datasets, few studies use these datasets. Despite all these negativities, we compared our proposed method with the original Scaled-YOLOv4 algorithm and some other studies. Table 6 gives a comparison of the proposed method with other state-of-the-art methods.

As shown in Table 6, since the SUN and PICCOLO datasets are new public datasets, the studies using these datasets are limited to a few. Rodruguez et al. proposed a YOLOv3-based method for real-time polyp detection. This method was trained and evaluated on SUN and PICCOLO datasets. Although the proposed method is successful on these datasets, its performance is slightly less than the Scaled-YOLOv4 algorithm. The performance of polyp detection in the SUN and PICCOLO datasets has increased further with the use of the ABC algorithm in the hyper-parameter optimization of the Scaled-YOLOv4 algorithm. Considering the performance of the models for the SUN dataset, the proposed method outperforms the method proposed by Rodriguez et al. by more than approximately 9% and by more than approximately 5% the YOLOv4-CSP (scratch) model. Likewise, when looking at the detection performance of the models for the PICCOLO dataset, it is more successful by more than 20% than the method proposed by Rodriguez et al. and approximately 5% more than the YOLOv4-CSP (scratch) model. Although all models have a significant performance in real-time detection, the Scaled-YOLOv4 algorithm is more successful in both inference speed and detection accuracy than the YOLOv3 algorithm.

5.3 Limitation of the study

The proposed method offers more successful results than the existing Scaled-YOLOv4 algorithm. It optimizes the hyper-parameters of YOLO algorithms more effectively in terms of time and cost. Despite all these positive aspects, the study was not considered in a wider perspective due to the insufficient polyp dataset available to the public. Some public datasets in the literature were not included in the study because they contain very few images with polyps and a small number of patients. There is a data-hungry need inherent in deep learning algorithms to show high performance.

6 Conclusion

In this study, we propose the ABC algorithm for hyper-parameter optimization of YOLO-based algorithms, one of the popular object detection algorithms, to improve real-time automatic polyp detection performance. By integrating the ABC algorithm into YOLO-based algorithms, we optimized the hyper-parameters, which contributed significantly to the improvement of detection accuracy. In experimental studies, it has been observed that ABC algorithm both gives more successful results and is user-friendly by optimizing the hyper-parameters of YOLO algorithms in a heuristic way at once. We shine a light on new researchers and research by presenting the most comprehensive performance of all models of the Scaled-YOLOv4 algorithm for real-time automatic polyp detection for the novel SUN and PICCOLO polyp datasets. In the future, we plan to do a wider study with larger datasets and new publicly available datasets. Thus, the generalization ability of the proposed method and the performance of hyper-parameter optimization will be more effective. In addition, we plan to optimize other structures existing in YOLO algorithms with ABC optimization algorithm. In this context, we are trying to carry out larger studies and we are making plans for the proposed method to be included in the clinic soon.

Acknowledgements Experimental computations were carried out on the calculation units in Artificial Intelligence and Big Data Application and Research Center, Erciyes University, Türkiye. Ethical permissions were granted by Acıbadem University, Türkiye with the permission number ATADEK-2021/02 and the decision number 2021-02/15.

Data availability The authors declare that the data supporting the findings of this study are included in this manuscript.

Declarations

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

References

1. Sung H, Ferlay J, Siegel RL et al (2021) Global cancer statistics 2020: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries. CA Cancer J Clin 71:209–249. https://doi.org/10.3322/caac.21660
2. Rawla P, Sunkara T, Barsouk A (2019) Epidemiology of colorectal cancer: incidence, mortality, survival, and risk factors. Prz Gastroenterol 14:89–103. https://doi.org/10.5114/pg.2018.81072
3. Corley DA, Jensen CD, Marks AR et al (2014) Adenoma detection rate and risk of colorectal cancer and death A BS TR AC T. N Engl J Med 14:1298–1306. https://doi.org/10.1056/NEJMoa1309086
4. Kaminski MF, Wieszczy P, Rupinski M et al (2017) Increased rate of adenoma detection associates with reduced risk of colorectal cancer and death. Gastroenterology 153:98–105. https://doi.org/10.1053/j.gastro.2017.04.006
5. Khan A, Sohail A, Zahoora U et al (2020) A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev 53:5455–5516. https://doi.org/10.1007/s10462-020-09825-6
6. Işık G, Artuner H (2020) Turkish dialect recognition in terms of prosodic by long short-term memory neural networks. J Fac Eng Archit Gazi Univ 35:213–224. https://doi.org/10.17341/gazimmfd.453677
7. Guo Y, Liu Y, Oerlemans A et al (2016) Deep learning for visual understanding: a review. Neurocomputing 187:27–48. https://doi.org/10.1016/j.neucom.2015.09.116
8. Kiliçarslan S, Celik M (2022) KAF + RSigELU: a nonlinear and kernel-based activation function for deep neural networks. Neural Comput Appl 34:13909–13923. https://doi.org/10.1007/S00521-022-07211-7/FIGURES/4
9. Ozkok FO, Celik M (2022) A hybrid CNN-LSTM model for high resolution melting curve classification. Biomed Signal Process Control 71:103168. https://doi.org/10.1016/J.BSPC.2021.103168
10. Pacal I, Karaboga D, Basturk A et al (2020) A comprehensive review of deep learning in colon cancer. Comput Biol Med 126. https://doi.org/10.1016/J.COMPBIOMED.2020.104003
11. Bora K, Bhuyan MK, Kasugai K et al (2021) Computational learning of features for automated colonic polyp classification. Sci Rep 11:1–16. https://doi.org/10.1038/s41598-021-83788-8
12. Theodosi A, Ouzounis S, Kostopoulos S et al (2021) Design of a hybrid deep learning system for discriminating between low- and high-grade colorectal cancer lesions, using microscopy images of IHC stained for AIB1 expression biopsy material. Mach Vis Appl 32:1–17. https://doi.org/10.1007/s00138-021-01184-8
13. Schiele S, Arndt TT, Martin B et al (2021) Deep learning prediction of metastasis in locally advanced colon cancer using binary histologic tumor images. Cancers (Basel) 13:2074. https://doi.org/10.3390/cancers13092074
14. Ali M, Ali R (2021) Multi-input dual-stream capsule network for improved lung and colon cancer classification. Diagnostics 11:1–18. https://doi.org/10.3390/diagnostics11081485
15. Tamang LD, Kim BW (2021) Deep learning approaches to colorectal cancer diagnosis: a review. Appl Sci 11:10982. https://doi.org/10.3390/app112210982
16. Sánchez-Peralta LF, Pagador JB, Sánchez-Margallo FM (2021) Artificial intelligence for colorectal polyps in colonoscopy. Artif Intell Med:1–15. https://doi.org/10.1007/978-3-030-58080-3_308-1
17. Kilicarslan S, Celik M, Sahin afak (2021) Hybrid models based on genetic algorithm and deep learning algorithms for nutritional Anemia disease classification. Biomed Signal Process Control 63: 1746–8094. https://doi.org/10.1016/j.bspc.2020.102231
18. Zhang R, Zheng Y, Poon CCY et al (2018) Polyp detection during colonoscopy using a regression-based convolutional neural network with a tracker. Pattern Recognit 83:209–219. https://doi.org/10.1016/j.patcog.2018.05.026
19. Wang C-Y, Yeh I-H, Liao H-YM (2021) You only learn one representation: unified network for multiple tasks, pp 1–11
20. Girshick R, Proc (2015) IEEE Int Conf Comput Vis 2015 Inter, pp 1440–1448. https://doi.org/10.1109/ICCV.2015.169
21. Liu W, Anguelov D, Erhan D et al (2016) SSD: single shot multibox detector. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics). 9905 LNCS, pp 21–37. https://doi.org/10.1007/978-3-319-46448-0_2
22. Duan K, Bai S, Xie L et al (2019) CenterNet: keypoint triplets for object detection. Proc IEEE Int Conf Comput Vis 2019-October, pp 6568–6577. https://doi.org/10.1109/ICCV.2019.00667
23. Liew WS, Tang TB, Lin CH, Lu CK (2021) Automatic colonic polyp detection using integration of modified deep residual convolutional neural network and ensemble learning approaches. Comput Methods Programs Biomed 206:106114. https://doi.org/10.1016/J.CMPB.2021.106114
24. Younas F, Usman M, Yan WQ (2022) A deep ensemble learning method for colorectal polyp classification with optimized network parameters. App Intell. https://doi.org/10.1007/s10489-022-03689-9
25. Lee J-N, · J-WC, Cho H-C (2022) Improvement of colon polyp detection performance by modifying the multi-scale network structure and data augmentation. J Electr Eng Technol 2022:1–9. https://doi.org/10.1007/S42835-022-01191-3
26. Nogueira-Rodríguez A, Domínguez-Carbajales R, Campos-Tato F et al (2021) Real-time polyp detection model using convolutional neural networks. Neural Comput Appl 4. https://doi.org/10.1007/s00521-021-06496-4
27. Hoang MC, Nguyen KT, Kim J et al (2021) Automated bowel polyp detection based on actively controlled capsule endoscopy: feasibility studycs 11:1878. https://doi.org/10.3390/diagnostics11101878
28. Wan J, Chen B, Yu Y (2021) Polyp detection from colorectum images by using attentive YOLOv5. Diagnostics 11:2264. https://doi.org/10.3390/diagnostics11122264
29. Pacal I, Karaboga D (2021) A robust real-time deep learning based automatic polyp detection system. Comput Biol Med 134. https://doi.org/10.1016/J.COMPBIOMED.2021.104519
30. Pacal I, Karaman A, Karaboga D et al (2022) An efficient real-time colonic polyp detection with YOLO algorithms trained by using negative samples and large datasets. Comput Biol Med 141: 105031. https://doi.org/10.1016/j.compbiomed.2021.105031
31. Souaidi M, El Ansari M (2022) Multi-scale hybrid network for polyp detection in wireless capsule endoscopy and colonoscopy images. Diagnostics (Basel) 12(8):2030. https://doi.org/10.3390/diagnostics12082030
32. Durak S, Bayram B, Bakırman T et al (2021) Deep neural network approaches for detecting gastric polyps in endoscopic images. Med Biol Eng Comput 59:1563–1574. https://doi.org/10.1007/s11517-021-02398-8
33. Bochkovskiy A, Wang C-Y, Liao H-YM (2020) YOLOv4: optimal speed and accuracy of object detection. https://doi.org/10.48550/arxiv.2004.10934
34. Tan M, Pang R, Le QV (2020) EfficientDet: scalable and efficient object detection. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit, pp 10778–10787. https://doi.org/10.1109/CVPR42600.2020.01079

35. Redmon J, Farhadi A (2018) YOLOv3: an incremental improvement. https://doi.org/10.48550/arxiv.1804.02767
36. Ren S, He K, Girshick R, Sun J (2017) Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39:1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031
37. Qian Z, Jing W, Lv Y, Zhang W (2022) Automatic polyp detection by combining conditional generative adversarial network and modified you-only-look-once. IEEE Sens J 22. https://doi.org/10.1109/JSEN.2022.3170034
38. Akay B, Karaboga D, Akay R (2022) A comprehensive survey on optimizing deep learning models by metaheuristics. Artif Intell Rev 55:829–894. https://doi.org/10.1007/s10462-021-09992-0
39. Karaboga D, Basturk B (2007) Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems, pp 789–798
40. Erkan U, Toktas A, Ustun D (2022) Hyperparameter optimization of deep CNN classifier for plant species identification using artificial bee colony algorithm. J Ambient Intell Human Comput 1:3. https://doi.org/10.1007/s12652-021-03631-w
41. Banharnsakun A (2019) Towards improving the convolutional neural networks for deep learning using the distributed artificial bee colony method. Int J Mach Learn Cybern 10:1301–1311. https://doi.org/10.1007/s13042-018-0811-z
42. Ozcan T, Basturk A (2019) Transfer learning-based convolutional neural networks with heuristic optimization for hand gesture recognition. Neural Comput Appl 31:8955–8970. https://doi.org/10.1007/s00521-019-04427-y
43. Chollet F (2017) Xception: deep learning with depthwise separable convolutions. Proc – 30th IEEE Conf Comput Vis Pattern Recognition, CVPR 2017, pp 1800–1807. https://doi.org/10.1109/CVPR.2017.195
44. Badem H, Basturk A, Caliskan A, Yuksel ME (2017) A new efficient training strategy for deep neural networks by hybridization of artificial bee colony and limited–memory BFGS optimization algorithms. Neurocomputing 266:506–526. https://doi.org/10.1016/j.neucom.2017.05.061
45. Katoch S, Chauhan SS, Kumar V (2021) A review on genetic algorithm: past, present, and future. Multimed Tools Appl 80:8091–8126
46. Jocher G, Chaurasia A, Stoken A et al (2022) ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference. https://doi.org/10.5281/ZENODO.6222936
47. Wang CY, Bochkovskiy A, Liao HYM (2020) Scaled-YOLOv4: scaling cross stage partial network. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp 13024–13033. https://doi.org/10.48550/arxiv.2011.08036
48. Wang CY, Mark Liao HY, Wu YH et al (2020) CSPNet: a new backbone that can enhance learning capability of CNN. IEEE Comput Soc Conf Comput Vis Pattern Recognit Work 2020-June, pp 1571–1580. https://doi.org/10.1109/CVPRW50498.2020.00203
49. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 2016-Decem, pp 770–778. https://doi.org/10.1109/CVPR.2016.90
50. Karaboga D, Gorkemli B, Ozturk C, Karaboga N (2014) A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif Intell Rev 42:21–57. https://doi.org/10.1007/s10462-012-9328-0
51. Lin TY, Maire M, Belongie S et al (2014) Microsoft COCO: common objects in context. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 8693 LNCS, pp 740–755. https://doi.org/10.1007/978-3-319-10602-1_48
52. WongKinYiu/ScaledYOLOv4 at yolov4-large. https://github.com/WongKinYiu/ScaledYOLOv4/tree/yolov4-large. Accessed 2 Apr 2021
53. AlexeyAB/darknet : YOLOv4 / Scaled-YOLOv4 / YOLO - Neural Networks for Object Detection (Windows and Linux version of Darknet). https://github.com/AlexeyAB/darknet. Accessed 2 Apr 2021
54. Reboiro-Jato A, Glez-Peña M, Lee K-S et al (2022) Citation: Nogueira-Rodríguez performance of convolutional neural networks for polyp localization on public colonoscopy image datasets. https://doi.org/10.3390/diagnostics12040898