




Underwater object detection: architectures and algorithms – a comprehensive review

Sheezan Fayaz¹ · Shabir A. Parah¹  · G. J. Qureshi²

Received: 10 March 2021 / Revised: 16 June 2021 / Accepted: 25 January 2022 /
Published online: 12 March 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

Underwater object detection is an essential step in image processing and it plays a vital role in several applications such as the repair and maintenance of sub-aquatic structures and marine sciences. Many computer vision-based solutions have been proposed but an optimal solution for underwater object detection and species classification does not exist. This is mainly because of the challenges presented by the underwater environment which mainly include light scattering and light absorption. The advent of deep learning has enabled researchers to solve various problems like protection of the subaquatic ecological environment, emergency rescue, reducing chances of underwater disaster and its prevention, underwater target detection, spooring, and recognition. However, the advantages and shortcomings of these deep learning algorithms are still unclear. Thus, to give a clearer view of the underwater object detection algorithms and their pros and cons, we proffer a state-of-the-art review of different computer vision-based approaches that have been developed as yet. Besides, a comparison of various state-of-the-art schemes is made based on various objective indices and future research directions in the field of underwater object detection have also been proffered.

Keywords Object detection · Light absorption · Light scattering · Deep-learning

1 Introduction

Water bodies cover almost two-thirds of the earth's surface producing almost half of oxygen and absorbing the maximum amount of carbon dioxide from the environment. To maintain these water bodies and other underwater ecosystem services, we require to monitor critical underwater habitats. Underwater ocular imaging technology provides the immense potential to

✉ Shabir A. Parah
shabireltr@gmail.com

¹ Department of Electronics and Inst. Technology, University of Kashmir, Hazratbal, Srinagar, India

² Higher Education Department, Government of J and K, Srinagar, India

monitor subaquatic scenes more efficiently, in terms of both time and cost. Several management strategies for the underwater environment employ remote sensing and spooing of subaquatic species and their habitats. In the last few years, the availability of subaquatic imagery has increased exponentially due to the use of autonomous underwater vehicles (AUV), digital cameras, and unmanned underwater vehicles (UUV) [79]. Millions of subaquatic pictures of coral reefs have been captured by the Integrated Marine Observing System (IMOS) around Australia, however, not more than 5% of them go through expert underwater analysis. For the National Oceanic and Atmospheric Administration, it is even lesser, just 1 to 2% [8]. For the aforementioned reasons, it is now a high time and research priority to automatically monitor and analyze underwater digital data. To perform such research and solve underwater issues, the state-of-the-art area of machine learning called deep learning offers potentially unparalleled opportunities for several subaquatic objects [89]. So far Manually designed low-level features have been exploited in conventional classification. Moreover, Linear Discriminant Analysis (LDA), Support Vector Machine (SVM), Principal Component Analysis (PCA), and other conventional machine learning tools get immediately saturated when the volume of training data increases. Hinton et al. [48] proffered learning of features by using deep neural networks (DNNs) to reduce and eliminate the shortcomings in conventional machine learning networks. For texts, images, sounds, etc., to make sense, deep learning algorithms transform input data via more layers than algorithms based on shallow learning [23]. Every layer transforms the signal using a processing unit, such as an artificial neuron, which learns the parameters via training [102]. Handcrafted features are being replaced by efficient deep learning algorithms for learning features and for hierarchical extraction of features [107]. Methods of deep learning represent an observation (e.g., an image) in a better way and create models for learning these representations from huge data. By using an ample amount of data for training, deep and big networks showed excellent performance. As an example, a convolutional neural network (CNN) trained via ImageNet has achieved unequalled precision in picture classification [61]. CNNs have been used in the area of image classification [61], object detection [64], digits and traffic signs recognition [18], face verification [72], etc., and showed excellent performance. However, algorithms based on deep learning have not been much used in underwater object detection and classification. A study on the present deep learning techniques for various underwater object detection and classification will help the researchers to know the challenges and survey highly efficient possibilities. Roughly about two-thirds of the surface of the earth is covered by water [132], however, comparatively not much of technologies related to underwater research have been exhaustively explored [73]. Besides, more importantly, security including naval battle, shipwreck, etc., is a momentous issue, therefore, for underwater surveillance marine object detection technologies are of utmost importance.

Humans can quickly spot scenes (sea-water, mountains), objects (sailboat, cruise), and visual details when seeing an image. However, for detecting marine objects in an image or video without manual intervention, a computer vision technique known as object detection is used. Object detection is mainly used to detect objects in a picture just like humans do, and then to instruct a computer so that it gets an understanding of what a picture contains. An object in an image can be detected from the back view, side view, and front view [105]. Conventional methods like manually classifying and recognizing objects in an underwater image, conventional statistical analysis and simulating ocean model, etc. highly rely on the availability of optical characteristics and are imprecise and inefficient for processing huge underwater data. On contrary, methods based on deep learning can process huge data meeting

the requirements of accurate and quick analysis of immense underwater data. Therefore, deep learning enables researchers to solve various underwater problems like protection of the underwater ecological environment, emergency rescue, underwater disaster mitigation and prevention, underwater target detection, spoofing, and recognition. Deep learning algorithms have been extensively used for marine systems like marine data classification and recognition (Convolutional Neural Network- CNN) [30], marine data reconstruction (CNN) [29], marine data prediction (Recurrent Neural Network-RNN), (CNN) [128]. The algorithm based on deep learning which is presented by Duo et al. [30] involves three stages: the stage of pre-processing, the network, and lastly the eddy extraction. In the pre-processing step, the remote sensing satellite altimeter data is pre-processed and the precise and small size sample data is improved to acquire the training set; subsequently, in the network stage, there is a deep learning-based integration model along with a network for object detection forming the main part. To facilitate the further survey, the trained model has been employed in the final stage for detection of the eddy extent and produce the coordinates of effective contour and eddy center. This algorithm produces a whole eddy detection network, that can efficiently employ remote sensing data. The shortcoming of this algorithm is that it is only suitable for Archiving, Validation, and Interpretation of Satellite Oceanographic (AVISO) sea level products. Ducourmau et al. [29] addresses the downscaling of data from ocean remote sensing by employing models of picture super-resolution built on deep learning, and more specifically Convolutional Neural Networks (CNNs). The main aim of the algorithm is to assess the relevance and the efficiency of deep learning networks that are applied to data from ocean remote sensing. Also, the focus has been on Sea Surface Temperature (SST) data derived from satellites.

Recent upgrades in RNN algorithms proffer significant solutions for problems of sequence prediction. The architecture of Long Short-Term Memory (LSTM) presents an improvisation to the RNN hidden layer and is successfully employed to carry out different tasks like supervised sequence learning. A prediction network that can completely utilize the Spatio-temporal information contained by the image sequences has been immensely desired. Thus, a prediction framework that employs the combination of the spatial and temporal information, called Combined Fully Connected Long Short-Term Memory and Convolution Neural Network model (CFCC-LSTM) has been proposed. So, Yang et al. [128] proposed a prediction network that brings together the spatial and temporal information, which is the Combined Fully Connected Long Short-Term Memory (FC-LSTM) and Convolution Neural Network model (CFCC-LSTM). This model consists of a single FC-LSTM and a single convolution layer. Firstly, a CFCC-LSTM network is proposed which aims at solving problems in sequence prediction, particularly for complicated SST pictures to fulfill the network requirement. Secondly, two data sets are introduced, the Bohai Sea data set and the China Ocean data set [127] to fulfill the data requirement.

These aforementioned algorithms mainly employ CNN. CNN network consumes a lot of computational time to perform prediction accurately because it uses multiple regions as input to perform object detection. These algorithms are not suitable for real-time underwater object detection. You Only Look Once (YOLO) based algorithms do not process multiple regions of the same image unlike RCNN, rather look at the complete image at once making the object detection task less complex and less time-consuming. However, not much of the work and research has been carried out in the area of underwater object detection using YOLO.

The existing deep learning algorithms employed for underwater images require a huge number of images and videos that are of high quality, as the high-quality videos and images

often have better discriminant features. Nevertheless, the effect of extensively changing environmental underwater conditions, such as turbid water conditions, amount of light, sophisticated background, range, and viewing angle is the major challenging task for acquiring high-quality pictures and videos. Therefore, research works to develop a unified model or framework are immensely required, by combining three steps: picture pre-processing, extracting feature, and classification of underwater object recognition task so that all the underwater images acquired by camera or some other image capturing equipment can be directly given to models. It is speculated the new model will substantially decrease the need for pictures. Besides, presently several approaches that employ the idea of deep learning to solve the problems of underwater object detection are built on transfer learning which means to apply the classic picture database like COCO or ImageNet for training the algorithm, and after that giving the new underwater dataset to the pre-trained model and properly tuning the model to practically apply it. Nevertheless, this task consumes a lot of time and is hard to perform, particularly in the pre-training process. ImageNet pre-training speeds up convergence in training, however, it does not surely proffer regularization or enhance the accuracy in the final target task. Thus, these points that have been highlighted above encourage researchers to think about the need for fine-tuning and pre-training.

The remaining portion of the paper is organized as; Section II discusses the motivation and contribution; Section III proffers the basics of computer vision and deep learning. Section IV discusses the concept of underwater object detection and presents the various existing deep learning methods used for underwater object detection. Section V discusses the experimental results and shows the comparison between some popular deep learning algorithms used for underwater object detection. Section VI presents the challenges. Section VII proffers the conclusion and future directions.

2 Motivation and contribution

With the help of deep networks having multiple levels of non-linearities, discriminative features and high-level abstractions can be learned from various complex datasets autonomously and effectively [131]. Also, the excellent computation of powerful GPU has highly boosted the efficiency of deep learning algorithms. Over the years apparently, several deep learning methods and algorithms for underwater object detection have been presented. However, the advantages and shortcomings of these methods have not been summarized in a single contribution. Also, future challenges and trends of the majority of such works have not been summarized in a single paper to our best of the knoweledge. Keeping in view these issues, an in-depth review of underwater object detection using deep learning methods is opportune and has both practical and theoretical importance in the ocean engineering community. The aim of this review is to highlight some renowned deep learning underwater object detection algorithms that can aid and pave the way for future research work in this particular field.

The major benefactions of this review are as follows:

1. An understandable and profound review of highly renowned deep learning algorithms for underwater object detection is presented.
2. The architectures and advantages of each algorithm have been thoroughly discussed and analyzed.

3. Experimental results of different deep learning algorithms for subaquatic object detection are inclusively reviewed and compared.
4. Futuristic trends and the possible challenges in marine object detection using deep learning techniques are robustly discussed.

3 Brief idea of computer vision and deep learning

Computer vision In computer vision computers incorporated with imaging sensors are used to mimic visual functions of humans that draw out features from the acquired data set, examine and classify them to help in process of decision making. Several fields of knowledge like image processing, high-level computer programming, artificial intelligence (AI), so on, are usually involved in computer vision. As an example, manufacturing industries use it to point out defections or enhance the quality [17, 57]. Also, there are efficient applications of emotion observation and face detection at the places like airports and various security checkpoints [10, 19, 20]. Doctors in the medical field make use of certain kinds of software for diagnoses to identify abnormal tissues and tumors using medical imaging [1]. An agricultural industry uses computer vision for systems meant for decision making to predict the total yield from the field [121]. Furthermore, a self-driving car is being designed by google having almost a visual range of 328 ft. This kind of car can also recognize traffic signals and avoid pedestrians as well [83]. Several state-of-the-art algorithms show that computer vision is modifying our day-to-day lives.

Deep learning The word deep in deep learning means a neural network employs many layers to imitate the human brain. Some algorithms that are based on deep learning using neural networks are Novel Nonlinear Hypothesis for the Delta Parallel Robot Modelling 2020 [6], SOFMLS: Online Self-Organizing Fuzzy Modified Least-Squares Network, 2009 [97], Wavelet-Based EEG Processing for Epilepsy Detection Using Fuzzy Entropy and Associative Petri Net 2019 [16], Stability Analysis of the Modified Levenberg-Marquardt Algorithm for the Artificial Neural Network Training 2020 [98], On the Estimation and Control of Nonlinear Systems With Parametric Uncertainties and Noisy Outputs, 2018 [84], and CNN based detectors on planetary environments: a performance evaluation, 2020 [33]. The idea of deep learning using the neural network has come to light some ten years ago. In 1998, deep learning was originally given and developed by Lecun et al. [67]. They developed a classifier that was five-layered and was known as LeNet5. It was based on a Convolutional Neural Network (CNN). Initially, based on the Modified National Institute of Standards and Technology (MNIST) dataset, LeNet was employed to detect hand-written bank cheques. In neural networks, activation functions and fully connected frameworks were already known. LeNet5 model brought in convolutional and pooling layers and it is thought to be the plinth for all convolutional networks. The LeNet5 network has two sets of layers i.e., convolutional layer and average pooling layer which is then followed by a flattening convolutional layer. Subsequently, there are two fully connected layers placed and lastly, a SoftMax classifier is employed. The input to the LeNet5 model is a grayscale image of size 32×32 . This image is passed via the initial convolutional layer having six filters or feature maps of dimension 5×5 and a stride equal to one. The size of the image gets changed from $32 \times 32 \times 1$ to $28 \times 28 \times 6$. Then average pooling is applied using a filter of size 2×2 and stride = 2. The resulting image dimensions will be reduced to $14 \times 14 \times 6$. In the next step, another convolutional

layer having sixteen feature maps and dimension 5×5 is employed with a stride = 1. In this stage, just ten feature maps out of sixteen feature maps are connected with six previous layer feature maps. Again, the fourth layer is a layer of average pooling having filter dimension 2×2 and stride = 2. The fifth layer is a convolutional layer that is fully connected having 1×1 sized 120 feature maps. This layer is followed by a fully connected layer and fully connected SoftMax layer \hat{y} with ten possible values that correspond to the ten digits ranging from 0 to 9. It proffers an error rate of 0.95% on test data. Figure 1 given below shows the process flow of LeNet5.

The experimental results of LeNet5 have been highlighted in Fig. 2. The graphs in the given figure show the training and validation loss vs several epochs and accuracy vs several epochs.

MNIST dataset This database has handwritten digits and 60,000 examples as a training set, and 10,000 examples as a test set. MNIST is considered a subset of a bigger set from the National Institute of Standards and Technology (NIST). The digits are centered in a picture of fixed size and also size-normalized. The MNIST dataset was developed from Special Database-3 and Special Database-1 of NIST which consist of binary pictures of handwritten digits. Originally, NIST defined SD-1 as the test set and SD-3 as the training set but SD-3 is easier and cleaner to be recognized as compared to SD-1. The basis for this is seen in the fact that Special Database-3 was gathered among Census Bureau employees. On the other hand, Special Database-1 was gathered among students from high school. To draw significant conclusions from experiments needs that the results should not depend on the selection of the test and training set among the entire collection of samples. Thus, it was mandatory to develop a novel dataset by blending datasets of NIST.

Deep learning has made immense achievements in the past years due to improvement in power used in computing and the explosion of huge data. Deep learning algorithms are based on immense data that is collected in a particular field. Resources used for learning from ample data are highly substantial. With the rise of efficient GPU, cloud storage, ASIC accelerators, and highly powerful computing facilities, it is now practicable to gather, manage, and examine massive datasets. The significance of massive data sets is that they reduce the chances of overfitting problems in deep learning. The improved computing power can increase the speed of the time-taking process of training. Deep learning algorithms are increasingly used in several fields and have considerable advantages over conventional object detection approaches in computer vision. Thus, the performance of several robotic systems is improved by utilizing deep learning. For example, Google's AlphaGo studied the learning behavior of humans and then competed with the popular Go player [7]. To use the concept of deep learning in the field of computer vision, adequate examples from pictures gathered beforehand is condemnatory. A good example would be ImageNet [2]. Conventional computer vision-based techniques

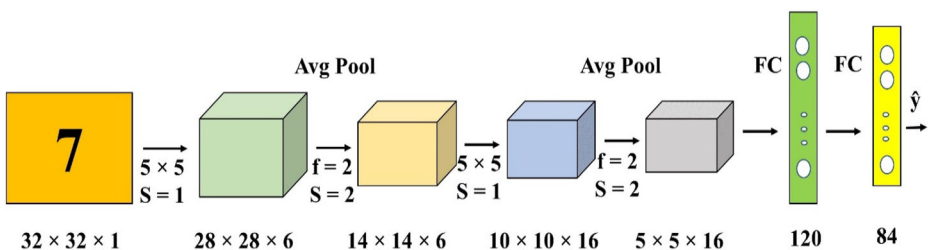


Fig. 1 Process flow of LeNet5

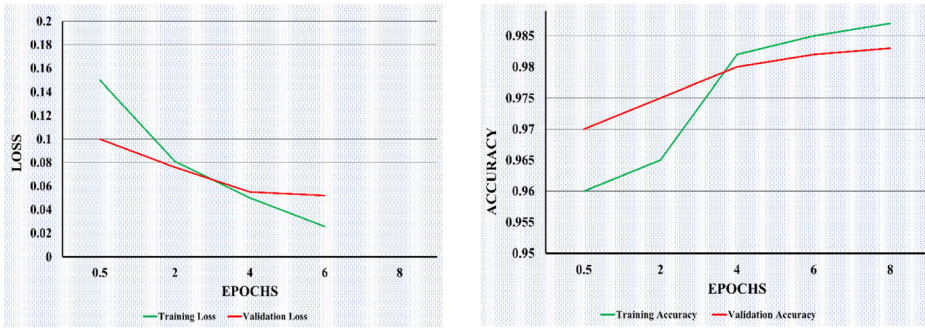


Fig. 2 Performance evaluation of LeNet5

suffered from the precision in the extraction of features, while approaches based on deep learning can be employed to improve the method via a neural network.

4 Underwater object detection

A computer vision approach employed for identifying and locating underwater objects in a subaquatic picture or video is known as underwater object detection. By using this sort of technique for identification and localization, underwater object detection can be utilized for counting objects in underwater scenes and also for determining and spooring their accurate locations by precisely labeling the objects in underwater images. To detect an underwater object, a bounding box is defined to locate an object in an underwater image. For obtaining the coordinates of the bounding box (X, Y) for an object in a subaquatic image a deep learning algorithm meant for underwater object detection is employed. Object detection not only informs us of what a subaquatic image has but also tells us where the object is in a subaquatic image. Figure 3 given below shows the results of underwater fish detection using deep learning algorithms [58, 119].

Ocean engineers frequently employ Automatic underwater Vehicles (AUVs) for subaquatic robotic capturing. In autonomous subaquatic capturing detection of underwater organisms is becoming increasingly important. Human divers usually capture seafood. However, diver fishing, not only leads to severe injury to divers' bodies but also in inefficient working, particularly when the water depth is higher than 20 mt. Robotic capturing via subaquatic robot to catch seafood has been proffered to solve the prevailing problem of subaquatic diver fishing. Figure 4 given below shows some unmanned underwater vehicles used for robotic capturing [86, 87].

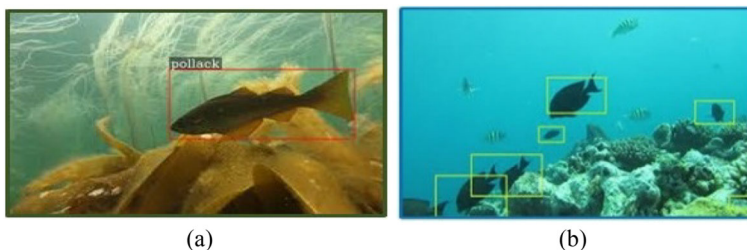


Fig. 3 **a** Pollack detection, **b** Coral reef fish detection [58, 119]



Fig. 4 Underwater autonomous vehicles [86, 87]

These vehicles not only decrease the chances of the divers' bodily injury but also decrease the seafood price. The method proposed by Han et al. [42] has been employed with such kind of a model as mentioned above, namely, an underwater remote operated vehicle (ROV). The entire network has been used for fishing underwater products. This underwater robot has a length of about 1 m, a width of about 0.8 m, and a weight of about 90 kg. The technique of gathering underwater products is of adsorption type; the real structure of this underwater robot is shown in Fig. 5. It is operated and controlled by remote. The main task to be performed is the detection of underwater objects and also, locating them.

Fishery robotics has drawn great attention and efforts of researchers due to the several merits of robotic capturing [59, 110, 113]. Norway has built a submarine employing a Remote Operated Vehicle (ROV) which enabled ocean engineers to realize sea urchin harvesting via remote aspiration manually. However, manipulation of the subaquatic robot is a tedious task and a tough problem. It requires a highly focused operator having rich experience. To reduce expenses and the challenges in operation, autonomous capturing is mandatory. In underwater object detection, algorithms based on conventional machine learning have been popular in past decades. As an example, Garcia et al. [35] performed object segmentation and identification via a process called generic segmentation. Sun et al. [108] presented an algorithm, namely, automatic recognition via shape-based identification and color-based identification. For conventional approaches, color and shape features are mostly considered for the detection and recognition of an object. However, subaquatic organisms exhibit different shapes in various subaquatic environments and due to ecological reasons, they have colors similar to the seabed scenes. The famous and efficient algorithms based on deep learning can enhance the perception ability to perceive underwater organisms. For example, Sermanet et al. [103] proffered the framework called the OverFeat algorithm, utilizing Convolution Neural Networks (CNNs) and multi-scale sliding windows to detect, recognize, and classify objects in an image. Ren et al. [95] proffered an algorithm, namely, Faster Region-based Convolutional Neural Network

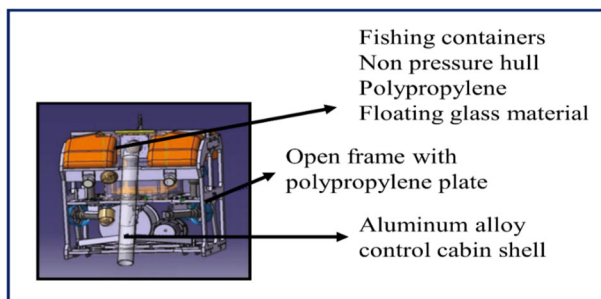


Fig. 5 Underwater ROV for fishing subaquatic organism [42]

(Faster RCNN) which uses a Region Proposal Network (RPN) to generate region proposal and then for classification and bounding box regression uses a CNN network. Figure 6 shown below highlights the image processing and object detection structure. This figure gives an overview of the basic structure of how images are processed and detected as well.

4.1 Deep learning algorithms for underwater object detection

Deep learning is a subset of a broader field called Machine Learning which mainly consists of artificial neural networks. Figure 7 given below highlights the general deep learning framework and mathematics behind it is discussed in the following portion.

Every neuron is portioned into two blocks:

- i) Calculating z by using input x_i where $x_i = \times 1, \times 2, \times 3, \times 4$ as shown in figure above:

$$z = \sum_i w_i * x_i + b \tag{1}$$

- ii) Calculating a using z :

$$a = \Psi(z) \tag{2}$$

Where w_i denotes the weights, b represents the bias, and Ψ is the activation function.

Learning process in deep learning:

The process of learning in a deep neural network is a step of computing weights of parameters that are associated with different regressions all over the framework. The main goal is to look for the best parameters which proffer the best approximation/prediction, beginning from the real value input. For this purpose, an objective function known as Loss Function is defined and it is represented as J . This parameter quantifies the amount of distance between the predicted values and the real values over the entire training dataset.

Two main steps are followed to minimize the value of J :

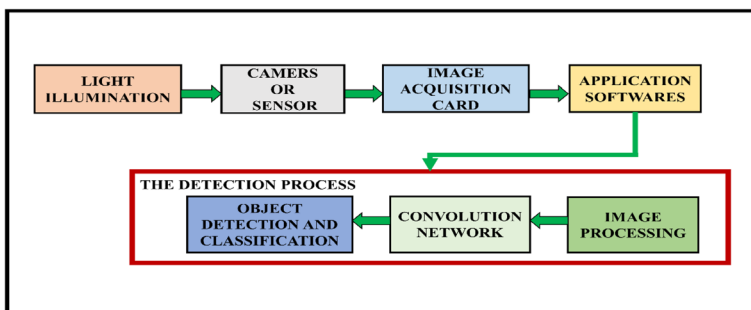


Fig. 6 Image processing and object detection structure

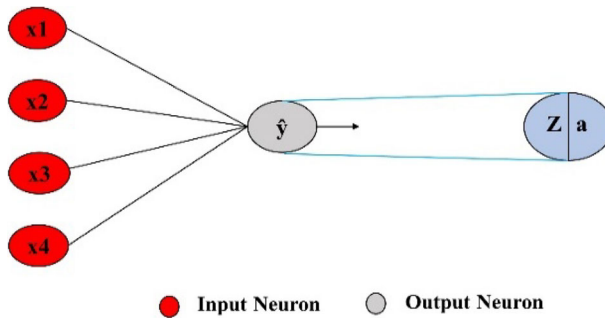


Fig. 7 Basic process flow of deep learning algorithm

- i) **Forward-Propagation:** The data is propagated via the framework either as a whole or as batches, and the loss function is computed on the batch. This loss function is defined as the summation of the inaccuracies that have been committed at the output that is being predicted for various rows.
- ii) **Back-propagation:** It involves the calculation of the cost function gradients with respect to various parameters. After this, a descent algorithm is applied for updating them.

The same process which repeats many times is called epoch number. The learning process is indicated as given below:

- Initializing the network parameters.
- For $i = 1, 2, \dots, N$: (N represents the number of epochs)
- Performing forward-propagation:
- $\forall i$, calculate the predicted value of input x_i via deep neural network: \hat{y}_i^θ
- Evaluating the function:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}_i^\theta, y_i) \quad (3)$$

Where m represents the training set size, θ indicates network parameters, L represents the cost^(*) function, (*) cost function L computes the number of distances that are between the predicted value and a real value over a single point, and y_i is the actual output.

- Performing back-propagation:
- Applying descent approach for updating parameters:

$$\theta = G(\theta) \quad (4)$$

To improve the real-time efficiency of underwater object detection, various deep learning algorithms are proposed to make the computer able to handle and deal with the subaquatic picture information in lesser time.

Usually, an advanced object detector is made of two components. Firstly, it consists of a backbone that is pre-trained using ImageNet. Secondly, it consists of a head that is employed to perform the prediction of object classes and their bounding boxes. Object detectors that run on GPU can have a backbone network like ResNet [46], VGG [106], DenseNet [54], or ResNeXt [125]. Object detectors that run on CPU, can have backbone network like SqueezeNet [63], ShuffleNet [81, 136], or MobileNet [51, 52, 101, 116]. The head portion is often classified into two types, namely, one-stage object detector and two-stage object detector. The most common object detector that is two-stage is the RCNN [39] series, also fast RCNN [38], faster RCNN [95], RFCN [22], and Libra RCNN [88]. It is possible to set up an anchor-free object detector that is a two-stage object detector, e.g., RepPoints [129]. The most common models for one-stage object detector are SSD [74], YOLO [91, 93, 94], and RetinaNet [70]. In the past few years, one-stage anchor-free object detectors have been developed. The object detectors of this type are CornerNet [65, 66], CenterNet [28], FCOS [118], etc. The detectors developed for object detection in the past few years usually insert few layers in between the backbone and head. These layers are often employed for the collection of feature maps from various stages. It can be called the neck of a detector employed for object detection. A neck usually consists of many bottom-up and top-down paths. Those networks which are equipped with such a mechanism include Path Aggregation Network (PAN) [75], Feature Pyramid Network (FPN) [71], Bidirectional Feature Pyramid Network (BiFPN) [117], and NAS-FPN [37]. Besides, some researchers focus on directly setting up a novel backbone (DetNet [68], DetNAS [14]) or a novel entire model (HitDetector [41], SpineNet [27]) to detect objects. Summing up, an ordinary detector used for object detection consists of many parts:

- Input: Picture, Patches, Picture Pyramid
- Backbones: ResNet-50 [46], VGG16 [106], SpineNet [27], CSPResNeXt50 [122], EfficientNet-B0/B7 [114], CSPDarknet53 [122]
- Neck:
- Added blocks: Atrous Spatial Pyramid Pooling (ASPP) [13], Spatial Pyramid Pooling SPP [44], Receptive Field Block (RFB) [76], Spatial Attention Module (SAM) [123]
- Blocks for Path-aggregation: PAN [75], FPN [71], Fully-connected FPN, NAS-FPN [37], Adaptively Spatial Feature Fusion (ASFF) [78], Bidirectional Feature Pyramid Network (BiFPN) [117], SFAM [137]
- Heads:
- (one-stage) Dense Prediction:
- RPN [95], SSD [74], RetinaNet [70] (anchor based), YOLO [94]
- CornerNet [65], CenterNet [28], FCOS [118] (anchor free), MatrixNet [90]
- (Two-stage) Sparse Prediction:
- Faster R-CNN [95], RFCN [22], Mask RCNN [47] (anchor based)
- RepPoints [129] (anchor free).

Some well-known deep learning techniques particularly those employing deep neural networks for digital underwater image detection and classification have been presented in this section. Each of the algorithms is highlighted in Fig. 8 and also discussed in the following portion of the paper.

The convolution operation performed between the filter and the input image gives a 2-dimensional matrix in which every element is calculated by summation of the elementwise product of the filter elements and the image elements spanned by the filter. Mathematically, for a given filter and picture we have:

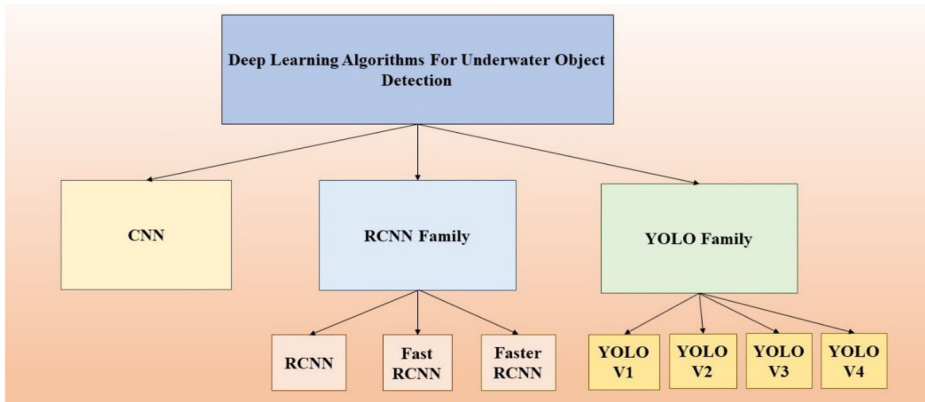


Fig. 8 Some deep learning algorithms used for underwater object detection

$$\text{conv}(I, K)_{x,y} = \sum_{i=1}^{n_H} \sum_{j=1}^{n_W} \sum_{k=1}^{n_C} K_{i,j,k} I_{x+i-1,y+j-1,k} \tag{5}$$

$$\text{dim}(\text{conv}(I, K)) = \left(\left\lfloor \frac{n_H + 2p-f}{s} + 1 \right\rfloor \right); s > 0 = (n_H + 2p-f, n_W + 2p-f); s = 0 \tag{6}$$

Where n_H and n_W represent the size of height and width of the input image, respectively, n_C represents the number of channels. The filter/kernel K should have the number of channels equal to the number of channels in the input image. Also, the filter has odd dimensions represented by f , s represents the stride, p represents the padding and $\lfloor x \rfloor$ represents the floor function of x . I represents the input image.

RCNN at first draws out from the input picture several region proposals. CNN model is then employed to carry out forward-propagation over every region proposal for feature extraction. Subsequently, features of every region proposal are employed to predict the bounding box and the class of that region proposal.

You Only Look Once (YOLO) algorithm uses CNNs for real-time object detection. As can be seen from the name itself, this algorithm needs only one forward-propagation via a neural network for detecting objects. This implies that in one go or single run of an algorithm prediction in the whole image is performed. Simultaneously, bounding boxes and different class probabilities are predicted using the CNN network.

4.1.1 Convolutional neural network

A Convolutional neural network is the simplest and extensively used deep learning approach for object detection. Three basic components are involved in a convolutional neural network.

1. Convolutional Layer
2. Pooling Layer
3. Output Layer

The detailed description of each of these layers is as follows:

1. Convolutional Layer

In this layer, an image of size 6×6 (say) is subjected to a weight matrix that extracts certain features from the image. The weight matrix is a 3×3 square matrix and is run over the input image so that all pixels are spanned to generate the convolved output. In below Fig. 9, the output is acquired by summing the values acquired by the element-wise product of the weight matrix and the portion that is highlighted in the input image. Following the same steps of convolution for the whole image with the stride of 1, the 6×6 image will be converted to a 4×4 image. Pixels are used again during the sliding of the weight matrix over the entire image. This results in parameter sharing in CNN. Figure 9 shows how convolution operation with a stride of 1 works.

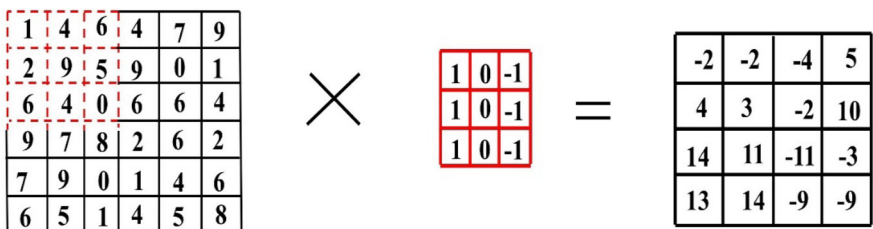
2. Pooling Layer

For large images, the number of trainable parameters must be reduced and for that purpose pooling layers are periodically introduced between successive convolution layers. Pooling is mainly done to reduce the spatial size of the image and it is applied independently on every depth dimension; therefore, the image depth remains unaltered. Max pooling is the most commonly used form of pooling layer.

3. The Output Layer

We require the output in the form of some class after several layers of convolution and pooling. These two layers can only extract features and decrease the number of parameters from the input images. However, to produce the output we have to include a fully connected layer to get an output equal to the number of required classes. It is difficult to have that number using just the convolution layers. Convolution layers produce 3-Dimensional activation maps but we just require the final output as to whether a picture belongs to a particular class or not. To calculate the inaccuracy in prediction, the final output layer has a loss function such as categorical cross-entropy. After the forward propagation is finished, the backpropagation starts for updating the weight and biases for error and reducing loss. Figure 10 given below shows how the entire network looks like.

An input image is given to the first convolutional layer. The convolved output is received as an activation map. The filters used in this layer take out relevant features from the given input picture and pass them further. Each filter will generate a different feature to help the accurate



STRIDE = 1

Fig. 9 Convolution operation with a stride of 1

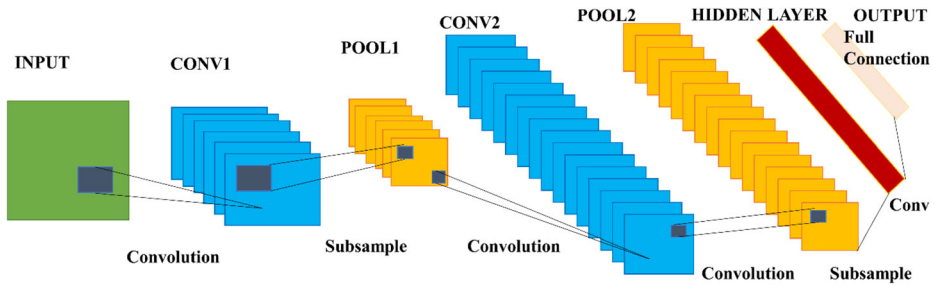


Fig. 10 Whole process flow of CNN

class prediction. If we want to retain the image size, we should use the same padding (zero padding), otherwise, we can use valid padding as it reduces the number of features. Then Pooling layers are used to further decrease the number of parameters. Many layers for convolution and pooling are used before the prediction is done. Convolutional layers are used to draw out features. As one goes deeper in this architecture more specific features are drawn out compared to a shallow network in which the drawn-out features are more generic. A Fully connected layer is the output layer in a convolutional neural network. Here the input obtained from the previous layers is flattened and then forwarded so as the output is transformed into various classes as desired by the algorithm. The final output is then produced via the output layer. In the fully connected layer, which is the output layer a loss function is being defined to calculate the mean square loss. Finally, the error gradient is computed. To update the bias values and filter (weights) the error is backpropagated. Therefore, in one forward and backward pass one cycle of training is completed.

For underwater object detection using CNN,

- Firstly, we take an underwater image as an input image.
- Then we split this image into multiple regions.
- We then consider each region as an individual underwater image.
- All these regions (pictures) are passed to the CNN network and classified into different classes.

As all the regions have been divided into a particular class, therefore, all these regions or images are combined to obtain the original input picture with detected underwater objects.

Many researchers have exploited CNN architecture for underwater object detection. Krizhevsky et al. [62] used the CNN model for dealing with the classification process and won the challenge of ILSVRC (ImageNet Large Scale Visual Recognition Challenge), which decreased the top 5 rates of the error to the percentage of 15.3. Thus, since then deep CNN has been extensively used. Elawady et al. [31] have utilized supervised CNNs for coral classification and worked on Heriot-Watt University's Atlantic Deep-Sea Dataset and Moorea labeled Corals and calculated Phase Congruency (PC), Weber Local Descriptor (WLD), and Zero Component Analysis (ZCA). They also took into consideration texture and shape for input pictures with spatial channels of the color [31]. Mahmood et al. [82] presented a scheme for feature extraction that is built on Spatial Pyramid Pooling (SPP) to make the traditional point-annotated underwater data compatible with the input constraints of CNNs. Sermanet et al. [103] proffered the technique utilizing multi-scale sliding windows and Convolution Neural Networks (CNNs) for object recognition, detection, and classification. Suxia et al. [21] in 2019

presented an approach for underwater object detection using CNN. This network has been employed to detect a fish in a blurry environment underwater. During the process of convolution, feature map size is to be considered. Three major factors affect feature map size and they are padding, depth, and stride. Figure 11 given below shows the extraction of the feature map with a depth of 3, the stride of 1, and zero padding [25, 26].

Two types of connections can be usually seen between two layers that are adjacent in a complex neural network and these are the locally connected layer and the fully connected layer as shown in Fig. 12.

In a neural network that is fully connected, all picture elements in the input image have a connection with every neuron of the hidden layer as indicated in Fig. 12a. In the CNN network, the two of the last layers are commonly fully connected and they are the SoftMax layer and the output layer, respectively. A massive number of parameters will lead to an increased amount of computation and will also delay the processing. In a neural network that is locally connected, only some of the picture elements in the input image have a connection with the hidden layer neuron as indicated in Fig. 12b. This kind of connection will increase the speed of the system and decrease the number of connections. For local or full connectivity in the CNN model used by Suxia et al. [21] in 2019, the parameters for each layer are given in Table 1.

System validation using ImageNET dataset Authors [21] downloaded pictures from renowned ImageNet ILSVRC [3] to perform a system validation testing via object classification before using the system for the ocean fish data set developed in their research. There are about 500 pictures having about 20 classes that range from frog, coral, fish, sea turtle, ship, etc. Here every RGB picture is rescaled to 448×448 . Ground truth pictures are acquired manually from operating Labelling software. All images are split into a grid of 7×7 cells. Every cell predicts the location of two bounding boxes and information of class composed of a $1 \times 1 \times 30$ vector. It is a vector that consists of coordinates of object center (x,y), height h , and width w , coordinates of the predicted probabilities of the underwater object, and confidence scores of the bounding box. To predict the location of the target picture, the target is exhibited in a bounding box. Errors are always present there between the predictions and ground truth. Errors are measured using the Loss function which consists of three: (Intersection over union) IoU error, coordinate error, and class error. Equation (7) given below shows the mathematical formula for the loss function.

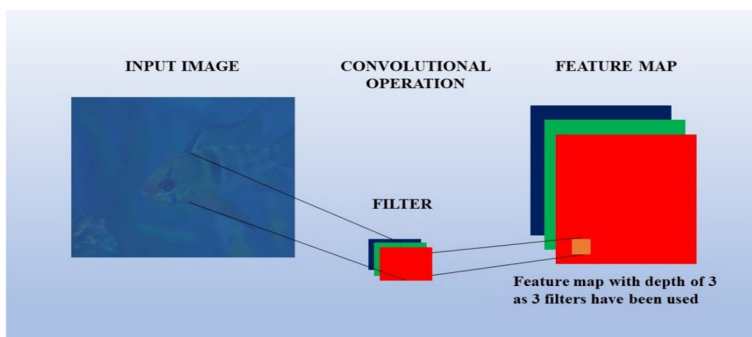


Fig. 11 Feature extractor using convolutional operation [25, 26]

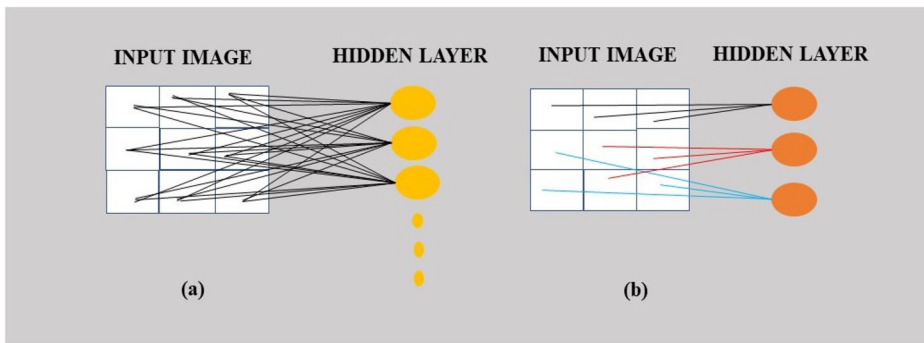


Fig. 12 **a** Fully connected neural network, **b** Locally connected neural network

$$\text{Loss} = \sum_{i=0}^{s^2} \text{CoordError} + \text{IOUError} + \text{ClassError} \quad (7)$$

Where CoordError denotes the error in predicting the coordinates, ClassError denotes the inaccuracy in the predicted class and IoU measures the accuracy in position as indicated in Fig. 13. The underwater image has been taken from the turbid dataset [5].

Table 1 Parameters in CNN model

Layer	Input size	Filter size	Stride	Output size
Conv. 1	$[N \times 448 \times 448 \times 3]$	$[7 \times 7 \times 3 \times 64]$	2	$[N \times 224 \times 224 \times 64]$
Maxpool 1	$[N \times 224 \times 224 \times 64]$	$[2 \times 2]$	2	$[N \times 112 \times 112 \times 32]$
Conv. 2	$[N \times 112 \times 112 \times 32]$	$[3 \times 3 \times 32 \times 192]$	1	$[N \times 112 \times 112 \times 192]$
Maxpool 2	$[N \times 112 \times 112 \times 192]$	$[2 \times 2]$	2	$[N \times 56 \times 56 \times 96]$
Conv. 3	$[N \times 56 \times 56 \times 96]$	$[3 \times 3 \times 128 \times 256]$	1	$[N \times 56 \times 56 \times 256]$
Conv. 4	$[N \times 56 \times 56 \times 128]$	$[3 \times 3 \times 128 \times 256]$	1	$[N \times 56 \times 56 \times 256]$
Conv. 5	$[N \times 56 \times 56 \times 256]$	$[1 \times 1 \times 256 \times 256]$	1	$[N \times 56 \times 56 \times 256]$
Conv. 6	$[N \times 56 \times 56 \times 256]$	$[3 \times 3 \times 256 \times 512]$	1	$[N \times 56 \times 56 \times 512]$
Maxpool 3	$[N \times 56 \times 56 \times 512]$	$[2 \times 2]$	2	$[N \times 28 \times 28 \times 256]$
Conv. 7	$[N \times 28 \times 28 \times 256]$	$[1 \times 1 \times 256 \times 256]$	1	$[N \times 28 \times 28 \times 256]$
Conv. 8	$[N \times 28 \times 28 \times 256]$	$[1 \times 1 \times 256 \times 256]$	1	$[N \times 28 \times 28 \times 256]$
Conv. 9	$[N \times 28 \times 28 \times 256]$	$[1 \times 1 \times 256 \times 256]$	1	$[N \times 28 \times 28 \times 256]$
Conv. 10	$[N \times 28 \times 28 \times 256]$	$[1 \times 1 \times 256 \times 256]$	1	$[N \times 28 \times 28 \times 256]$
Conv. 11	$[N \times 28 \times 28 \times 256]$	$[3 \times 3 \times 256 \times 512]$	1	$[N \times 28 \times 28 \times 512]$
Conv. 12	$[N \times 28 \times 28 \times 512]$	$[3 \times 3 \times 512 \times 512]$	1	$[N \times 28 \times 28 \times 512]$
Conv. 13	$[N \times 28 \times 28 \times 512]$	$[3 \times 3 \times 512 \times 512]$	1	$[N \times 28 \times 28 \times 512]$
Conv. 14	$[N \times 28 \times 28 \times 512]$	$[3 \times 3 \times 512 \times 512]$	1	$[N \times 28 \times 28 \times 512]$
Conv. 15	$[N \times 28 \times 28 \times 512]$	$[1 \times 1 \times 512 \times 512]$	1	$[N \times 28 \times 28 \times 512]$
Conv. 16	$[N \times 28 \times 28 \times 512]$	$[3 \times 3 \times 512 \times 1024]$	1	$[N \times 28 \times 28 \times 1024]$
Maxpool 4	$[N \times 28 \times 28 \times 1024]$	$[2 \times 2]$	2	$[N \times 14 \times 14 \times 512]$
Conv. 17	$[N \times 14 \times 14 \times 512]$	$[1 \times 1 \times 512 \times 256]$	1	$[N \times 14 \times 14 \times 256]$
Conv. 18	$[N \times 14 \times 14 \times 256]$	$[3 \times 3 \times 256 \times 1024]$	1	$[N \times 14 \times 14 \times 1024]$
Conv. 19	$[N \times 14 \times 14 \times 1024]$	$[1 \times 1 \times 1024 \times 512]$	1	$[N \times 14 \times 14 \times 512]$
Conv. 20	$[N \times 14 \times 14 \times 512]$	$[3 \times 3 \times 512 \times 1024]$	1	$[N \times 14 \times 14 \times 512]$
Conv. 21	$[N \times 14 \times 14 \times 512]$	$[3 \times 3 \times 512 \times 1024]$	1	$[N \times 14 \times 14 \times 1024]$
Conv. 22	$[N \times 14 \times 14 \times 1024]$	$[3 \times 3 \times 1024 \times 1024]$	1	$[N \times 7 \times 7 \times 1024]$
Conv. 23	$[N \times 7 \times 7 \times 1024]$	$[3 \times 3 \times 1024 \times 1024]$	2	$[N \times 7 \times 7 \times 1024]$
Conv. 24	$[N \times 7 \times 7 \times 1024]$	$[3 \times 3 \times 1024 \times 1024]$	1	$[N \times 7 \times 7 \times 1024]$
Fully conn.1	$[N \times 7 \times 7 \times 1024]$	$[1024 \times 4]$	Multi	$[1 \times 4096]$
Fully conn. 2	$[1 \times 4096]$	$[4096 \times 7 \times 7 \times 30]$	Multi	$[7 \times 7 \times 30]$

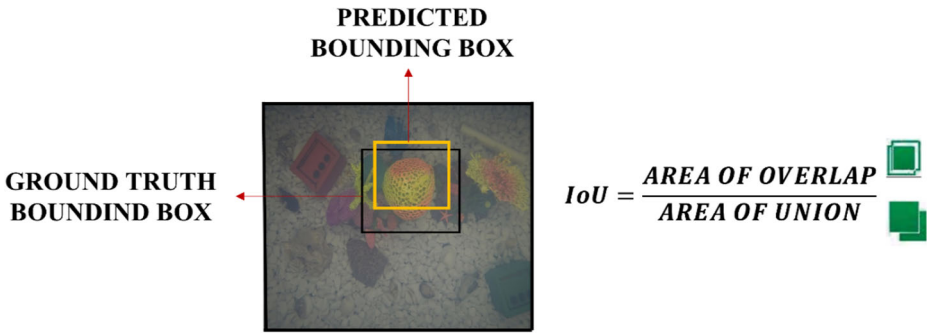


Fig. 13 Intersection over union

K bounding boxes will be predicted by every grid cell in a picture. These bounding boxes enclose an underwater object to predict the object class and localization. Besides, confidence is associated with every bounding box, and a confidence score is not associated with the class of undersea objects. This score just shows how accurately the predicted bounding box encloses the real undersea object.

$$\text{Confidence} = \text{Pr}(\text{Object}) \times \text{IoU} \tag{8}$$

where Pr denotes simply the probability and Pr(Object) denotes the probability of the underwater object of interest. Its value is 1 if an object is in the grid cell, and its value is 0 if the object is not present in the grid cell. Most of the time, the loss function is calculated as the summation of squared errors [53]. It has three parts namely, confidence errors, probabilities errors, and localization errors.

$$\begin{aligned} \text{Loss} = & \sum_{i=0}^{s^2} \sum_{j=0}^B \left[(x_i - x'_i)^2 + (y_i - y'_i)^2 + (w_i - w'_i)^2 + (h_i - h'_i)^2 \right] \\ & + \sum_{i=0}^{s^2} \sum_{j=0}^B (c_i - c'_i)^2 + \sum_{i=0}^{s^2} \sum_{c \in \text{class}} (P_i(c_i) - P_i(c'_i))^2 \end{aligned} \tag{9}$$

where h_i, w_i , denote the height and width of the ground truth bounding box; x_i, y_i denote the ground truth coordinates of the center of underwater objects; h'_i, w'_i represent the height and width of the predicted bounding box, x'_i, y'_i represent the predicted coordinates of the center of underwater objects.

Ground truth preparation for real ocean environment Suxia et al. [21] after testing the CNN model with nondegraded pictures without noise created their fish dataset from underwater. It has been tough for them to get pictures of other kinds like coral, sea turtle, etc. In the portion of their research, the only underwater object to be detected is fish. To collect about 410 pictures underwater, they have obtained several fish in a single underwater picture, therefore, the detection is quite challenging. The similar approach was selected to create the collection of ground truth pictures.

4.1.2 Networks based on region-based convolutional neural network (RCNN)

The deep learning algorithms based on the CNN framework are employed in several fields. Presently, in different engineering research areas RCNN, Fast RCNN, Faster RCNN, and some

advanced versions are extensively used. Overall, in the area of image recognition, the advent of the network built on CNN is fast.

1. Region-Based CNN (RCNN)

Girshick (facebook expert) in 2014, presented the Regional Convolution Neural Network (RCNN) which is the combination of the CNN model and Region Proposal Network [40]. The results of object detection on the dataset namely, PASCAL VOC2007 reached about 66% mean Average Precision (mAP). Based on RCNN, He et al. in 2015 proffered the SPP-Net framework [44], which immensely improved the efficiency of object detection.

RCNN proffers a bunch of boxes in the underwater picture and examines if any one of the boxes has an underwater object. The RCNN model does not work on a huge number of regions. It performs a selective survey to draw out these boxes from an underwater picture and these extracted boxes are known as regions. Four regions contribute to the formation of an object and these are textures, colors, enclosure, and varying scales. In selective search, these patterns are identified in the underwater picture and different regions are proffered based on that. A Brief overview of how selective search for underwater images is carried out is given below:

- Initially, it takes an underwater image. The underwater image given below in Fig. 14 has been taken from Turbid Dataset [5].
- In the second step, it produces initial sub-segmentations as shown in Fig. 15 such that several regions are obtained from this picture.
- This method then joins the regions that are similar based on texture resemblance, color resemblance, shape resemblance, and size resemblance to form a bigger region as indicated in Fig. 16.
- Lastly, the final location of the object called Region of Interest is produced by these regions.



Fig. 14 Underwater image [5]

Fig. 15 Initial sub-segments [104]

The steps carried out in RCNN for object detection are given below:

- a. A pre-trained convolutional neural network is first taken.
- b. In the second step, the CNN framework is retrained. The last layer is trained as per the number of classes that are to be detected.
- c. In the third step, the Region of Interest (ROI) for each picture is obtained. Every region is then reshaped so that it can match the input size of CNN.
- d. After all the regions are obtained, the Support Vector Machine (SVM) is trained so that it classifies background and underwater objects. Single binary SVM is trained for each class.
- e. Finally, a linear regression model is trained to produce bounding boxes that are tighter for every identified underwater object in the picture.

A better understanding of the working of an RCNN network can be achieved by following a visual example.

Firstly, an underwater image is taken as an input image. ROIs are then obtained by employing some proposal approach like selective search. This is indicated in Fig. 17.

Reshaping of all of these regions according to the input of CNN is then performed, and every region is passed through Conv-Net. CNN model then performs the extraction of features for every region and these regions are split into various classes using SVMs. Lastly, for every identified region bounding boxes are predicted using Bounding Box Regression (*Bbox reg*). This process is indicated in Fig. 18.

Fig. 16 Combined similar regions [104]

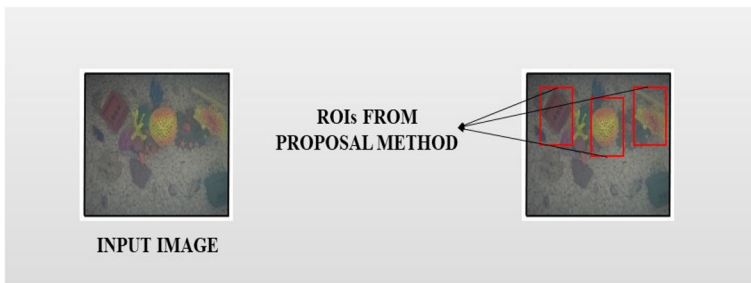


Fig. 17 An input underwater image and ROIs from the input underwater image, respectively [5]

Disadvantages of RCNN framework

This network has its shortcomings, RCNN can be used for underwater object detection though. To train an RCNN framework is a slow and costly process due to the below-mentioned steps:

- Based upon selective search extracting about 2000 picture regions for each picture.
- Feature extraction by employing a CNN framework for each picture region. For example, we have an N number of pictures, then there will be $N \times 2000$ number of CNN features.
- The whole object detection process employing RCNN involves three models:
 1. CNN to extract features.
 2. Classifier namely, Linear SVM to identify objects in the image.
 3. Regression framework to tighten the bounding boxes.

All the aforementioned operations together result in the slow processing of RCNN. This framework consumes around 40 to 50 s to do predictions for every new picture, which makes the framework burdensome and not practical to build if given a massive dataset.

2. Fast RCNN

Ross et al. in 2014 presented an approach known as Fast RCNN. This framework converts the problem of identifying an object into a regression problem [38] and the mean Average Precision (mAP) was improvised by about 30% as compared to the earlier best result of about 53.3% in a challenge in 2012 called ImageNet Large Scale Visual Recognition Challenge. It involved massive calculation due to the extraction of different sized features of about thousands of proposals in every picture.

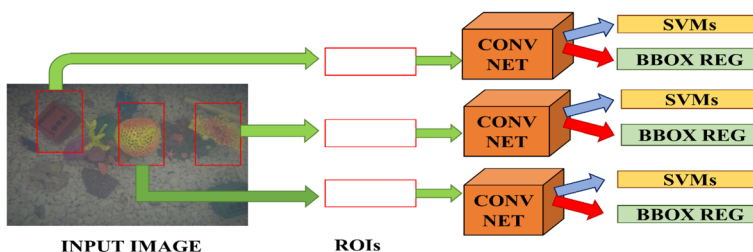


Fig. 18 Process flow of RCNN

To reduce the computational time of RCNN, a CNN model is made to run just once per picture instead of making it run 2000 times per picture. All the ROIs (portions in an image that contain some underwater objects) are then obtained.

The author Ross Girshick of the RCNN framework presented the idea of making the CNN model run just once per picture and then looking for a way out for sharing that computation across the 2000 regions. The input picture is fed to the CNN model in the Fast RCNN framework. The CNN model in turn produces the convolutional feature maps which are used to extract region proposals. An ROI pooling layer is then used for reshaping each region of the proposal into a fixed size to feed it into FCN (a fully connected network).

In the previous two approaches, a picture is taken as input. This picture is then given to a CNN framework which outputs the Regions of Interest and a layer of ROI pooling is applied on each of these regions for reshaping them according to the input of CNN. After these steps, every region is then given to FCN (a fully connected network). A layer called SoftMax is employed on top of FCN (a fully connected network) to generate classes. The Linear Regression layer is employed along with the SoftMax layer to generate coordinates of the bounding box for the classes that are predicted.

Fast RCNN employs only one model instead of employing three different frameworks as in RCNN. The single model used in Fast RCNN performs feature extraction from the regions, splits them into various classes, and simultaneously generates the bounding boxes for the classes that are identified.

A Fast RCNN model uses the whole image and set of proposals of the object as input. Similar to Region Proposal Network (RPN), the model initially processes the entire image using the VGG16 network which is the base convolutional network and generates a 512-dimensional feature map. After that from feature maps, every proposal of the object is mapped to ROI. Then max-pooling utilized by a layer called ROI pooling converts the ROI features into an invariant 7×7 spatial extent. A series of fully connected layers is then given the feature vector of 7×7 . This sequence of FCN then finally branches into a pair of sibling output layers, namely, SoftMax layer and bounding box regression. SoftMax showing the estimates of SoftMax probability over a background class and K object classes. The bounding box regression shows the boundary box coordinates for the K object classes. For bounding box regression and classification training, multi-task loss L can be shown as follows:

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v) \quad (10)$$

where, p is equal to p_0, p_1, p_k represents probability distribution that is discrete over $K + 1$ outputs, t^u represents the predicted bounding box coordinate, u denoted the true objects' class number, v denotes the bounding box of ground truth, L_{cls} represents the log loss for true class u , L_{loc} represents the loss in boundary box regression, λ denotes the hyperparameter that governs the balance in the two functions. This is the thought of classification and bounding box regression.

A detailed understanding of the working of Fast RCNN can be obtained from the visual example which is explained with the help of Fig. 19.

Firstly, an underwater image is taken as input by the Fast RCNN model. This image is then given to the CNN model which in turn generates Regions of Interest. These ROIs are then given to the ROI pooling layer to ensure that all the regions are of equal sizes. Lastly, for classification and for returning bounding boxes by employing SoftMax and Linear Regression layers at the same time, these regions are given to Fully Connected Network. In this way Fast

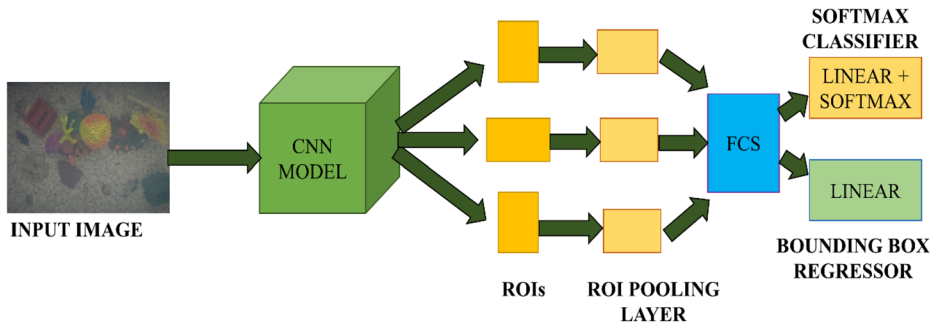


Fig. 19 Process flow of fast RCNN

RCNN solves two big issues such as instead of passing about 2000 regions per picture, only one region is passed to CNN, and employing a single model instead of employing three different models to extract features, it performs classification and generates bounding boxes.

Disadvantages of fast RCNN

Fast RCNN also employs the concept of selective search to search for the ROIs which consumes a lot of time and hence the network becomes slow. Fast RCNN consumes almost 2 s per picture for detecting objects. This much amount of time is better as compared to the RCNN model, however, if massive real-life datasets are considered, then not even a Fast RCNN works that fast anymore.

There is yet another algorithm for object detection that is more efficient than Fast RCNN and it is popularly known as Faster RCNN.

3. Faster RCNN

Faster RCNN has acquired great accomplishments lately in object detection. Ren et al. [95] presented a Faster Region-Based Convolution Neural Network, utilizing Region Proposal Network (RPN) to generate region proposal and after that, for bounding box regression and classification it uses a CNN framework. As compared to the conventional approaches, algorithms based on deep learning are more robust to changing environments such as variations in illumination, perspective distortion, and motion blur. From among the latest algorithms based on deep learning, the performance of Faster RCNN has been excellent in a lot of areas. Sa et al. [100] employed the Faster RCNN model with transfer learning to perform object detection with better performance. Hoang et al. [49] presented a Faster RCNN technique to autonomously perform detection of whether the driver uses a Mobile Phone or he is holding a steering wheel. Zhang et al. [135] proffered a method of detection of a pedestrian by employing RPN which is then followed by boosted forests on high-resolution, shared convolutional feature maps. Hai Huang et al. [55] in 2019 proposed an approach that uses the Faster RCNN framework to assess various subaquatic organisms' data augmentation approaches.

The faster RCNN framework has two subnetworks namely, Fast RCNN and the Region Proposal Network (RPN) [38]. Same input feature maps are shared by Fast RCNN and RPN that have been extracted using the base convolutional network. Hai Huang et al. [55] have employed a pre-trained VGG16 framework [106] as the base network. VGG16 is pre-trained on the dataset namely, ImageNet, and RPN is employed to produce proposals. Also, Fast RCNN is employed for the classification of these proposals.

Region proposal network (RPN)

RPN is utilized to produce proposals by using the feature maps that are put in. In [55] feature maps of dimension 512 are drawn out by VGG16 which is the base convolutional network that takes the whole image as an input. The put-in feature maps are utilized as the input of a spatial window that is 3×3 , each sliding window is being mapped to a 512-dimensional feature vector. Double sibling fully connected layers— a box regression layer plus a box classification layer are given these feature vectors as input. A New concept of Anchors was proffered in [95]. The center of an anchor is at the sliding window which is 3×3 , every sliding window consists of nine anchors in combination with three aspect ratios [1: 1, 1: 2, 2: 1] and three scales [1282, 2562, 5122]. The layer called Box Classification examines the positivity of an anchor if it is positive or not. Also, the Box Regression layer generates the bounding box coordinates. To train the RPN framework, a loss function can be given as follow:

$$L = \frac{1}{N_{cls}} \sum L_{cls}(p, p^*) + \lambda \frac{1}{N_{reg}} \sum p^* L_{reg}(t, t^*) \tag{11}$$

where p denotes the probability that a proposal is an object, p^* represents the proposal true label (if a proposal being an object, then, $p^* = 1$, if not the, $p^* = 0$), t and t^* denote the predicted and ground truth boundary box coordinates, respectively. N_{reg} and N_{cls} denote the two normalization parameters, L_{cls} represents the loss in classification, and over two classes it represents log loss i.e., object versus not object. L_{reg} denotes the regression loss, the product p^*L_{reg} implies that only for those anchors that are positive, the regression loss is turned on.

$$t_x = \frac{(x-x_a)}{w_a} \quad t_y = \frac{(y-y_a)}{h_a} \tag{12}$$

$$t_w = \log\left(\frac{w}{w_a}\right) \quad t_h = \log\left(\frac{h}{h_a}\right) \tag{13}$$

$$t_x^* = \left(\frac{x^*-x_a}{w_a}\right) \quad t_y^* = \left(\frac{y^*-y_a}{h_a}\right) \tag{14}$$

$$t_w^* = \log\left(\frac{w^*}{w_a}\right) \quad t_h^* = \log\left(\frac{h^*}{h_a}\right) \tag{15}$$

The parameterization of four coordinates for bounding box regression is given in Eq. 15, x and y in the above equations represent the box center coordinates, h and w denote the height and width, respectively, of the bounding box, x^* , x_a , and x are for the ground-truth bounding box, anchor box, and predicted bounding box, respectively. Similarly, for y , w , and h . A Fast RCNN model [38] is employed to classify the proposals of the object which have been detected by an RPN.

Detailed understanding of working of faster RCNN

The advanced version of the Fast RCNN model is the Faster RCNN model. The main difference in both the networks is that the Fast RCNN network employs selective search for producing ROIs and on the other hand Faster RCNN network employs Region Proposal Network (RPN). RPN uses feature maps of pictures as input and produces a set of proposals of an object, each having score of objectness as an output.

The steps given below are followed in the Faster RCNN model:

1. The network first takes a picture as an input and then passes it to the CNN model. CNN returns the picture feature map.
2. These feature maps are given to RPN which returns the proposals of an object with their score of objectness.
3. These proposals are given to an ROI pooling layer so that all the object proposals have the same size.
4. Lastly, a fully connected layer having on top of it SoftMax layer and linear regression layer is applied on the proposals for classification and for generating the objects' bounding boxes.

The flow of the whole process of Faster RCNN is indicated in Fig. 20 given below.

Figure 21 given below shows how an RPN network works. The feature maps are taken by the Faster RCNN model from the CNN model and then gives to the RPN i.e., Region Proposal Network. On these feature maps, RPN then employs a sliding window and at every window, RPN produces a k number of Anchor boxes having varied sizes and shapes.

Fixed-sized bounding boxes namely, anchor boxes are positioned in the image. These boxes have various sizes and shapes. Two things are predicted by RPN for every anchor box:

- Firstly, it predicts the probability of an anchor being an object and does not predict the class of the predicted object.
- Secondly, it checks if the boundary box regressor to adjust the anchor boxes better fits the predicted object or not.

After defining boundary boxes of various sizes and shapes, the ROI pooling layer is applied to them. There are then object proposals without any class assigned to these proposals. Every

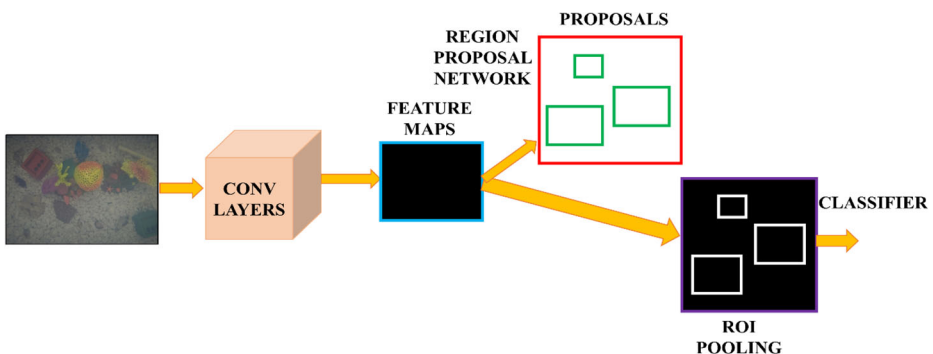


Fig. 20 Process flow of faster RCNN

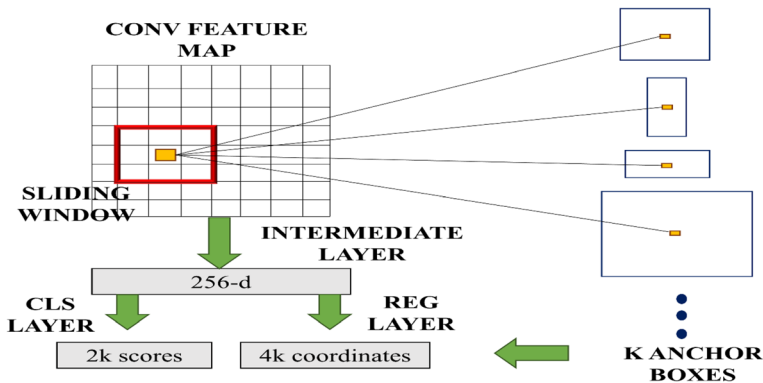


Fig. 21 Working of RPN

proposal is cropped so that each of them consists of an object and that is the basic function of the ROI pooling layer. This layer draws out feature maps having been fixed for every anchor. Feature maps are then given to a fully connected layer having SoftMax layer and linear regression layer. Lastly, these layers classify the objects and also predict the boundary boxes for the objects that are identified.

Disadvantages of faster RCNN

The aforementioned object detection techniques used for underwater object detection utilize regions for the identification of the objects. These above-mentioned algorithms do not see the whole picture in one go, rather look at the portions of the picture sequentially. This gives rise to two problems:

- The technique needs several passes through one image for all objects to be extracted.
- Since in these algorithms various systems are operating back-to-back, the performance of these networks further ahead relies on the performance of previous systems.

4.1.3 Networks based on YOLO (You Only Look Once) framework

The algorithms based on RCNN primarily employ regions for object localization within the picture. These frameworks do not look at the whole picture, rather look only at the portions of the pictures that have a greater possibility of having an object. On the other hand, the YOLO model uses the complete image in one instance and performs the prediction of the boundary box coordinates and for these bounding boxes, it also predicts the class probabilities. The major benefit of employing the YOLO framework is its extremely high speed.

Sung et al., [109] in 2017 presented an underwater fish detection plus fish classification technique employing the YOLO model. These researchers trained the framework of YOLO by using some 829 pictures and acquired classification precision of about 93% on 100 pictures of fish species. Xu and Matzner [126] in 2018 applied the YOLO model for fish detection in three varied datasets and obtained 53.92% of mean average precision. Liu et al., [77] 2018 presented YOLO along with a parallel correlation filter for detection and sporing of subaquatic fish.

They proffered results of simulation on two datasets of undersea fish and also presented the fish tracker effectiveness. YOLO framework has four popular versions, i.e., yolov1, yolov2, yolov3, and yolov4. Each version of YOLO has a gradual improvisation over the earlier version.

1. YOLO-V1

This algorithm is unbelievably fast and can operate on 45 frames/s [94]. A generalized representation of objects in an image can also be understood by the YOLO framework. Therefore, this algorithm is quite trending nowadays and shows a comparatively best performance. The steps followed by the YOLO model to detect an object in an image are briefly discussed below:

1. The algorithm first takes an input image.
2. It then divides an input image into grid cells say 3×3 as shown in the Fig. 22 given below.
3. Then it applies image classification and localization on each grid cell. YOLO performs the prediction of the boundary boxes and the associated probabilities for the class of an object.

Labeled data is passed to the YOLO model to perform its training. For example, the underwater image is divided into a 3×3 grid cell, and if only three classes are defined in which the underwater objects need to be categorized into then for each grid cell, y label is going to be an 8-dimensional vector as shown below in Fig. 22.

Here, p_c is used to indicate the probability of the presence of an object that is whether an object is in the grid cell or not, b_x , b_y , b_h , b_w are used to specify the boundary box only if an object is present in the grid cell, c_1 , c_2 , c_3 define the classes of objects.

In total, the YOLO-v1 consists of 26 layers having 24 layers for convolution operation that is followed by two Fully Connected layers. The major disadvantage of YOLO-v1 is that it is not able to detect tiny objects. This version reframes object detection as just a single regression problem from picture elements to coordinates of bounding box and class probabilities. This framework is quite simple and trains on entire images. This model has many advantages over conventional approaches for object detection. The separate object detection components are unified into one neural network. To predict the bounding box YOLO-v1 utilizes features from the whole image for the prediction of each boundary box. Simultaneously, across all classes, it

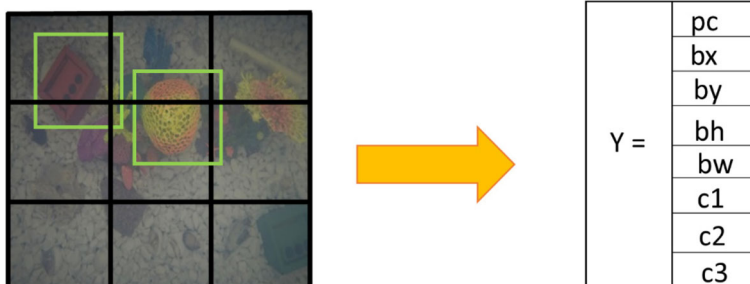


Fig. 22 YOLO object detection

also estimates all boundary boxes for an image. The YOLO model proffers end-to-end training with a speed that is real-time while ensuring high mean precision [94]. Given below is an equation that indicates the total number of detections per image.

$$\text{Total detections to be done per image} = s \times s((B*5) + C) \quad (16)$$

Here, $S \times S$ represents the total number of divisions YOLO makes in an image, B denotes the number of detected bounding boxes all over the complete image, and for every box 5 elements are calculated, namely, coordinates of detected object's center (x,y), height, width, and confidence score. C represents the conditional probability for several classes.

2. YOLO-V2 / YOLO9000

YOLO version one has a lot of disadvantages as compared to state-of-the-art object detection models [92]. As compared to Fast RCNN, YOLO-v1 error analysis indicates that it presents a substantial number of inaccuracies in localization. Besides, this model has comparatively low recall compared to methods based on region proposal. Therefore, this version of YOLO [92] focuses mainly on the improvisation of localization and recall while also maintaining precision in object classification. Generally, computer vision advances towards deeper and bigger networks [46, 106, 112]. Better work usually results from training bigger networks or combining various models. Yolo-v2 however, offers more precise detection that is also fast. The network is simplified and representations are made easier to learn instead of scaling up the network. Various ideas from earlier work have been implemented to improve the performance of the YOLO model.

Batch normalization The concept of Batch normalization results in substantial convergence improvements while reducing the requirement of other regularization forms [56]. On all convolutional layers in YOLO batch normalization is used which proffers greater than 2% mAP improvement. The framework is also regularized using Batch normalization and with batch normalization, a dropout from the framework can be removed without overfitting.

High-resolution classifier Many state-of-the-art object detection approaches employ classifiers that are pre-trained on ImageNet [99]. In AlexNet mostly classifiers process input pictures that are less than 256×256 in size [61]. Version one of YOLO performs the training of the classifier network on an image size of 224×224 and expands the resolution of images to 448 for object detection. This implies the framework has to switch to learning detection and also adjust to the latest input resolution at the same time. In YOLO-v2 initially, the classification framework is fine-tuned at the resolution of 448×448 for about 10 epochs on dataset ImageNet. This gives the framework some amount of time for adjusting the network filters to perform better in case of higher resolution input. This network for classification with high resolution proffers an increment of about 4% in mAP.

Dimension clusters Two problems are experienced with anchor boxes if employing them with YOLO-v1. The first issue is the dimensions of the anchor box are hand-picked. The framework learns to adjust the anchor boxes properly but when better network priors are picked to begin with, it becomes easier for the framework to learn to estimate better detections. K-means clustering is used for the training set boundary boxes to automatically search for

better priors instead of selecting priors manually. If standard k-means along with Euclidean distance are employed, bigger boxes produce more inaccuracy than smaller boxes. However, priors that result in better Intersection over Union (IOU) scores that are not dependent on the box size are required. A Brief idea of the architecture of YOLO-v2 is given below in steps:

- After every convolutional layer, there is the addition of Batch Normalization layers.
- It consists of 30 layers in contrast to the YOLO-v1 model which has 26 layers.
- Anchor Boxes are introduced.
- There is no fully connected layer in the model.
- Still bad with small objects

Some updates are presented to previous versions of YOLO by making some design modifications. The modified architecture is a little larger than earlier YOLO versions but has more accuracy [93].

Darknet-19 which is a new classification model is used as a base model of YOLO-V2. Same as the VGG network, YOLO-V2 usually employs filters of size 3×3 and twice the number of channels succeeding each pooling step [106]. The final framework, namely, Darknet-19, consists of 19 Conv. and 5 max-pooling layers. A complete description of this framework is given in Table 2. The Darknet-19 framework needs only 5.58 billion number of operations to perform processing of a picture yet acquires 72.9% top-1 precision and 91.2% top-5 precision on ImageNet.

3. YOLO-V3

Yolo-v3 is comparatively fast than the previous versions. Using 320×320 images, this version shows a time consumption of about 22 ms with 28.2 mAP, which is as error-free as Single Shot Detector (SSD) but is three times faster than SSD [93]. Similar to YOLO-v2 or YOLO9000, this version also performs the prediction of bounding boxes employing dimension clusters as anchor boxes [112]. An objectness score is predicted by YOLO-v3 for every bounding box by employing logistic regression. When the boundary box prior is overlapping a ground-truth object, the objectness score must be 1. The prediction is ignored when the boundary box prior is not being the best but overlaps an object that is the ground-truth object by an amount greater than some threshold [95]. Every box estimates the classes contained by the boundary box employing multi-label classification. A layer of SoftMax is not used, as it is unimportant for efficient performance, however, independent logistic classifiers are simply used. At the time of training, binary cross-entropy loss is employed for the predictions of classes. This kind of formulation aids if more complex domains are considered such as the Open Images Dataset [60]. Such kind of dataset has several overlapping labels e.g., person and woman. Employing a layer of SoftMax uses the assumption every box consists of exactly a single class which usually is not the case. Hence, a multi-label method models the data in a better way.

Version-3 of YOLO makes predictions of boxes at three different scales. The system takes out features of those scales employing the same concept as that of feature pyramid networks [71]. Multiple convolutional layers are added and the last layer of these layers performs prediction of 3-dimensional tensor encoding boundary box, class predictions, and objectness. For the determination of priors for boundary box, k-means clustering is still employed. Yolo-v3 model is a hybrid algorithm of the model employed in YOLO-v2 (Darknet-19) and the

Table 2 Darknet-19

Type	Filters	Size/Stride	Output
Convolution	32	3×3	224×224
Maxpool		2×2/2	112×112
Convolution	64	3×3	112×112
Maxpool		2×2/2	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		2×2/2	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		2×2/2	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		2×2/2	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
AVGPOOL		Global	1000
SOFTMAX			

new-fangled residual network. This version of YOLO uses 3×3 and 1×1 CNN layers in succession but it also involves some shortcut connections and is substantially bigger. It consists of 53 CNN layers; therefore, it is called Darknet-53. Figure 23 shows the Darknet-53 algorithm.

4. YOLO-V4

The main aim of YOLO-v4 is the fast speed for performing operations of a neural network and optimization for parallel computations. Yolo-v4 proffers two options of neural networks that are real-time [11]:

- For GPU (Graphics Processing Unit), a small number of groups (1 to 8) in Conv. layers are employed: CSPResNeXt50 / CSPDarknet53
- For VPU (Vision Processing Unit), grouped-convolution is used, but YOLO-v4 refrains from employing Squeeze-and-excitement (SE) blocks - particularly this includes the following frameworks: EfficientNet-lite / MixNet [115] / GhostNet [43] / MobileNetV3

The objective of this version of YOLO is to look among put in network resolution for the optimal balance, the Conv. layer number, the parameter number, and the number of the output layer. For example, an immense survey reveals that the CSPResNext50 is significantly better than CSPDarknet53 in the case of object classification when used on the ILSVRC2012 (ImageNet) dataset [24]. Conversely, the CSPDarknet53 framework is better than

	TYPE	FILTERS	SIZE	OUTPUT
	Convolutional	32	3×3	256×256
	Convolutional	64	3×3 / 2	128×128
1×	Convolutional	32	1×1	
	Convolutional	64	3×3	
	Residual			128×128
	Convolutional	128	3×3 / 2	64×64
2×	Convolutional	64	1×1	
	Convolutional	128	3×3	
	Residual			64×64
	Convolutional	256	3×3 / 2	32×32
8×	Convolutional	128	1×1	
	Convolutional	256	3×3	
	Residual			32×32
	Convolutional	512	3×3 / 2	16×16
8×	Convolutional	256	1×1	
	Convolutional	512	3×3	
	Residual			16×16
	Convolutional	1024	3×3 / 2	8×8
4×	Convolutional	512	1×1	
	Convolutional	1024	3×3	
	Residual			8×8
	AvgPool		Global	
	Connected		1000	
	Softmax			

Fig. 23 Darknet-53

CSPResNext50 in the case of object detection when used on the MS COCO dataset [69]. The other objective is to choose extra blocks to increase the receptive field and the finest technique of parameter aggregation from various backbone levels for various detection levels: for example, FPN (Feature Pyramid Network), ASFF (Adaptively Spatial Feature Fusion), BiFPN (Bi-directional Feature Pyramid Network).

In YOLO-v4, the Spatial Pyramid Pooling (SPP) layer on top of the CSPDarknet53 network is added, since it considerably enlarges the receptive field, takes out the most substantial context features causing almost no decrement in the speed of network operation [11]. PANet is used as the technique of parameter aggregation from various backbone levels for various object detector levels, rather than using FPN as in YOLO-v3. Lastly, as an architecture of YOLO-v4, CSPDarknet53 is chosen as a backbone, SPP as an extra module, PANet as the path-aggregation neck, and YOLO-v3 anchor-based head. CGBN-Cross-GPU Batch Normalization or costly specialized devices are not used which allows anybody to reproduce the state-of-the-art results on a traditional graphic processor like RTX 2080Ti or GTX 1080Ti. Figure 24 given below gives an overview of the architecture of YOLO-v4.

Limitations of YOLO

Strong spatial constraints are imposed by the YOLO model on boundary box predictions as every grid cell just performs a prediction of two boxes and it can only have a single class. This kind of spatial constraint restricts the number of close objects that the YOLO framework can predict. YOLO model has to struggle with tiny and small objects that seem to be in groups, like flocks of birds. As this model trains to perform prediction of boundary boxes from data, the model struggles to generalize to the objects in novel or unusual configurations or aspect ratios. YOLO framework also employs relatively rough features to predict bounding boxes as this architecture has many layers of down-sampling from the input picture. Lastly, as the YOLO model is trained on a loss function that approximates the performance of object detection, the

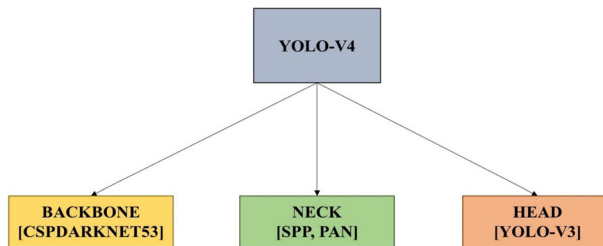


Fig. 24 YOLO-v4 architecture

loss function of the YOLO model treats inaccuracies equally in small boundary boxes vs large boundary boxes. A small inaccuracy in a big box is generally not serious however a small inaccuracy in a smaller box has a much higher influence on Intersection over Union (IOU). The major source of inaccuracy is inaccurate localizations.

4.2 Some other underwater object recognition algorithms

Krizhevsky et al., in 2012 [61] proposed the AlexNet model which consists of 5 Conv. layers, that are stacked by nonlinear and/or pooling layers followed by 3 layers that are fully connected. Employing creative methods which also includes DA (data augmentation) [46, 106], dropout [9], local response normalization (LRN) and overlapping pooling [61], rectified linear units (ReLUs) [85, 124], the AlexNet algorithm could win the championship having the testing inaccuracy of 15.4% and outstripped the second winner having the testing inaccuracy of 26.2%. ZFNet which is a variant of the algorithm namely, AlexNet was presented by Zeiler and Fergus in 2014 [133]. The major differences in the AlexNet and ZFNet are in first- layer and second-layer visualization: (a) The size of the filter of a first layer was diminished to 7×7 from 11×11 , and (b) Also, the convolutional stride was diminished to 2 from 4, thereby having immense information in these two layers. The process of visualization has been facilitated by an operation called deconvolution using a deconvolution layer [134]. Alberto et al. in 2017 [4] proffered a Visual Geometry Group Network (VGGNet) which is a type of CNN model and has been build-up by the VGG (Visual Geometry Group), Oxford University. As different deep CNNs' variants [106], the typical frameworks are VGG-16 and VGG-19. GoogLeNet is another framework proffered by Szegedy et al. in 2015 [111] and it is based on the idea of Network in Network (NiN), whereby almost 22 layers present in the framework called inception framework are employed with the learned parameters. GoogLeNet framework aims to decrease the number of feature filters required at every layer, thereby diminishing complexity in computation. He et al. in 2015 [45] presented a Residual Neural Network (ResNet) having a great depth of 152 layers, and also won the competition called ILSVRC-2015 having the testing inaccuracy of about 3.57%. In this framework, residual blocks take advantage of residual learning, and the output comprises \mathbf{X} which is the original input and the output of the final layer that is (\mathbf{X}) .

Unsupervised models based on deep learning include Deep Belief Networks (DBNs) and different assembled variants of Auto Encoder (AE) which includes stacked sparse AE (SAE) [80] denoising AE (DAE) [120], and contractive AE (CAE) [96]. Generally, in algorithms that are unsupervised learning algorithms, parameters are optimized via a greedy training approach that is performed layer-wise and is divided into 2 phases, namely, pre-training and fine-tuning.

Besides Chen et al. [15] in 2020 proposed the Sample-Weighted Hyper Network (SWIPENet). The architecture of the algorithm includes many semantic rich and high-resolution hyper feature maps that are inspired by the algorithm, namely Deconvolutional Single Shot Detector (DSSD) [34]. A fast down-sampling network for detection called SSD that has several up-sampling deconv layers is augmented by DSSD to enhance the feature map resolution. The DSSD network, first of all, constructs several down-sampling conv layers for extracting semantic-rich feature maps that are beneficial for classifying underwater objects. After performing many down-sampling functions, the extracted feature maps are quite rough to proffer adequate information to locate small objects accurately, thus, several up-sampling deconv layers along with skip connection are employed to restore the high-resolution feature maps. Nevertheless, the entire details of the information lost during down-sampling are not completely restored even if the resolution has been improved and recovered. For improving the DSSD algorithm, Chen et al. [15] employ dilated conv layers [12, 130] to acquire strong semantics and that too without any loss of intricate information which is important for object localization.

Figure 25 shown above presents the outline of the SWIPENet algorithm. It has several conv, dilated conv, deconv blocks, and a new sample-weighted loss. The initial layers of this network are based on the structure of the VGG16 network. Unlike DSSD, four dilated conv layers have been added to the model with ReLU activations. This modification can acquire huge receptive fields with no sacrifice on the feature map resolution (huge receptive fields result in stronger semantics). Furthermore, feature maps are up-sampled by using deconv layer and then using skip connection for passing the fine details present in low layers to details present in high layers. Lastly, several hyper feature maps are constructed by the deconv layers. The prediction of the proposed SWIPENet algorithm deploys varied three deconv layers and these are Deconv1 2, Deconv2 2, and Deconv3 2 (represented as Deconvx 2 as indicated in Fig. 25), whose size progressively increase and performing prediction of the objects of several scales. At every location of these three deconv layers, six default boxes are defined and also, employ a 3×3 conv kernel to generate $C + 1$ class scores where C denotes the total number of the classes of object and 1 denotes the background class, and four coordinate offsets that are relative to the original shape of the default box.

5 Experimental results

The comparison of YOLO-v1 with some other real-time object detection algorithms using PASCAL VOC 2007 is performed. To get the idea of the differences between RCNN and YOLO-v1 variants, the inaccuracies on VOC 2007 made by YOLO-v1 and Fast RCNN (one of the highest performing versions of RCNN [63]) are explored. Keeping in view the different profiles of inaccuracies, it is inferred that YOLO-v1 can be employed to rescore detections made by Fast RCNN and decrease the inaccuracies from background false positives, providing a substantial boost to performance. Comparison of YOLO-v1 is made to Fast RCNN as indicated in Fig. 26 since Fast RCNN is among the excellent performing object detectors when used for PASCAL and the detections of this network are publicly available. Hoiem et al. [50] methodology and tools have been used. For every category during test time, N number of top predictions are looked at for the category. Every prediction can be either correct or is categorized based on the type of error:

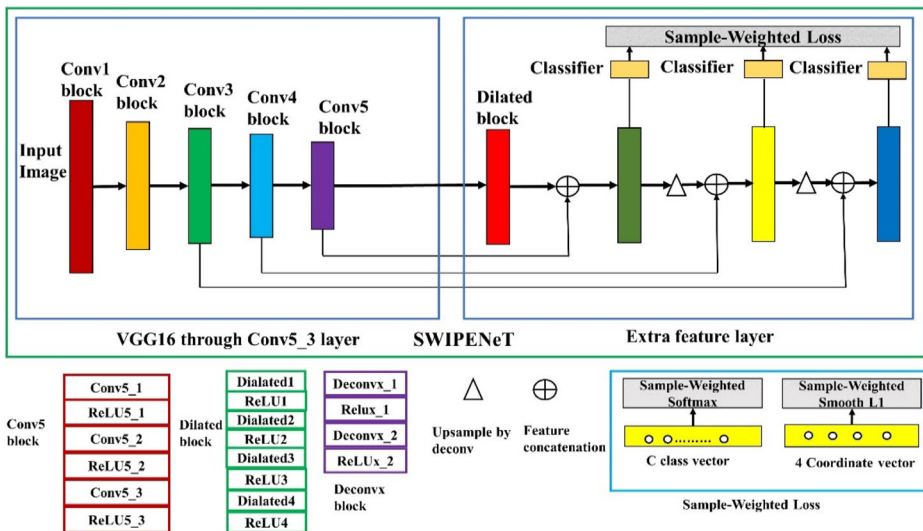


Fig. 25 Overview of SWIPENet

- Correct implies correct class with Intersection Over Union (IOU) > 0.5
- Localization implying correct class with 0.1 < IOU < 0.5
- Similar implies class is similar with IOU > 0.1
- Other implies class is wrong having IOU > 0.1
- Background implies IOU < 0.1 for any object

On comparing RCNN based algorithms and the YOLO model using PASCAL VOC 2007, it can be clearly seen that the RCNN model does not proffer real-time processing and also takes a substantial precision hit from not consisting of good proposals. Fast RCNN increases the speed of the RCNN stage used for classification however, it still depends on a selective search that can take about 2 s per picture to produce boundary box proposals. Therefore, it presents high Mean Average Precision (mAP) but at about 0.5 Frames Per Second (fps) which is still quite far from real-time operational speed. The novel Faster RCNN has replaced selective search with a neural network that proposes boundary boxes, same as Erhan et al. [32]. The version VGG-16 of Faster RCNN has higher mAP, however, being 6 times slower than the YOLO model. YOLO-V2 has also been trained for object detection on VOC 2007.

Figure 27 shown below highlights the comparison of the performance of some detection frameworks on PASCAL VOC 2007. It shows results for mAP and FPS of YOLO-V2 and some other state-of-the-art object detection algorithms. Version 2 of the YOLO framework achieves about 73.4 mAP with a speed far greater than other competing approaches.

Also, for comparing the performance of YOLO-v3 and YOLO-v4, the average precision and Frames Per Second (FPS) of the two algorithms are highlighted and compared using the COCO dataset as shown in Fig. 28 [11].

For YOLO-v4, in the experiments carried out on the COCO dataset for object detection, the hyperparameters that are default have been set as follows: the step decay rate of learning scheduling strategy is employed with starting rate of learning as 0.01 and then multiplied with 0.1 at the 400,000th and the 450,000th steps, respectively; also, the steps of training are

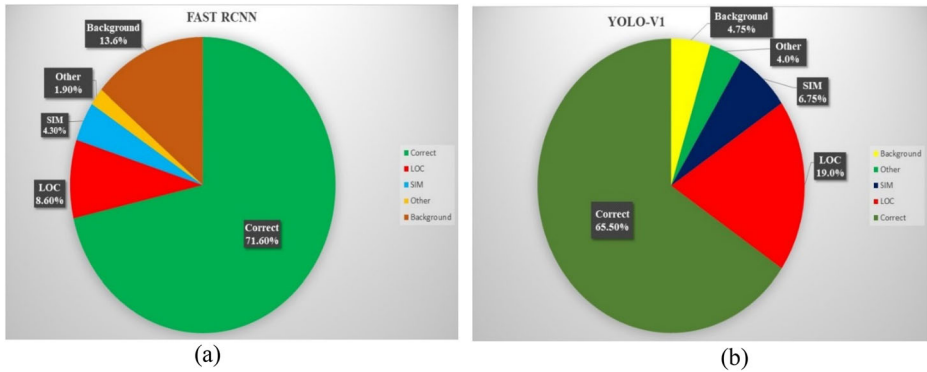


Fig. 26 Error analysis: **a** Fast R-CNN vs. **b** YOLO. These Pie-charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category)

500,500. The graph given above indicates that YOLO-V4 enhances the Average Precision (AP) and Frames Per Second (FPS) of YOLO-V3 by about 10% and 12%, respectively.

In the process of object detection, Mean Average Precision, Precision, and Recall are usually used for assessing the accuracy and the definition is.

- i) Mean Average Precision (mAP): It compares the detected bounding box and the ground-truth bounding box. Then it outputs a score. The greater the score, the greater is the accuracy of the model in the detections performed.
- ii) Precision: It is defined as the ratio of the number of correctly classified positive samples to the total number of samples that are classified as positive (incorrectly or correctly). This parameter gauges the accuracy of the model to classify a sample as positive.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{17}$$

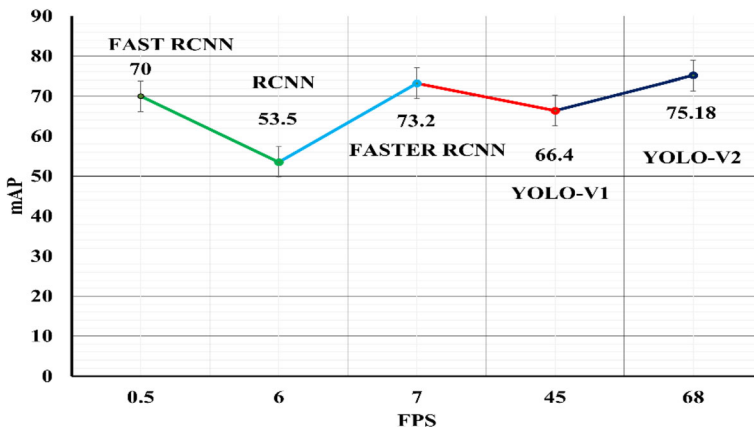


Fig. 27 Comparing speed and mean average precision of RCNN, FAST RCNN, FASTER RCNN, YOLO-V1 and YOLO-V2 algorithms on PASCAL VOC 2007

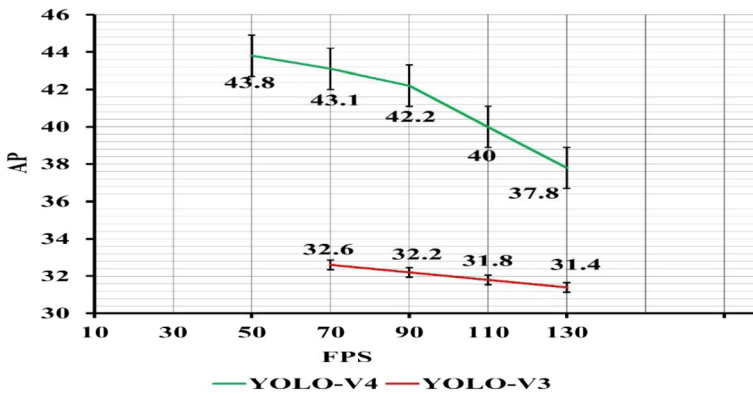


Fig. 28 Comparison of YOLO-V3 and YOLO-V4

iii) Recall: It is defined as the ratio of the number of correctly classified positive samples as positive to the total number of positive samples. This parameter gauges the ability of a model so that it can detect positive samples. The greater the recall, the greater number of positive samples are detected.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{18}$$

To evaluate the accuracy of Faster-RCNN and Yolo-v3 mean Average Precision (mAP), recall rate (Recall), and speed (Frames Per Second - Fps) have been used. The results have been indicated in Table 3. The dataset used comes from the official website, namely, UNDERWATER ROBOT PICKING CONTEST (URPC) and this dataset has been changed into the VOC2007 format. For making it usable, its size was changed to 18,982, out of which 4746 pictures were employed as a testing set and 14,236 pictures have been employed as training set and validation set.

Table 4 given below shows the Mean Average Precision (mAP) and precision of different iteration times by Fast RCNN, Faster RCNN, and Yolo-v3 with IoU =0.7. The GPU employed for the process to compute these parameters is NVIDIA GTX 1080ti and the total images are 30,000, which have been artificially labeled one by one, out of those images 8520 are taken for training, 8530 images used as a validation set, and 12,950 images used as a test set. The images that are used to testify the algorithms are taken from the ‘‘Underwater Robot Picking Contest’’.

Besides, the basic Convolutional Neural Network is being compared to the renowned RCNN family consisting of RCNN, FAST RCNN, and FASTER RCNN algorithms. The comparison is being performed based on time consumption. Table 5 highlights the limitations

Table 3 Test results for faster RCNN and Yolo-v3

Technique	mAP (%)	Recall (%)	FPS
Faster RCNN	69.7	75.6	8
Yolo-v3	76.1	89.5	20

Table 4 Comparison of underwater object detection algorithms based on precision and mean average precision

Iteration	Faster RCNN						Yolo-v3					
	Fast RCNN			Faster RCNN			mAP (%)			Precision (%)		
	mAP (%)	Precision (%)	Scallop	Sea Urchin	Sea Cucumber	Scallop	mAP (%)	Precision (%)	Scallop	Sea Urchin	Sea Cucumber	Precision (%)
2000	27.26	30.13	24.87	26.79	27.53	30.18	27.29	25.13	35.43	37.14	35.42	33.74
4000	37.56	40.51	33.93	38.23	38.74	40.80	39.35	36.06	45.90	48.12	45.50	44.08
6000	41.83	44.45	39.67	41.36	43.15	45.30	42.80	41.35	49.61	51.81	49.87	47.16
8000	45.37	48.67	41.59	45.85	46.59	48.35	47.35	44.08	52.40	54.56	53.14	49.51
10,000	48.22	51.33	45.50	47.84	50.28	52.09	50.76	48.00	55.89	58.17	56.99	52.50
12,000	50.90	53.75	47.65	51.31	52.53	53.96	53.44	50.20	58.34	59.77	59.48	55.77
14,000	53.09	55.69	49.38	54.20	54.43	56.18	54.78	52.31	60.58	63.39	60.91	57.44
16,000	55.04	58.85	54.92	58.85	57.32	59.66	56.98	55.34	62.02	64.09	62.50	59.47
18,000	56.66	60.49	56.81	58.49	58.62	60.35	59.55	55.95	64.18	66.79	65.15	60.58
20,000	58.63	62.12	55.27	58.49	60.93	62.30	61.86	58.63	66.00	68.87	66.20	62.93
22,000	60.42	63.95	56.67	60.63	63.07	64.33	63.93	60.95	67.22	70.37	68.02	63.26
24,000	61.35	64.57	57.29	62.19	64.37	65.94	64.67	62.51	68.88	71.50	70.54	64.60
26,000	63.40	66.60	59.65	63.94	66.38	68.07	67.17	63.90	70.44	72.85	71.83	66.64
28,000	65.03	68.81	60.92	65.36	68.15	69.98	68.67	65.82	72.00	74.79	73.17	68.03
30,000	66.84	70.09	62.24	68.19	69.42	70.49	70.53	67.24	71.99	74.84	73.44	67.70
32,000	67.68	70.73	63.78	68.53	70.68	72.78	71.47	67.79	72.24	74.47	73.78	68.47
34,000	69.26	72.65	64.11	71.03	71.72	73.96	72.17	69.01	72.15	74.62	74.01	67.81
36,000	70.96	74.75	65.90	72.23	73.75	74.44	75.02	71.79	71.88	74.87	72.82	67.95
38,000	71.25	74.83	66.95	71.98	74.80	76.83	75.53	72.03	71.69	74.05	72.37	68.63
40,000	72.88	76.46	68.08	74.09	75.75	77.53	76.13	73.58	72.70	75.55	73.70	68.83
42,000	73.23	76.30	68.71	74.68	75.99	77.41	76.96	73.59	71.96	75.34	73.21	67.33
44,000	73.13	76.19	68.70	74.49	74.86	77.65	73.26	73.67	71.57	74.83	72.59	67.28
46,000	72.82	76.00	67.53	74.93	74.46	76.19	74.33	72.85	71.87	74.58	73.41	67.61
48,000	73.01	76.46	68.49	74.07	74.62	76.16	74.41	73.30	71.59	74.19	73.39	67.20
50,000	72.84	76.35	67.48	74.69	74.64	76.40	73.26	74.25	71.44	74.32	72.32	67.69

Table 5 Comparison of various underwater object detection algorithms

Algorithm	Features	Prediction time	Limitations
CNN	Splits input picture into several regions and then performs classification of every region into different classes.	Slow	It requires multiple regions to perform prediction precisely and therefore consumes a lot of computational time.
RCNN	Employs selective search to produce regions. About 2000 regions are extracted from each picture.	40–50 s	A lot of computational time is consumed as every region is given to the CNN model individually. It also employs three different networks for doing predictions.
Fast RCNN	Each picture is given just once to the CNN model and it extracts feature maps. On these feature maps, a selective search is employed to output predictions. It combines all of the three models that are employed in RCNN.	2 s	As the selective search method is time-consuming so the time consumption is still more.
Faster RCNN	In this model, RPN replaces the method of selective search. RPN makes the model faster than the rest of the aforementioned models.	0.2 s	RPN also consumes time and multiple systems operate back-to-back. Thus, the system performance relies on the performance of the previous system.

in these algorithms and also the amount of time consumed to perform the operation of object detection.

5.1 Comparison of various object detection algorithms used for underwater images

For underwater object detection, the most frequently employed approaches cannot be applied due to the vision being of low-quality and the objects to be detected being small. Thus, it has been quite challenging for the researchers to search for an efficient underwater object detection algorithm keeping in view the subaquatic conditions. The decision-making for selecting an efficient underwater object detection algorithm highly relies on the performance of the networks employed for the particular task. The performance of the underwater object detection algorithms can be measured using some renowned parameters like Mean Average Precision (mAP). Mean Average Precision is the mean of the precision of the total of the detection classes and it is extensively employed to evaluate the object detection systems. To compare various underwater object detection algorithms, the dataset has been prepared in Pascal VOC form. The comparison of subaquatic detection algorithms such as CNN [42], Fast RCNN [36], Faster RCNN [95], and YOLO-V3 are indicated in Fig. 29 [42].

From the aforementioned comparisons and results, it is inferred that the accuracy in detection using the Faster RCNN model is better as compared to the other techniques, however, the difference being not very great. As compared to the YOLO-V3 model [93], the CNN technique can yield more precise detection. The convergence of the techniques is different; the YOLO-V3 model converges after 28,000 iteration times, which is earlier compared to the Fast RCNN model and Faster RCNN model. All the techniques cannot improvise the accuracy of object detection after 40,000 iteration times, the main cause is a scarcity of the underwater image dataset, and also the pictures of the dataset are alike, particularly the background of the underwater pictures is similar. In subaquatic object detection, this is the major reason contributing to the fact that the subaquatic data in deep water bodies is too hard to acquire.

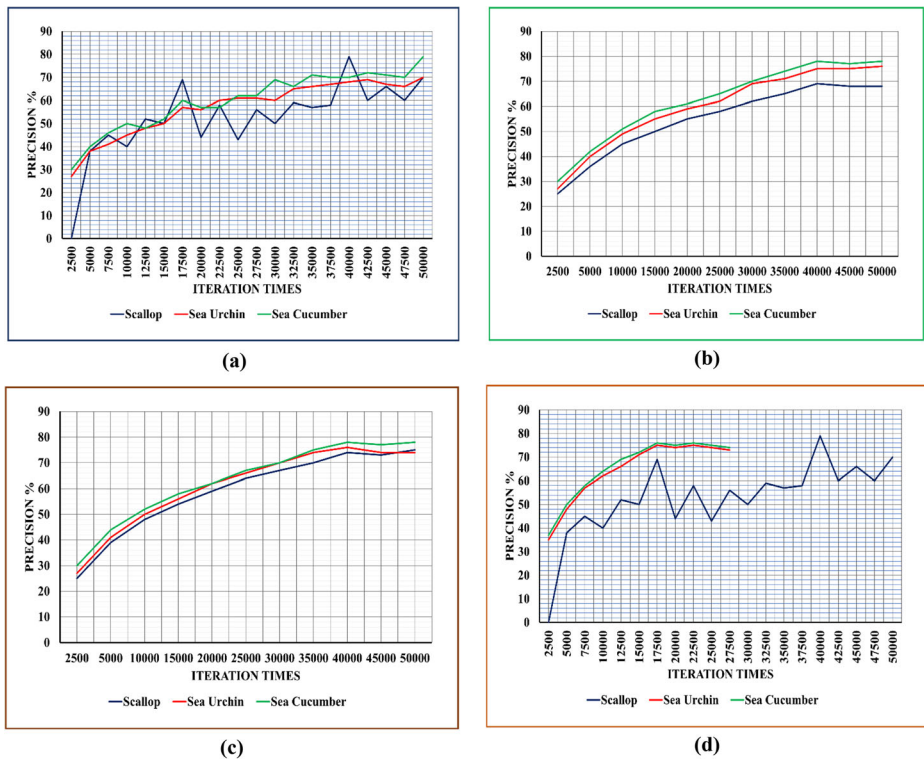


Fig. 29 mAP vs Iteration Times of **a** CNN, **b** FAST RCNN, **c** FASTER RCNN, and **d** YOLO-V3, respectively

The curves of loss function are indicated in Fig. 30. It can be seen that the values of loss for all the techniques are convergent, and the amplitude of loss values for the YOLO-V3 model is smaller as compared to the Fast RCNN model [36] and Faster RCNN model [95].

For underwater object detection, the precision of all of the aforementioned techniques is sufficient for applications, however, real-time underwater object detection is more required. The object detection speed has been indicated in Table 6 shown below. It can be clearly seen the YOLO-V3 model [93] has a very high speed of object detection, about 4 times faster as compared to the Faster RCNN framework [95].

Table 7 given below highlights some existing deep learning-based underwater object detection algorithms.

6 Challenges

The most mandatory issue and highly challenging task is the recognition of optical content for the analysis of underwater pictures. Variability in Intra-class results in the variation of optical content via scales, views, a deformation that is non-rigid, and illumination. Particularly, for performing classification and detection of seagrasses, the differences in boundary in various classes are highly ambiguous as compared to what is for corals or fish. Besides, in digital pictures, there is more ambiguity in optical content as water depth increases. The scarcity of annotated datasets presently at hand to detect objects is another significant hurdle. The object

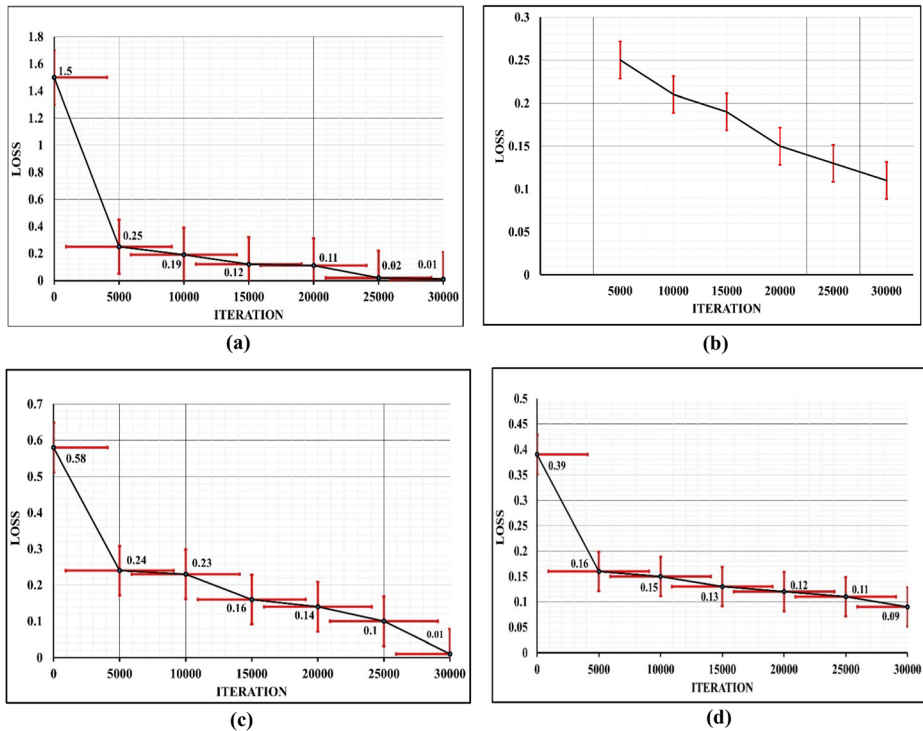


Fig. 30 The loss curves of **a** CNN, **b** Fast RCNN, **c** FASTER RCNN, and **d** YOLO-V3, respectively

detection techniques require not only to precisely localize and classify objects but also require to have faster and real-time predictions.

7 Conclusion and future directions

Underwater object detection based on deep learning has become a research hotspot because of its highly powerful ability to learn and data pre-processing efficiency. A detailed review of underwater object detection algorithms employing the concept of deep learning has been presented and mainly categorized into CNN, RCNN family, and YOLO family. Besides, it would be relevant to mention here that the detection speed of the YOLO family is quite fascinating and suitable for real-time applications. Multiple subproblems in underwater object detection algorithms have also been comprehensively compared and reviewed. Also, deep learning-based underwater object detection algorithms, their architectures, and issues have been summarized. It was important to highlight all the techniques used for the analysis of underwater data to make it easy to concentrate on possible future work built on the approach of

Table 6 Detection speed of various methods (IoU = 0.7, Learning Rate = 0.001)

Approach	Fast RCNN	FASTER RCNN	YOLO-V3
Time Cost (ms)	96	85	20

Table 7 Some famous and existing deep learning-based underwater object detection methods

S. No.	Methods
1.	Fast and accurate fish detection and recognition of underwater images with fast RCNN by Li et al. in 2015
2.	Underwater fish detection plus fish classification technique employing the YOLO model, by Sung et al. in 2017
3.	Fish detection using deep learning by Suxia et al. in 2019
4.	Faster RCNN for marine organism detection and recognition using data augmentation, by Hai Huang et al. in 2019
5.	Sample-Weighted Hyper Network (SWIPENet), by Chen et al. in 2020
6.	Underwater image processing and object detection based on deep CNN method, by Han et al. in 2020

deep neural networks. A lot of work has been done on coral classification and detection employing deep learning, however, meager research work has been carried out for seagrass which is also equally important for the underwater ecosystem. The precision, robustness, and effectiveness of any algorithm meant for detection and classification can be significantly increased when both features based on texture and color are combined. For improved results of seagrass classification and detection, accumulation of features that are hand-crafted features and neural networks are employed. Thus, there exists an opportunity for developing an effective and efficient deep learning method for marine seagrass imagery and that can be the focus of a researcher for future work.

Various problems still prevail despite the development of various algorithms based on deep learning for underwater object detection. Many such challenges and issues have been based on the literature review including the requirement of pre-training, public underwater dataset, recognition of subclasses, the requirement of a unified framework, and the requirement of general network structure. These are the issues that the research community needs to target and solve as it will help to further explore see life in an effective way.

References

1. Abdel-Maksoud E, Elmogy M, Al-Awadi R (2015) Brain tumour segmentation based on a hybrid clustering technique. *Egypt Inform J Cairo Univ* 16(1):71–81
2. About ImageNet (n.d.) <http://image-net.org/about-overview>. Accessed 8/7/21
3. About ImageNet (n.d.) <http://image-net.org/about-overview>. Accessed 8/7/21
4. Alberto GG, Sergio OE, Sergiu O, Victor VM, Jose GR (2017) A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*
5. Amanda D, Felipe C, Joel G, Silvia B (2016) A dataset to evaluate underwater image restoration methods. In: *IEEE OCEANS 2016-Shanghai*, pp. 1–6. <https://doi.org/10.1109/OCEANSAP.2016.7485524>
6. Aquino G, Rubio JDJ, Pacheco J, Gutierrez GJ, Ochoa G, Balcazar R, Cruz DR, Garcia E, Novoa JF, Zacarias A (2020) Novel nonlinear hypothesis for the delta parallel robot modelling. *IEEE Access* 8: 46324–46334. <https://doi.org/10.1109/ACCESS.2020.2979141>
7. BBC News (2016) Artificial intelligence: Google's AlphaGo beats Go master Lee Se-dol
8. Beijbom O, Edmunds PJ, Kline DI, Mitchell BG, Kriegman D (2012) Automated annotation of coral reef survey images. In: *IEEE computer society conference on computer vision and pattern recognition*, IEEE Press: Rhode Island, pp. 1170–1177
9. Bell RM, Koren Y (2007) Lessons from the netflix prize challenge. *ACM Sigkdd Explor Newslett* 9(2): 75–79

10. Biswas S, Wang Y, Cui S (2015) Surgically altered face detection using log-gabor wavelet. In: Proceedings of the 12th international computer conference on wavelet active media technology and information processing (ICCWAMTIP), IEEE, Chengdu, China, pp. 154–157
11. Bochkovskiy A, Wang C-Y, Liao M et al (2020) YOLOv4: Optimal Speed and accuracy of object detection. arXiv:2004.10934v1
12. Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2015) Semantic image segmentation with deep convolutional nets and fully connected CRFs. In: ICLR
13. Chen L-C, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2017) DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans Pattern Anal Mach Intell (TPAMI)* 40(4):834–848
14. Chen Y, Yang T, Zhang X, Meng G, Xiao X, Sun J (2019) DetNAS: backbone search for object detection. In: Advances in neural information processing systems (NeurIPS), pp. 6638–6648
15. Chen L, Liu Z, Tong L, et al. (2020) Underwater object detection using invert multi-class adaboost with deep learning. *IEEE explore*, Auckland University of Technology
16. Chiang H-S, Chen M-Y, Huang Y-J (2019) Wavelet-based EEG processing for epilepsy detection using fuzzy entropy and associative petri net. *IEEE Access* 7:103255–103262. <https://doi.org/10.1109/ACCESS.2019.2929266>
17. Chien C-F, Chen Y-J, Han Y-T, et al. (2018) AI and big data analytics for wafer fab energy saving and chiller optimization to empower intelligent manufacturing. In: Proceedings of 2018 e-Manufacturing & Design Collaboration Symposium (eMDC), IEEE, Hsinchu, Taiwan, pp. 1–4
18. Ciregan D, Meier U, Schmidhuber J (2012) Multi-column deep neural networks for image classification. 2012 IEEE conference on computer vision and pattern recognition, Providence, RI, pp. 3642–3649. <https://doi.org/10.1109/CVPR.2012.6248110>
19. Cui S, Ekwonah O, Wang Y (2018) The analysis of emotions over keystroke dynamics. In: 2018 international conference on information, electronic and communication engineering (IECE2018), DESTech Publications, Beijing, China, pp. 28–29
20. Cui S, Wang Y, Ekwonah O (2019) Keystroke dynamics on user authentication. In: 4th international conference on cybernetics (CYBCONF), Beijing, China, pp. 5–7
21. Cui S, Zhou Y, Wang Y, Zhai L (2019) Fish detection using deep learning. *Appl Comput Intell Soft Comput* 13:Article ID 3738108. <https://doi.org/10.1155/2020/3738108>
22. Dai J, Li Y, He K, Sun J (2016) R-FCN: object detection via region-based fully convolutional networks. In: Advances in neural information processing systems (NIPS), pp. 379–387
23. Deng L, Yu D (2013) Deep learning: methods and applications. *Found Trends Signal Process* 7(3–4):197–387. <https://doi.org/10.1561/20000000039>
24. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L (2009) ImageNet: a large-scale hierarchical image database. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 248–255. <https://doi.org/10.1109/CVPR.2009.5206848>
25. Drews Jr P, Edo N, Moraes F, Botelho S, Campos M (2013) Transmission estimation in underwater single images. In: 2013 IEEE International Conference on Computer Vision Workshops. <https://doi.org/10.1109/ICCVW.2013.113>
26. Drews P Jr, Nascimento ER, Botelho S, Campos M (2016) Underwater depth estimation and image restoration based on single images. *IEEE Comput Graph Appl* 36:24–35. <https://doi.org/10.1109/MCG.2016.26>
27. Du X, Lin T-Y, Jin P, Ghiasi G, Tan M, Cui Y et al (2019) SpineNet: learning scale-permuted backbone for recognition and localization. arXiv preprint arXiv:1912.05027
28. Duan K, Bai S, Xie L, Qi H, Huang Q, Tian Q (2019) CenterNet: Keypoint triplets for object detection. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp. 6569–6578. <https://doi.org/10.1109/ICCV.2019.00667>
29. Ducourmau A, Fablet R (2016) Deep learning for ocean remote sensing: An application of convolutional neural networks for super-resolution on satellite-derived SST data. In: 9th workshop on pattern recognition in remote sensing, pp. 1–6. <https://doi.org/10.1109/PRRS.2016.7867019>
30. Duo Z, Wang W, Wang H (2019) Oceanic mesoscale eddy detection method based on deep learning. *Remote Sens* 11(16):1921
31. Elawady M (2014) SparseM: coral classification using deep convolutional neural networks. M.sc. thesis, Hariot-Watt University
32. Erhan D, Szegedy C, Toshev A, Anguelov D (2014) Scalable object detection using deep neural networks. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE conference, pp. 2155–2162. <https://doi.org/10.1109/CVPR.2014.276>

33. Federico F, Elsa R, Humberto S, Víctor P (2020) CNN based detectors on planetary environments: a performance evaluation. *Front Neurorobot* 14:590371(ISSN 1662-5218. <https://doi.org/10.3389/fnbot.2020.590371>
34. Fu CY, Liu W, Ranga A, Tyagi A, Berg AC (2017) Dssd: Deconvolutional single shot detector. arXiv preprint arXiv:1701.06659
35. Garcia J, Fernandez J, Sanz P, Marin R (2010) Increasing autonomy within underwater intervention scenarios: the user interface approach. In: *Proceedings of the IEEE systems conference*, pp. 71–75
36. Ghani ASA, Isa NAM (2015) Enhancement of low-quality underwater image through integrated global and local contrast correction. *Appl Soft Comput* 37(C):332–344. <https://doi.org/10.1016/j.asoc.2015.08.033>
37. Ghiasi G, Lin T-Y, Le QV (2019) Learning scalable feature pyramid architecture for object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7036–7045
38. Girshick R (2015) Fast R-CNN. In: *2015 IEEE international conference on computer vision (ICCV)*, IEEE, Santiago, Chile, pp. 1440–1448
39. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 580–587. <https://doi.org/10.1109/CVPR.2014.81>
40. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: *IEEE conference on computer Vision & Pattern Recognition*, IEEE, Columbus, OH, USA
41. Guo J, Han K, Wang Y, Zhang C, Yang Z, Wu H, Chen X, Xu C (2020) HitDetector: hierarchical trinity architecture search for object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*
42. Han F, Yao J, Zhu H, Wang C (2020) Underwater image processing and object detection based on deep CNN method. *J Sensors* 20:Article ID 6707328. <https://doi.org/10.1155/2020/6707328>
43. Han K, Wang Y, Tian Q, et al. (2020) GhostNet: more features from cheap operations. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. <https://doi.org/10.1109/CVPR42600.2020.00165>
44. He K, Zhang X, Ren S, Sun J (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans Pattern Anal Mach Intell (TPAMI)* 37(9):1904–1916
45. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on Imagenet classification. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034. <https://doi.org/10.1109/ICCV.2015.123>
46. 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 (CVPR)*, pp. 770–778. <https://doi.org/10.1109/CVPR.2016.90>
47. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask R-CNN. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2961–2969
48. Hinton GE, Osindero S (2006) A fast learning algorithm for deep belief nets. *Neural Comput* 18(7):1527–1554
49. Hoang Ngan Le T, Zheng Y, Zhu C, Luu K, Savvides M (2016) Multiple scale Faster R-CNN approach to diver’s cell-phone usage and hands on steering wheel detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 46–53
50. Hoiem D, Chodpathumwan Y, Dai Q (2012) Diagnosing error in object detectors. In: *Computer Vision–ECCV*, Springer, pp. 340–353
51. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T et al (2017) MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861
52. Howard A, Sandler M, Chu G, Chen L-C, Chen B, Tan M et al (2019) Searching for MobileNetV3. In: *Proceedings of the IEEE international conference on computer vision (ICCV)*. <https://arxiv.org/abs/1905.02244>. Accessed 10/7/21
53. Hu M, Yang Y, Shen F, Zhang L, Shen H, Li X (2017) Robust web image annotation via exploring multi-facet and structural knowledge. *IEEE Trans Image Process* 26(10):4871–4884
54. Huang G, Liu Z, Maaten LVD, Weinberger KQ (2017) Densely connected convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 4700–4708, arXiv:1608.06993
55. Huang H, Zhou H, Yang X, Zhang L, Qi L, Zang A-Y (2019) Faster RCNN for marine organisms detection and recognition using data augmentation. *Neurocomputing* 337:372–384
56. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167

57. Jia J (2009) A machine vision application for industrial assembly inspection. In: Proceedings of 2009 second international conference on machine vision, IEEE, Dubai, UAE, pp. 172–176
58. Knausgard KM, Wiklund A, Sørtdalen TK et al (2020) Temperate fish detection and classification: a deep learning based approach. *Appl Intell Manuscr.* <https://doi.org/10.1007/s10489-020-02154-9>
59. Koreitem K, Girdhar Y, Cho W, Singh H, Pineda J, Dudek G (2016) Subsea fauna enumeration using vision-based marine robots. In: Proceedings of the IEEE conference on computer and robot vision, pp. 101–108
60. Krasin I, Duerig T, Alldrin N, et al. (2017) Openimages: A public dataset for large-scale multi-label and multi-class image classification. Dataset available from <https://github.com/openimages>. Accessed 10/7/21
61. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in neural information processing systems, vol.1 (NIPS), Lake Tahoe, Nevada, 25(2):1097–1105. <https://doi.org/10.5555/2999134.2999257>
62. Krizhevsky A, Sutskever I, Hinton GE (2017) ImageNet classification with deep convolutional neural networks. *Commun ACM* 60(6):84–90
63. Landola 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 preprint arXiv:1602.07360
64. Lavery PS, McMahon K, Weyers J, Boyce MC, Oldham CE (2013) Release of dissolved organic carbon from seagrass wrack and its implications for trophic connectivity. *Mar Ecol Prog Ser* 494:121–133
65. Law H, Deng J (2018) “CornerNet: Detecting objects as paired keypoints. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 734–750
66. Law H, Teng Y, Russakovsky O, Deng J (2019) CornerNet-lite: efficient keypoint based object detection. arXiv preprint arXiv:1904.08900
67. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proc IEEE* 86(11):2278–2324
68. Li Z, Peng C, Yu G, Zhang X, Deng Y, Sun J (2018) DetNet: design backbone for object detection. In: Proceedings of the European conference on computer vision (ECCV), pp. 334–350
69. Lin T-Y, Maire M, Belongie S, et al. (2014) Microsoft COCO: common objects in context. In: Proceedings of the European conference on computer vision (ECCV), pp. 740–755
70. Lin T-Y, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp. 2980–2988. <https://doi.org/10.1109/ICCV.2017.324>
71. Lin T-Y, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 2117–2125. <https://doi.org/10.1109/CVPR.2017.106>
72. Liu M (n.d.) AU-aware deep networks for facial expression recognition. In: 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, IEEE Press, Shanghai, pp. 1–6. <https://doi.org/10.1109/FG.2013.6553734>
73. Liu Z, Zhang Y, Yu X, Yuan C (2016) Unmanned surface vehicles: An overview of development and challenges. *Annu Rev Control* 41:71–93. <https://doi.org/10.1016/j.arcontrol.2016.04.018>
74. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC (2016) SSD: single shot multibox detector. In: Proceedings of the European conference on computer vision (ECCV), pp. 21–37
75. Liu S, Qi L, Qin H, Shi J, Jia J (2018) Path aggregation network for instance segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 8759–8768. <https://doi.org/10.1109/CVPR.2018.00913>
76. Liu S, Huang D, et al. (2018) Receptive field block net for accurate and fast object detection. In: Proceedings of the European conference on computer vision (ECCV), pp. 385–400
77. Liu S, Li X, Gao M, Cai Y, Nian R, Li P, et al. (2018) Embedded online fish detection and tracking system via yolo-v3 and parallel correlation filter. In: OCEANS 2018 MTS/IEEE Charleston. IEEE, pp 1–6
78. Liu S, Huang D, Wang Y (2019) Learning spatial fusion for single-shot object detection. arXiv preprint arXiv:1911.09516
79. Lu H, Li Y, Zhang Y, Chen M, Serikawa S, Kim H (2017) Underwater optical image processing: a comprehensive review. *Mobile Netw Appl* 22:1202–1211. <https://doi.org/10.1007/s11036-017-0863-4>
80. Luo Y, Wan Y (2013) A novel efficient method for training sparse auto-encoders. In: 2103 6th international congress on image and signal processing, pp. 1019–1023
81. Ma N, Zhang X, Zheng H-T, Sun J (2018) ShuffleNetV2: practical guidelines for efficient cnn architecture design. In: Proceedings of the European conference on computer vision (ECCV), pp. 116–131. <https://arxiv.org/abs/1807.11164>. Accessed 10/7/21
82. Mahmood A, Bennamoun M, An S, Sohel F, Boussaid F, Hovey R, Kendrick G, Fisher RB (2016) Coral classification with hybrid feature representations. In: IEEE International Conference on Image Processing, IEEE Press, Arizona, pp. 519–523

83. Marr B (n.d.) Key milestones of Waymo – Google’s self-driving cars. <https://www.forbes.com/sites/bernardmarr/2018/09/21/keymilestones-of-waymo-googles-self-driving-cars/#3831b2965369>. Accessed 18/12/2021
84. Meda-Campaña JA (2018) On the estimation and control of nonlinear systems with parametric uncertainties and noisy outputs. *IEEE Access* 6:31968–31973. <https://doi.org/10.1109/ACCESS.2018.2846483>
85. Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp. 807–814
86. Online Available at (n.d.) “<https://www.pureadvantage.org/news/2016/11/15/underwater-robots>”. Accessed 18/12/2021
87. Online Available at (n.d.) “<https://www.ecagroup.com/en/solutions/h800-rov-remotely-operated-vehicle>”. Accessed 18/12/2021
88. Pang J, Chen K, Shi J, Feng H, Ouyang W, Libra LD (2019) R-CNN: towards balanced learning for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 821–830
89. Qin H, Li X, Yang Z, Shang M (2015) When underwater imagery analysis meets deep learning: a solution at the age of big visual data. In: OCEANS 2015 - MTS/IEEE Washington, IEEE Press, Washington DC, pp. 1–5. <https://doi.org/10.23919/OCEANS.2015.7404463>
90. Rashwan A, Kalra A, Poupart P (2019) Matrix nets: a new deep architecture for object detection. In: Proceedings of the IEEE international conference on computer vision workshop (ICCV workshop), pp. 2025–2028. <https://doi.org/10.1109/ICCVW.2019.00252>
91. Redmon J, Farhadi A (2017) YOLO9000: better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7263–7271. <https://doi.org/10.1109/CVPR.2017.690>
92. Redmon J, Farhadi A (2017) Yolo9000: better, faster, stronger. In: Computer vision and pattern recognition (CVPR), 2017 IEEE conference, pp. 6517–6525. <https://doi.org/10.1109/CVPR.2017.690>
93. Redmon J, Farhadi A (2018) YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767
94. 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 (CVPR), pp. 779–788. <https://doi.org/10.1109/CVPR.2016.91>
95. Ren S, He K, Girshick R, Sun J (2015) Faster RCNN: towards real-time object detection with region proposal networks. In: Proceedings of the advances in neural information processing systems, pp. 91–99. <https://doi.org/10.1109/TPAMI.2016.2577031>
96. Rifai S, Vincent P, Muller X, Glorot X, Bengio Y (2011) Contractive autoencoders: explicit invariance during feature extraction. In: Proceedings of the 28th international conference on international conference on machine learning, pp. 833–840
97. Rubio J (2009) SOFMLS: online self-organizing fuzzy modified least-squares network. *IEEE Trans Fuzzy Syst* 17(6):1296–1309. <https://doi.org/10.1109/TFUZZ.2009.2029569>
98. Rubio J (n.d.) Stability analysis of the modified Levenberg-Marquardt algorithm for the artificial neural network training. In: IEEE Transactions on Neural Networks and Learning Systems, <https://doi.org/10.1109/TNNLS.2020.3015200>
99. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) ImageNet large scale visual recognition challenge. *Int J Comput Vis (IJCV)* 115:211–252. <https://doi.org/10.1007/s11263-015-0816-y>
100. Sa I, Ge Z, Dayoub F, Uppcroft B, Perez T, McCool C (2016) Deepfruits: a fruit detection system using deep neural networks. *Sensors* 16(8)
101. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C (2018) MobileNetV2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 4510–4520
102. Schmidhuber J (2015) Deep learning in neural networks: an overview. *Neural Netw* 61:85–117
103. Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun Y (2014) Overfeat: integrated recognition, localization and detection using convolutional networks. Preprint arXiv:1312.6229
104. Sharma P (2018) A step-by-step introduction to the basic object detection algorithms part 1. Analytics Vidhya, Introduction
105. Sharma A, Singh PK, Khurana P (2016) Analytical review on object segmentation and recognition. In: 2016 6th international conference – cloud system and big data engineering (confluence), pp. 524–530. <https://doi.org/10.1109/CONFLUENCE.2016.7508176>
106. Simonyan K, Zisserman A (n.d.) Very deep convolutional networks for large scale image recognition. In: IEEE International Conference on Learning Representations, pp. 730–734. <https://arxiv.org/abs/1409.1556>. Accessed 18/12/2021

107. Song HA, Lee S-Y (2013) Hierarchical representation using NMF. In: Lee M, Hirose A, Hou Z-G, Kil RM (eds) ICONIP 2013. LNCS, 8226:466–473. Springer, Heidelberg. https://doi.org/10.1007/978-3-642-42054-2_58
108. Sun F, Yu J, Chen S, Xu D (2014) Active visual tracking of free-swimming robotic fish based on automatic recognition. In: Proceedings of the IEEE conference on intelligent control and automation, pp. 2879–2884
109. Sung M, Yu S-C, Girdhar Y (2017) Vision based real-time fish detection using convolutional neural network. In: OCEANS 2017 – Aberdeen. IEEE, pp. 1–6, <https://doi.org/10.1109/OCEANSE.2017.8084889>
110. Swart S, Zietsman J, Coetzee J, Goslett D, Hoek A, Needham D, Monteiro P (2016) Ocean robotics in support of fisheries research and management. *Afr J Mar Sci* 38(4):525–538
111. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9. <https://doi.org/10.1109/CVPR.2015.7298594>
112. Szegedy C, Ioffe S, Vanhoucke V (2016) Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261
113. Takagi M, Mori H, Yimit A, Hagihara Y, Miyoshi T (2016) Development of a small size underwater robot for observing fisheries resources-underwater robot for assisting abalone fishing. *J Robot Mechatron* 28(3): 397–403
114. Tan M, Le QV (2019) EfficientNet: rethinking model scaling for convolutional neural networks. In: Proceedings of international conference on machine learning (ICML)
115. Tan M, Le QV (2019) MixNet: mixed depthwise convolutional kernels. In proceedings of the British machine vision conference (BMVC)
116. Tan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, Le, QV (2019) MNASnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2820–2828
117. Tan M, Pang R, Le QV (2020) EfficientDet: scalable and efficient object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). <https://doi.org/10.1109/CVPR42600.2020.01079>
118. Tian Z, Shen C, Chen H, He T (2019) FCOS: fully convolutional one-stage object detection. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp. 9627–9636. <https://doi.org/10.1109/ICCV.2019.00972>
119. Villon S, Chaumont M, Subsol G, Villéger S, Claverie T, Mouillot D (2016) Coral reef fish detection and recognition in underwater videos by supervised machine learning: comparison between deep learning and HOG+SVM methods. *ACTVS: advanced concepts for intelligent vision systems*, Lecce, Italy. fihal-01374123
120. Vincent P, Larochelle H, Bengio Y, Manzagol PA (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning, pp. 1096–1103
121. Wang Y, Lan Y, Zheng Y, Lee K, Cui S, Lian J (2013) A UGV based laser scanner system for measuring tree geometric characteristics. In: Proceedings of 2013 SPIE International Symposium on Photoelectronic Detection and Imaging, SPIE, Beijing China, vol. 8905
122. Wang C-Y, Liao H-YM, Wu Y-H, Chen P-Y, Hsieh J-W, Yeh I-H (2020) CSPNet: a new backbone that can enhance learning capability of cnn. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPR Workshop). <https://doi.org/10.1109/CVPRW50498.2020.00203>
123. Woo S, Park J, Lee J-Y, Kweon IS (2018) CBAM: convolutional block attention module. In: Proceedings of the European conference on computer vision (ECCV), pp. 3–19
124. Xavier G, Antoine B, Yoshua B (2011) Deep sparse rectifier neural networks. In: Proceedings of the 14th international conference on artificial intelligence and statistics, pp. 315–323
125. Xie S, Girshick R, Dollár P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 1492–1500
126. Xu W, Matzner S (2018) Underwater fish detection using deep learning for water power applications. *arXiv preprint arXiv: 1811.01494*
127. Yang Y, Dong J, Sun X, Lguensat R, Jian M, Wang X (2016) Ocean front detection from instant remote sensing SST images. *IEEE Geosci Remote Sens Lett* 13(12):1960–1964
128. Yang Y, Dong J, Sun X, Lima E, Mu Q, Wang X (2017) A CFCC-LSTM model for sea surface temperature prediction. *IEEE Geosci Remote Sens Lett* 15(2):207–211

129. Yang Z, Liu S, Hu H, Wang L, Lin S (2019) RepPoints: point set representation for object detection. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp. 9657–9666. <https://doi.org/10.1109/ICCV.2019.00975>
130. Yu F, Koltun V, Funkhouser T (2017) Dilated residual networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 472–480
131. Yuan X, Huang B, Wang Y, Yang C, Gui W (2018) Deep learning-based feature representation and its application for soft sensor modeling with variable-wise weighted SAE. *IEEE Trans Ind Inf* 14(7):3235–3243
132. Yuh J, Marani G, Blidberg DR (2011) Applications of marine robotic vehicles. *Intell Serv Robot* 4(4):221–231
133. Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: European Conference on Computer Vision, pp. 818–833
134. Zeiler MD, Taylor GW, Fergus R (2011) Adaptive deconvolutional networks for mid and high-level feature learning. In: 2011 International Conference on Computer Vision, pp. 2018–2025. <https://doi.org/10.1109/ICCV.2011.6126474>
135. Zhang L, Lin L, Liang X, He K (2016) Is faster RCNN doing well for pedestrian detection? In: Proceedings of the European Conference on Computer Vision, Springer, pp. 443–447
136. Zhang X, Zhou X, Lin M, Sun J (2018) ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 6848–6856
137. Zhao Q, Sheng T, Wang Y, Tang Z, Chen Y, Cai L, Ling H (2019) M2det: a single-shot object detector based on multi-level feature pyramid network. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI) 33:9259–9266

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