


Deep Learning Approach For Objects Detection in Underwater Pipeline Images

Boris Gašparović ^{a,b}, Jonatan Lerga ^{a,b}, Goran Mauša ^{a,b}, and Marina Ivašić-Kos ^{b,c}

^aFaculty of Engineering, University of Rijeka, Rijeka, Croatia; ^bCenter for Artificial Intelligence and Cybersecurity, University of Rijeka, Rijeka, Croatia; ^cFaculty of Informatics and Digital Technologies, University of Rijeka, Rijeka, Croatia

ABSTRACT



In this paper, we present automatic, deep-learning methods for pipeline detection in underwater environments. Seafloor pipelines are critical infrastructure for oil and gas transport. The inspection of those pipelines is required to verify their integrity and determine the need for maintenance. Underwater conditions present a harsh environment that is challenging for image recognition due to light refraction and absorption, poor visibility, scattering, and attenuation, often causing poor image quality. Modern machine-learning object detectors utilize Convolutional Neural Network (CNN), requiring a training dataset of sufficient quality. In the paper, six different deep-learning CNN detectors for underwater object detection were trained and tested: five are based on the You Only Look Once (YOLO) architectures (YOLOv4, YOLOv4-Tiny, CSP-YOLOv4, YOLOv4@Resnet, YOLOv4@DenseNet), and one on the Faster Region-based CNN (RCNN) architecture. The models' performances were evaluated in terms of detection accuracy, mean average precision (mAP), and processing speed measured with the Frames Per Second (FPS) on a custom dataset containing underwater pipeline images. In the study, the YOLOv4 outperformed other models for underwater pipeline object detection resulting in an mAP of 94.21% with the ability to detect objects in real-time. Based on the literature review, this is one of the pioneering works in this field.

ARTICLE HISTORY

Received 31 August 2022
Revised 31 October 2022
Accepted 4 November 2022

Introduction

Submarine pipelines are mainly used to carry oil, gas, and water. Harsh underwater environment conditions often change the appearance and state of installed pipes. In order to guarantee the regular operation of the subsea pipeline infrastructure, the detections of submarine pipeline components and leakage are essential. Since remotely operated vehicles (ROVs) can adapt to the harsh sea environment, they can replace human visual underwater inspections. Nowadays, computer vision is used to assist the ROVs in completing various underwater tasks, such as underwater pipeline object detection and inspection, tracking, scene reconstruction, and other (Jacobi and Karimanzira

CONTACT Jonatan Lerga  jlerga@griteh.hr  Faculty of Engineering, University of Rijeka, Vukovarska 58, Rijeka 51000, Croatia

© 2022 The Author(s). Published with license by Taylor & Francis Group, LLC.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

2013; Lu et al. 2017). The primary operational challenge for underwater vehicles is that the underwater environment often significantly affects visual sensing despite using high-quality cameras.

The performance of a vision-based inspection is severely impacted by the quality of underwater imagery, which is often highly degraded by optical artifacts. Those artifacts include poor visibility, light refraction, absorption, scattering, and attenuation. Light scattering is caused by a light ray incident on the object reflected and deflected multiple times by particles present in the water before reaching the camera; this reflection introduces a homogeneous background noise to the image. Attenuation causes exponential decay of light between the image scene and the camera (Uplavikar, Wu, and Wang 2019). The subsea environment presents a unique challenge to the perception that is not present on the land; sea-land has a significant diversity of underwater image distributions. The images captured in deep oceanic water look different from those captured in muddy waters or shallow coastal waters. Color distribution can be manipulated by varying degrees of attenuation encountered by light traveling in the water with different wavelengths. As light propagation differs underwater (compared to the air), a unique set of non-linear image distortion occurs, propelled by various factors (such as attenuation and scattering). Underwater tends to have a dominating green or blue hue since red wavelengths get absorbed in deep water (Schettini and Corchs 2010).

Object detection is a critical problem that is utilized in a wide range of industries for sorting, inspection, monitoring, and other purposes. The traditional vision-based detection method for underwater pipeline and cable detection is based on the edge information in images (Narimani, Nazem, and Loueipour 2009). Harsh underwater environments impact methods that use edge information by reducing object detection accuracy. In order to improve detection speed and accuracy, the generic method based on Convolutional Neural Network (CNN) occupies a dominant position in object detection research today. The CNN can be divided into two main categories (Zhao et al. 2019): Region Proposal-Based Framework (two-stage) and Regression/Classification-Based Framework (one-stage).

The region proposal-based framework is a two-step process that first gives a coarse scan of the whole scenario and then focuses on regions of interest (RoIs). Girshick et al. (2014) proposed R-CNN, which adopts the CNN to produce RoIs in order to localize and segment objects and a pretrained linear Support-Vector Machine (SVM) classifier to categorize the produced region of interests. The R-CNN training is expensive in memory and time. Features are extracted from different RoIs and stored on the disk. The Fast R-CNN achieved impressive improvements in both accuracy and efficiency, but not enough for real-time detection (Girshick 2015). The Faster R-CNN uses a Region Proposal Network that shares full-image convolutional features with the detection network (Ren et al. 2015). It has been used for real-time

detection, face detection (Jiang and Learned-Miller 2017), pedestrian recognition (Zhao et al. 2016), seagrass detection (Moniruzzaman et al. 2019) and in other fields where inference speed in real-time is not crucial.

A regression-based framework, also called the single-stage detector based on global regression, performs mapping straight from the image pixels to bounding box coordinates and class probabilities, which can reduce computational cost. To overcome the problem of the poor real-time performance of the target detection in R-CNN, Redmon et al. (2016) proposed a novel real-time object detector called YOLO. It makes use of the whole topmost feature map to classify and locate objects in one step. Based on YOLO, Redmon et al. proposed YOLOv2 (2017) and YOLOv3 (2018). YOLOv2 adopts a max-pooling layer and batch normalization, which improves detection accuracy and speed. YOLOv3 uses RESNET and faster R-CNN RPN, which improves spatial representation. Bochkovskiy et al. (Bochkovskiy, Wang, and Liao 2020) proposed YOLOv4 based on a combination of new features, which improve detection accuracy.

The rest of the paper is structured as follows. [Section 2](#) provides an overview of the related work in the field of object detection. The methodology of our study and elaboration on trained deep-learning models are provided in [section 3](#), followed by the description of the experiment setup given in [section 4](#). A detailed assessment of the obtained results is provided in [section 5](#). The paper conclusion and future work directions are given in [section 6](#).

Related Work

Object detection is one of the tasks of computer vision systems, where its goal is to recognize objects and locate them in an image. Deep learning models are shown to be capable of recognizing and extracting information from images in difficult environments while simultaneously working with a vast amount of data. Underwater object detection is generally achieved by sonar, laser, and cameras. Compared to sonar and laser, the cameras are low-cost, and they can capture more types of visual information with high temporal and spatial resolution.

YOLO has been adopted by various researchers for the purpose of underwater object detection because of its high detection efficiency. As an example, Xu and Matzner (2018) utilized YOLOv3 for underwater fish detection for waterpower application. With high turbidity, rapid velocity, and murky water, the datasets utilized to train and test the model were challenging. The testing of the model yields a mean average precision (mAP) value of 54.92%. Another version of YOLO was used for fish detection in research by Sung, Yu, and Girdhar (2017). They trained the YOLOv1 detector on a custom dataset consisting of 929 fish images with annotation having no negative class images. Testing of the model achieved 65.3% mAP. Raza and Hong (2020) improved

the YOLOv3 method for detecting fish in demand for monitoring the marine ecosystem. The improved version of YOLOv3 uses k-means clustering to increase the anchor boxes, transfer learning technique, improved loss function, and increased detection scale. The results show it outperforms the original YOLOv3 on the task of fish detection by 4% in terms of the mAP. Asyraf et al. (2021) investigated four versions of the YOLOv3 detector (they trained the original YOLOv3, Tiny-YOLOv3, YOLOv3-SPP, and Tiny-YOLOv3-PRN) on two open-source datasets to determine the efficiency of the model's ability to detect underwater life. Results showed significant evidence that YOLOv3 can detect underwater objects with a ranging mAP score from 74.88% to 97.56%. Application of the newer version of the YOLO detector, YOLOv4, was demonstrated in research performed by Rosli et al. (2021) for underwater animal detection. The dataset used to train and test the model was challenging due to the varying visibility. The training results show the mAP score of 97.86%.

Aside from fish detection, computer vision has been employed for a variety of other underwater applications. Chen et al. (2021) utilized YOLOv4 for underwater target recognition on a dataset named Underwater Robot Picking Contest (URPC). The URPC dataset contains 4757 images of four target categories: echinus, starfish, holothurian, and scallop. The detection results show 73.48% mAP. Training and testing of the YOLOv4 on the same URPC dataset were conducted by Zhang et al. (2021) achieving testing results of 81.01% mAP. In order to protect the underwater biodiversity, Tian et al. (2021) tackle the problem of aquatic environment pollution. They developed a computer-vision-based autonomous underwater garbage cleaning robot utilizing a modified YOLOv4 detection network. The detection with the trained model achieved results of 90.3% mAP. Lei et al. (2022) utilized the YOLOv4 method for detecting swimming and drowning behavior patterns. Their study resulted in the mAP value of 89.23% for drowning and 93.86% for swimming behavior, respectively.

Underwater object detection is also used in aquaculture for formulating scientific feeding strategies that can effectively reduce feed waste and water pollution, which is a win-win scenario in terms of economic and ecological benefits. The detection of uneaten feed pellets provides rich information for formulating scientific feeding strategies. Hu et al. (2021) utilized improved YOLOv4 to detect uneaten feed pellets in underwater images. The custom dataset consists of blurred and high-density images captured from a net cage located in the cold-water mass area of the Yellow Sea of China. The original YOLOv4 method was improved by changing the PANet network structure, adding the DenseNet shortcut connection, and reducing the number of network layers. The training and testing results of the improved YOLOv4 method achieved the mAP score of 92.61% on the test dataset.

Another use of underwater computer vision is pipeline detection, which is also the focus of this paper. Underwater pipeline detection was done in research by Zhao, Wang, and Du (2020). The researchers used the YOLOv3 algorithm to locate the oil spill point of the underwater pipeline. In a training network, there are two types of detection targets: pipeline and leakage point. The trained model was able to achieve 77.5% of leakage point detection accuracy with 36 frames per second of processing time. Detection accuracy for the pipeline was 93.67%. Based on the literature review, we found just this one paper applying the deep CNN for underwater pipeline object detections (limited to distinguishing just two object classes); hence, to the best of our knowledge, our study may be considered one of the pioneer researches in the field. Next, we present deep-learning models utilized for this purpose in our study.

Methodology

This section, presenting the methodology set up and elaboration on trained deep-learning models, is divided into two subsections. The first subsection explains the architectures of each version of the utilized YOLO object detector; the second describes Faster RCNN, an object detection method whose detection results are later compared to YOLO results.

Introduction to the YOLO Architectures

For our case study, we chose the YOLO method because it achieves near-state-of-the-art performance for object detection tasks in a variety of applications. The original YOLO paper (Redmon et al. 2016) describes the proposed algorithm that is based on regression; instead of selecting the interesting part of an image, and predicts class probabilities and bounding boxes for the whole image in one run of the algorithm.

The network architecture of the original YOLO model is based on the CNN, as shown in Figure 1. It is the first implementation of the single-stage detector concept and uses reduction layers of dimension 1×1

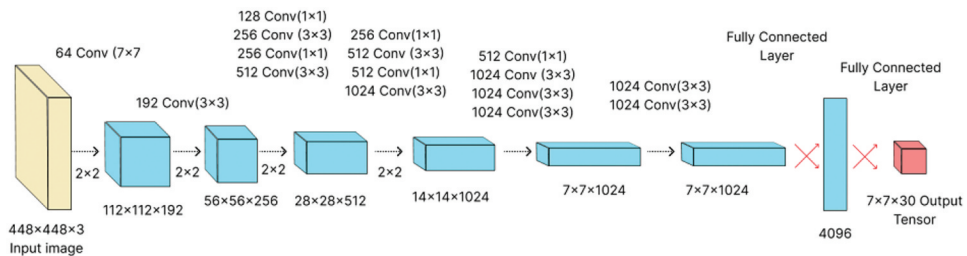


Figure 1. Yolov1 architecture.

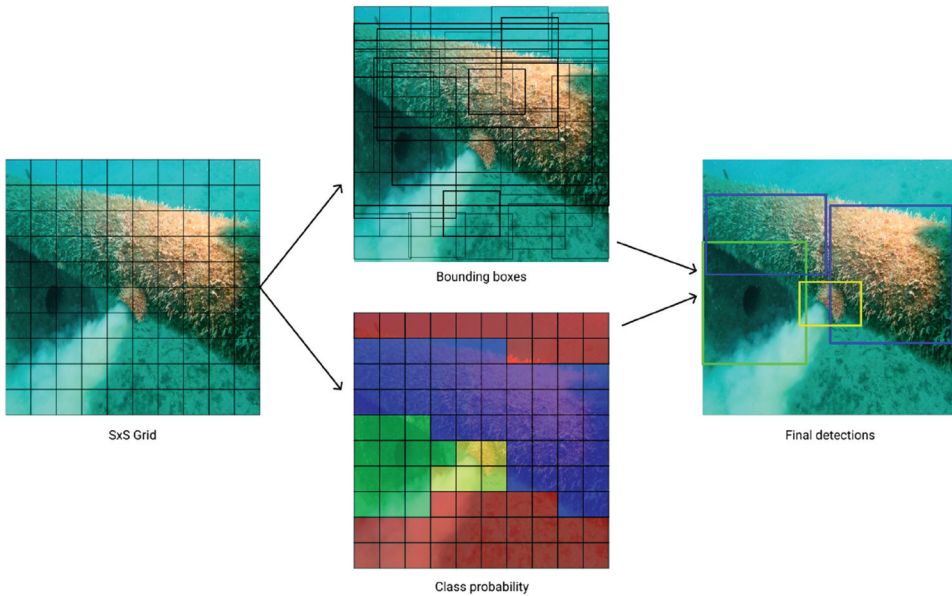


Figure 2. The detection pipeline of YOLO.

followed by a convolutional layer of dimension 3×3 and batch normalization and leaky ReLU activation function. The YOLOv1 network has 24 convolutional layers and two fully connected layers. Its detection pipeline is shown in [Figure 2](#). The convolutional layers perform feature extraction, while fully connected layers predict bounding box location and class probabilities. YOLO splits the input image into cells, typically a $S \times S$ grid. Each cell is then responsible for predicting two bounding boxes with correspondent probabilities. YOLO determines the probability that the cell contains a particular class during the one pass of the forward propagation. The bounding box around an object has a confidence value corresponding to the IoU score of the bounding box and the ground truth box. Versions YOLOv2 (Redmon and Farhadi 2017) and YOLOv3 (Redmon and Farhadi 2018) use max-pooling layers and different way of generating bounding box proposals with network depths of 19 and 53 layers. Additionally, YOLOv3 can perform multilabel classification achieved by replacing the softmax with logistic regression to calculate the possibility that an input belongs to a specific tag.

YOLOV4

The YOLOv4 (Bochkovskiy, Wang, and Liao 2020) network is composed of four distinct sections: input, backbone, neck, and dense prediction. The structure is shown in [Figure 3](#). The backbone of YOLOv4 is defined as the essential feature-extraction architecture. The backbone is Darknet53, which was used in the original YOLOv3, but it has been enhanced with Cross-Stage-

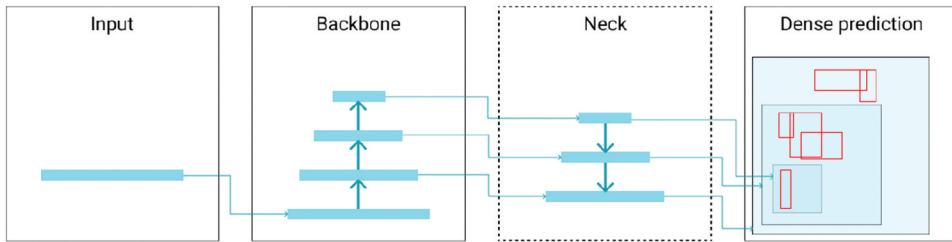


Figure 3. YOLOv4 structure.

Partial (CSP) connections (Wang et al. 2020). As a result, the backbone was named CSPDarknet53. This backbone can improve CNN's learning potential by assisting in the development of a robust object detection model, especially in our case of underwater computer vision. CSPDarknet53 consists of 53 layers of 3×3 and 1×1 filters, 725×725 receptive fields, and 27.6 M parameters. This architecture has proven superior to its competitor architecture, CSPResNet50 (Bochkovskiy, Wang, and Liao 2020). The authors of YOLOv4 chose a modified version of Path Aggregation Network (PANet) (Liu et al. 2018) as the architecture's neck. For the prediction step, each feature needs to be flattened first, which is accomplished with Spatial Pyramid Pooling (SPP) (He et al. 2015). The SPP significantly increases receptive field performance by bringing out contextual features. The head section consists of dense prediction, which plays an important role in producing the final prediction and locating bounding boxes. This same head section can be found in the YOLOv3 implementation, which detects the bounding box coordinates and confidence score for a specific class. In short, the YOLO head works in three steps. First, it divides the entire image into $N \times N$ grids. Each grid has five parameters (x , y , h , w ; and c ; confidence score), where (x, y) is the offset value between the prediction box and the respective grid cell-bound. Parameters (h, w) are the height and width from the prediction box to the entire image; confidence score c is the probability of the class object. Second, CNN extracts the feature and predicts classes with class probability scores. Finally, non-maximum suppression is used to eliminate repetitive bounding boxes. Improvements created to help enrich the YOLOv4 capability for underwater usage are Mosaic and Cutmix data augmentation process (Yun et al. 2019).

The data augmentation method, named Mosaic, was introduced by the original YOLOv4 authors. It mixes four training images, resulting in mixing four different contexts. This allows the detection of objects outside their normal context. In addition, batch normalization calculates activation statistics from four different images on each layer, significantly reducing the need for a large mini-batch size. Regional dropout strategies were used as data augmentation steps to enhance the performance of the CNNs. These augmentations remove informative pixels in training images

by overlaying them with a patch of either black pixels or random noise. It makes the model focus on non-discriminative parts of the object but causes information loss. The CutMix augmentation helps the model classify two objects from their partial views in the same images by taking two images and labeling pairs. Its strategy is to cut out and paste patches among training images where the ground truth labels are also mixed proportionally to the area of the patches. The Cutmix augmentation increases localization ability by making the model focus on less discriminative parts of the classified object.

YOLOv4 Tiny

YOLOv4 Tiny (Wang, Bochkovskiy, and Liao 2021) is a simplified and lightweight version of YOLOv4 that may be used to design applications for mobile and embedded devices. It works on the same idea as the original model, but with a different set of parameters that minimize the convolutional layer's depth. YOLOv4 Tiny has only two YOLO heads as opposed to three in YOLOv4, and it has been trained from 29 pretrained convolutional layers as opposed to YOLOv4 which has been trained from 137 pretrained convolutional layers. Supposing that the size of the input figure is 416×416 and feature classification is 80, the YOLOv4 Tiny network structure is shown in Figure 4. Those changes helped the network achieve faster detections. The YOLOv4 Tiny method uses a feature pyramid network to extract feature maps with different scales and increase object detection speed without using the spatial pyramid pooling and path aggregation network used in the YOLOv4 method. At the same time, the YOLOv4 Tiny uses two different scale feature maps that are 13×13 and 26×26 to predict the detection results. However, the accuracy for YOLOv4 Tiny is approximately two-thirds that of the YOLOv4 when tested on the MS COCO dataset (Lin et al. 2014).

CSP-YOLOv4

Wang, Bochkovskiy, and Liao (2021) proposed a network scaling approach that modifies not only the depth, width, and resolution but also the structure of the network. CSP-YOLOv4 was introduced to get a better speed/accuracy trade-off by converting the first CSP stage in the backbone into the original DarkNet residual layer. The PAN architecture is CSP-ized in order to reduce the amount of computation effectively.

Modified backbone of YOLOv4

The YOLOv4 has CSPDarknet53 as its backbone. The model backbone can be modified in order to have different detection results. Our paper uses a modified version of the YOLOv4 backbone to compare results obtained with the original backbone CSPDarknet53. Models ResNet50-YOLO and DenseNet201-YOLO

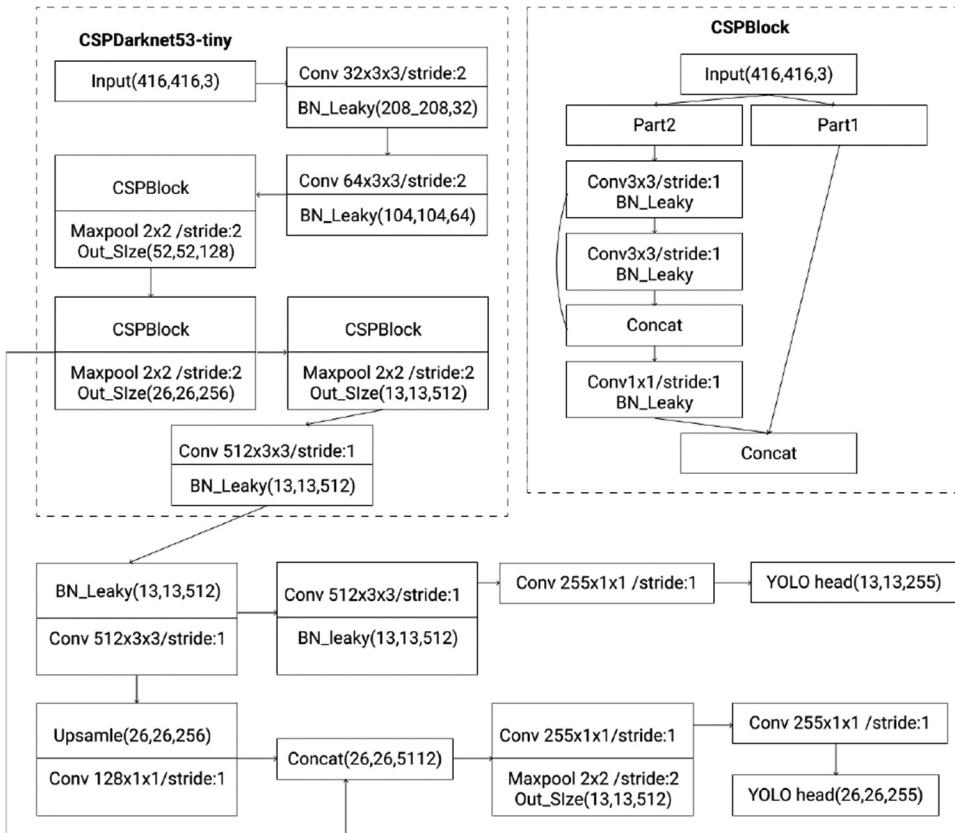


Figure 4. Yolov4 tiny network structure.

were used to train and test the detection and recognition of underwater targets. ResNet50 is a deep convolutional neural network that is 50 layers deep. He et al. (2016) proposed an innovative neural network that won the top position at the ILSVRC competition. The strength of this model lies in skip connections that connect blocks of the network which enables the same performance for higher layers. The residual network (ResNet) improves the efficiency of deep neural networks by adding outputs from previous layers to the outputs of stacked layers, making it possible to train much deeper networks. In a DenseNet architecture, each layer is connected to every other layer, hence the name Densely Connected Convolutional Network. DenseNet requires fewer parameters, as there is no need to learn redundant feature maps. DenseNet concatenates the output feature maps of the layer with the incoming feature maps.

Faster RCNN

We compared results in underwater object detection achieved by the YOLO-based models described above to Faster R-CNN. In the field of object

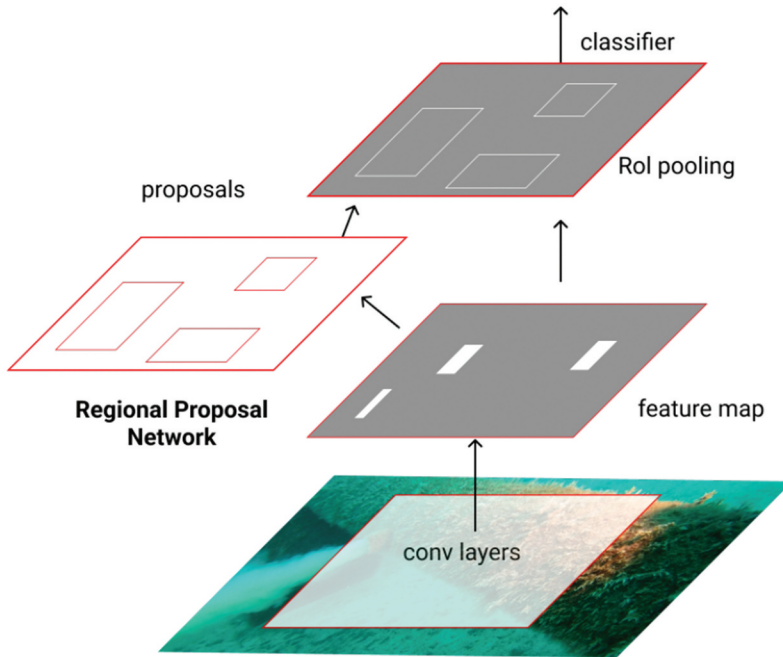


Figure 5. An overview of object detection with faster R-CNN.

detection, the Faster R-CNN is a classic two-stage method. Ren et al. improved the R-CNN method for object detection by adding region proposal networks (RPN) that share CNN layers with the same network for object detection (Ren et al. 2015). Overview of object detection with Faster R-CNN is shown in Figure 5. A Faster R-CNN object detection network consists of a feature proposal network for extracting the useful features of the target, an RPN whose task is to propose regions of interest, and a Fast R-CNN detector to classify the regions (Girshick 2015). The whole structure of the feature proposal network consists of 13 convolutional layers. Each convolutional layer is followed by a maximum pooling layer. In practical application, the more convolutional layers used, the more image features extracted, and the better the recognition effect of the network on unknown images. The features are used as input to the box regression and classification layer. The RPN outputs the proposed regions and their region score. The core idea of Faster R-CNN is to avoid the two-stage detection technique. The RPN network is created with extra CNN layers, which perform regression simultaneously to produce the region proposal and the region score. The spatial window sliding technique is used to generate region proposals from the convolutional feature map. For every sliding window location, RPN predicts more than one region proposal. Fast R-CNN is responsible for classifying the region of interest and fine-tuning the location border, judging whether the region of interest identified by RPN contains the target and the target category. In this work, Detectron2 Faster

RCNN implementation was used (Detectron2 is a PyTorch-based modular object detection library) (Wu et al. 2019).

Experiment Setup

This section, presenting the experimental results, is divided into three subsections. The first subsection discusses the preparation of the underwater dataset for YOLOv4 models. The second describes the training process and requirements, and the last subsection lists evaluation measures for trained object detectors.

Data Preparations

The data for the experiment were collected by a remotely operated vehicle (ROV) recording underwater pipelines and from different camera angles. After recording the videos, two frames per video second were extracted to create the dataset for analysis. Namely, the dataset consists of 3021 images taken from three main camera shooting directions (above, left, and right angles), with every shooting angle having the same number of representing images. Extracted frames are then labeled using a labeling tool YOLO-Label (2019), as shown in Figure 6. The dataset distribution is shown in Table 1. The dataset was split up into 80:10:10 ratio, training part of dataset consisted of 2415 different images, testing and validation part of dataset comprised of 303 images taken from three camera angles. Each part of the dataset contains the same number of different camera angle images. An annotation of each image is

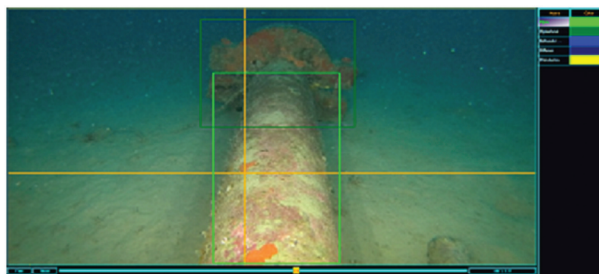


Figure 6. Snapshot of labelling tool.

Table 1. Dataset description.

Dataset	Description
Annotated Image:	3021 annotations
Number of Classes	5
Training	2415 images
Validation	303 images
Testing	303 images

given in the text file. Images were labeled in YOLO format containing details on object class, bounding box coordinates, and the height and width of the bounding box (with the most bottom-left point as the origin). Bounding box coordinates consist of center x and center y , which represent the coordinates of the center points of the bounding box. The distance of center from x -axis is represented as center x , and center y is the distance of the center from the y -axis. The coordinates are normalized to lie within the range $[0, 1]$ which makes them easier to work with even after scaling or stretching images.

Training

Neural network framework (in particular, open-source framework Darknet (Redmon 2013–2016)) is used to provide flexible APIs and configuration options for performance optimization since it is designed to facilitate and fasten the training of deep learning models (Shatnawi et al. 2018). Darknet is written in C and CUDA, allowing for the execution of the training and detection in the Graphical Processing Unit (GPU). The training was performed on the workstation with the following hardware: Intel(R) Xeon(R) CPU E5–2620 v4 @2.10 GHz, NVIDIA GeForce RTX 2080 Ti (11GB of graphic memory), and 128GB RAM.

The training setup has five types of detection targets: pipeline, leakage point, concrete weight, concrete mat, and pipe coupling. Thus, the configuration files were modified in order to define parameters used during training. In particular, the number of full connection layers output of the YOLOv4 is set to 5 because we have five classes, and the number of filters is obtained by $(classes + 5) \times 3$. The number of filters for YOLOv4@ResNet50 is set to 50 due $(classes + 5) \times 5$. The YOLOv4 uses 30 filters and can detect up to three objects per grid cell, while YOLOv4@Resnet50 uses 50 filters with the ability to detect five objects per grid cell. The subdivision number for training YOLOv4-Tiny was 64, as for YOLOv4, and 16 which takes more of an image into account during processing. Here, we were able to use a smaller number of subdivisions for YOLOv4-Tiny since its network is shallower compared to other models. Transfer learning is utilized for all YOLO models. Models were pretrained on the public COCO dataset. All training parameters are given in Table 2.

Table 2. Training parameters configuration.

Model	Batch size	Subdivision	Width x Height	Momentum	Decay	Learning rate	Activation
YOLO ¹	64	64	416x416	0.949	0.00005	0.0013	Mish
Tiny ²	64	64,16	416x416	0.9	0.0005	0.00261	Leaky
CSP ³	64	64	416x416	0.949	0.0005	0.001	Mish
Mod ⁴	64	32	416x416	0.9	0.0005	0.0001	Leaky
Mod ⁵	64	8	416x416	0.9	0.0005	0.0001	Leaky

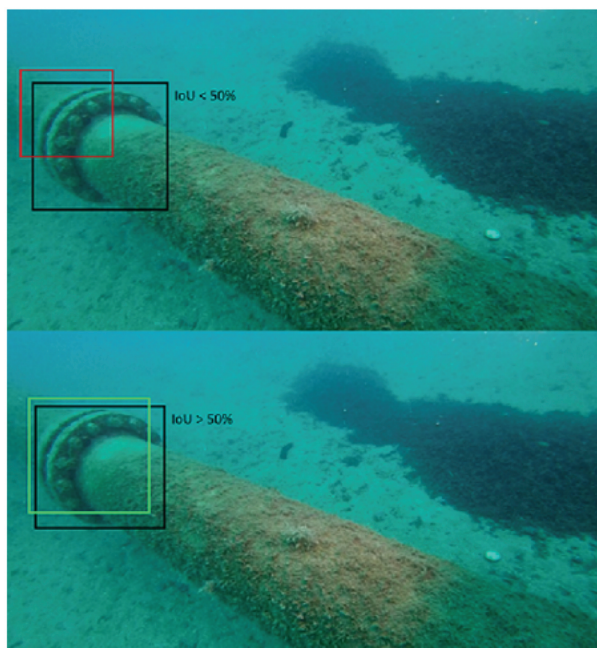


Figure 7. Visual representation of IoU criterion.

When the confidence of output is less than the threshold value of 0.5, it was interpreted as there is no target class. The output of the corresponding full connection layer is interpreted as the target class when maximum confidence taken is greater than a threshold value. The detection results are compared to the ground truth in order to determine whether the detection is a true positive. The detected bounding box's intersection over union (IoU) score should be at least 50%. [Figure 7](#) shows an example of positive and negative object detection for intersection over union (IoU) score in the case of pipeline detection. In the case of multiple detections of the same object, only one detection is counted as a true positive. Non-maximal suppression is used to choose the correct detection result.

The network's input should be an image, so the video is processed by extracting frames, which are then forwarded to the YOLO algorithm for object detection. The YOLO output provides the confidence score and the class ID of the object class in the bounding box. After the training, the detection model was tested on the test dataset, which was not included in the training and validation process.

Performance Evaluation

As evaluation metrics, detection accuracy, mean average precision (mAP), and frames per second (FPS) are used. The detection accuracy, as in Equation 1,

refers to the ratio of the number of prediction boxes to the total number of prediction boxes when the intersection ratio of prediction boxes and annotation boxes is greater than 0.5:

$$Accuracy = \frac{\text{prediction boxes}}{\text{total number of prediction boxes}} \quad (1)$$

The mean average precision (mAP) is calculated by taking the mean Average Precision (AP) over all classes for the selected IoU threshold, denoted in Equation 2. The mean AP represents the area under the precision-recall curve, while k stands for number of classes.

$$mAP = \frac{\sum_{i=1}^k AP_i}{k} \quad (2)$$

The Frames Per Second (FPS) metric, as in Equation 3, is used to express how fast the model can process the input in one second. The *Number of Frames* represents the number of processed images, while *Total detection time* is a time frame of usually 1 second.

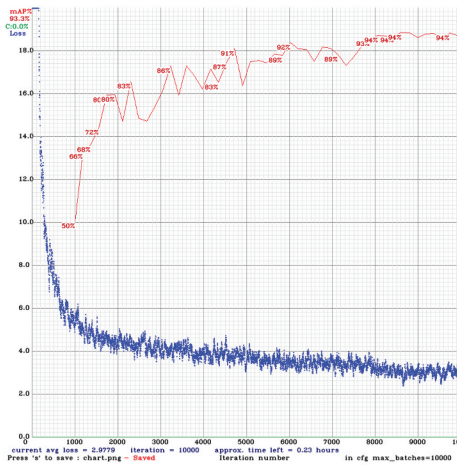
$$FPS = \frac{\text{Number of Frames}}{\text{Total detection time}} \quad (3)$$

Next, we present numerical results for tested object detector models on our dataset.

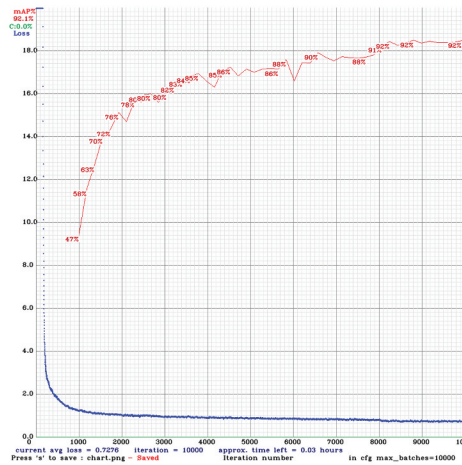
Results and Discussions

Next, we present underwater object detection results for different implementations of YOLOv4 and Faster RCNN models. Namely, we compare the obtained results for YOLOv4, YOLOv4 Tiny, CSP-YOLOv4, YOLOv4@ResNet50, and YOLOv4@DenseNet201, as well as for the Faster RCNN object detection. The training and testing of the model were conducted on the same custom dataset.

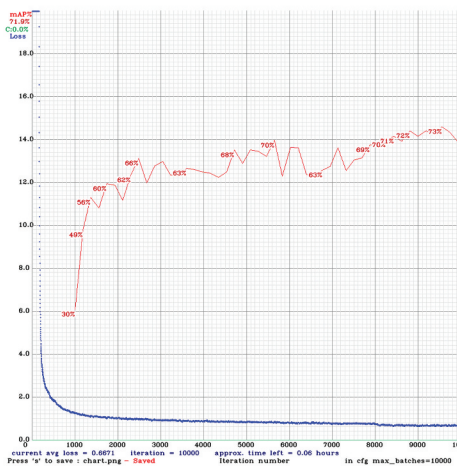
First, we will discuss the training process and loss charts. The losses in each batch were calculated from the log file generated during the training phase, where [Figure 8\(a\)](#) shows the loss and mAP plotted against iteration for the YOLOv4 model. The loss decreases, and mAP increases with iterations. The network can be further trained until the average loss decreases below 0.2, and the final loss expectation is 0. The YOLOv4 model started to converge with a good performance at about the 7500th iteration and having a stagnant performance at about the 10000th iteration. It took 23.7 hours to complete the training. [Figure 8\(b\)](#) presents the loss and mAP graph for the YOLOv4-Tiny@16 model, which shows impressive results of average loss below 0.8. Training of the YOLOv4-Tiny@16 method lasted 3.2 hours. The model started



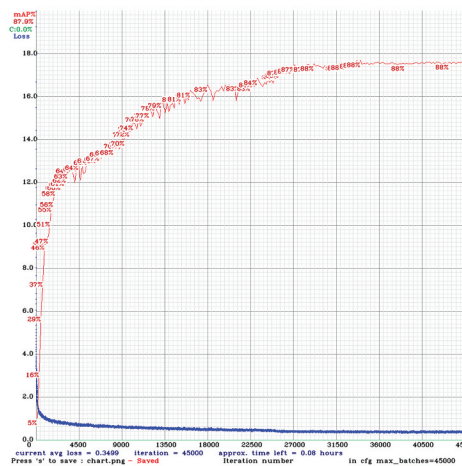
(a) YOLOv4



(b) YOLOv4-Tiny@16



(c) YOLOv4-Tiny@64



(d) YOLOv4@DenseNet201

Figure 8. Loss function and mAP performance.

to converge after some 8000 iterations. **Figure 8(c)** shows the loss and mAP graph for the YOLOv4-Tiny@64 model with a great average loss below 0.7 and poor mAP metrics. Poor results for the YOLOv4-Tiny@64 show that the larger minibatch sent to the GPU processor is better for 416×416 image size. With subdivision set to 16, a better generalization of the problem is obtained. Training of the YOLOv4-Tiny@64 model lasted 6.4 hours, while it started to converge at about the 8500th iteration. Finally, **Figure 8(d)** shows the loss and mAP graph for YOLO@DenseNet201. A deep network with numerous training epochs of 50,000 resulted in a long training time that lasted some 65.1 hours. The same number of training epochs were used for the YOLOv4@Resnet50, with training lasting 58.8 hours. Those two deep models started to converge at about the 37500th iteration.

Table 3. Performance evaluation result.

Architecture	Accuracy					mAP(%)
	<i>pipeline</i>	<i>leakage point</i>	<i>concrete weight</i>	<i>concrete mat</i>	<i>pipe coupling</i>	
YOLOv4	98.53	81.83	94.62	96.00	96.40	94.21
YOLOv4-Tiny@64	54.03	66.78	76.07	83.09	79.39	72.97
YOLOv4-Tiny@16	86.84	81.27	96.05	98.78	97.46	92.43
CSP-YOLOv4	91.04	84.37	97.85	98.80	85.03	93.97
YOLOv4 @ResNet50	78.05	34.92	84.70	95.48	89.38	79.50
YOLOv4 @DenseNet201	91.26	63.83	90.69	97.92	95.65	88.40
Faster RCNN	65.48	45.99	72.37	80.45	76.10	70.49

The object detection results obtained by the trained models on our custom underwater pipeline image dataset are as follows. In general, excellent results had been achieved for YOLO object detectors, as shown in Table 3. Table 3 presents results achieved by each trained object detecting method in terms of mean average precision (mAP) and accuracy for each target class. The mAP is an often-used metric that calculates average precision for each class across varied Intersection over Union (IoU). In this study, the threshold was set to 0.5, and it was shown that the YOLOv4 delivered the best mAP result of 94.21% on the tested dataset. This mAP result proves the superiority of the CSPDarknet53 backbone compared to other competitive methods. It should also be noted that the YOLOv4-Tiny@16 model showed high classification efficiency, achieving the mAP of 92.43%, accomplished for a short training time of only some 3.2 hours. As expected, the obtained results show that the deeper model architectures deliver higher mAP.

The underwater object detection was tested on both images and videos. The trained neural networks detect targets in given images and display bounding boxes around the detected object. Different implementations of the YOLOv4 prove its ability to be trained and detect objects in underwater environments. This can be seen in Figure 9 showing detecting performances of all tested models. Deeper models, such as YOLOv4, CSP YOLOv4, and YOLOv4@DenseNet201, reveal the benefits of the architecture and prove that better generalization of the problem is ensured by a larger minibatch sent to the GPU processor.

The processing speed should also be highlighted in addition to classification and detection performances. Table 4 shows the processing speed performance of each tested model required to classify the input images in the test dataset correctly. The obtained result confirmed that the networks, especially YOLOv4 and YOLOv4-Tiny, could simultaneously detect target classes in real-time while the video is playing. Here, we should emphasize the difference in FPS performance for YOLOv4-Tiny compared to other implementations. Namely, the tiny model could achieve high FPS due to the model's small size (shallow network), resulting in faster inference speed. The detection rate of Faster RCNN was not taken into account since it is a two-stage object detector, and thus, it is not intended for real-time object detection.

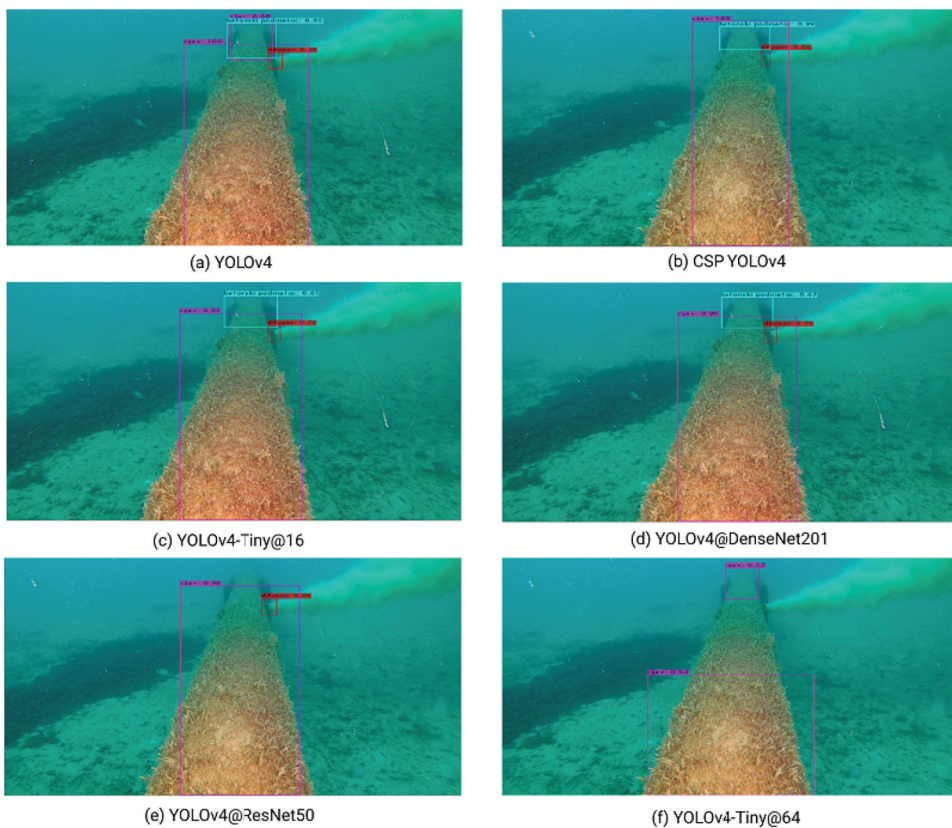


Figure 9. Detection performance of YOLO models.

Table 4. Processing speed inference.

Architecture	Frame Per Second(FPS)
YOLOv4	25
YOLOv4-Tiny	38
CSP YOLOv4	15
YOLOv4@ResNet50	17
YOLOv4@DenseNet201	16

Comparing our obtained results in underwater pipeline object detection to other underwater object detection studies found in literature (such as Chen et al. 2021; Hu et al. 2021; Rosli et al. 2021; Tian et al. 2021; Zhang et al. 2021), we can conclude that our obtained result of mAP 94.21% is quite remarkable. Please note that the underwater environment is rather challenging, and the above-referred papers detect different underwater objects (like, for example, fish and not the underwater pipeline objects), often achieving a smaller mAP. A similar problem of underwater pipeline detection and its component is done by Zhao, Wang, and Du (2020), distinguishing only between two target classes (while our research dealt with five underwater target classes). Comparison of detection results from different research is not possible due to a lack of public

datasets. We could compare results obtained from the trained model on the custom dataset with the YOLOv4 model trained only on the COCO dataset. The YOLOv4 trained only on the COCO dataset without fine-tuning on our dataset cannot detect any target class. We can conclude that the transfer done in this research learning were successful. As proof, results were compared with the YOLOv4 model trained from scratch. That model achieved a smaller mAP of 90.98%.

To conclude, our study investigates the performances of seven deep-learning architectures for pipeline component detection from images in challenging underwater environments, achieving remarkable mAP of up to 94.21% on a custom dataset. As a suggestion for future work, a more challenging dataset should be obtained, containing different underwater conditions. Another applicable aim of the project could be detection of pipeline failures. Also, the study can be extended to inside pipeline detections if a dataset is acquired.

Conclusion

In this paper, different implementations of the YOLOv4 were trained and tested on a same custom underwater image dataset to investigate the model's robustness in detecting pipeline objects in demanding underwater scenarios. Detection results of YOLOv4, YOLOv4-Tiny, CSP-YOLOv4, YOLOv4@ResNet, and YOLOv4@DenseNet were compared on test dataset. Further, achieved detection results were compared to the two-stage object detector Faster RCNN. The study was focused on detecting five object classes in the different subsea environments from different camera angles. The YOLOv4 method outperformed other competitive methods in terms of mAP (achieving mAP of 94.21%), with YOLOv4-Tiny achieving the highest FPS and high mAP of 92.43%. In comparison to other similar methods, our method gives promising results dealing with the problem of underwater pipeline detection. This research could be used in the future by autonomous underwater vehicles (AUVs) and remotely operated vehicles (ROVs) to inspect underwater pipelines. In order to achieve additionally improved performance metrics, it is possible to use various image enhancement methods for improving the quality of the underwater imagery dataset.

Acknowledgment

Author would like to thank the company Vectrino d.o.o. (Boze Milanovica 2b, 51000 Rijeka, Croatia, www.vectrino.hr) for providing us dataset used in the paper. This research was fully supported by the Croatian Science Foundation under the project IP-2018-01-3739, EU Horizon project "INNO2MARE: Strengthening the Capacity for Excellence of Slovenian and

Croatian Innovation Ecosystems to Support the Digital and Green Transitions of Maritime Regions” (101087348), EU Horizon 2020 project ”National Competence Centres in the Framework of EuroHPC (EUROCC)”, IRI2 project”ABsistemDCiCloud” (KK.01.2.1.02.0179), University of Rijeka projects uniri-tehnic-18-17 and uniri-tehnic-18-15, and Croatian–Slovenian bilateral project BI-HR/20-21-043.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Boris Gašparović  <http://orcid.org/0000-0002-6052-8785>

Jonatan Lerga  <http://orcid.org/0000-0002-4058-8449>

Goran Mauša  <http://orcid.org/0000-0002-0643-4577>

Marina Ivašić-Kos  <http://orcid.org/0000-0002-1940-5089>

References

- Asyraf, M. S., I. S. Isa, M. I. F. Marzuki, S. N. Sulaiman, and C. C. Hung. 2021. CNN-based YOLOv3 comparison for underwater object detection. *Journal of Electrical and Electronic Systems Research (JEESR)* 18 (APR2021):30–37. doi:10.24191/jeesr.v18i1.005.
- Bochkovskiy, A., C.-Y. Wang, and H.-Y.M Liao. 2020. “Yolov4: Optimal speed and accuracy of object detection.” *arXiv preprint arXiv:2004.10934*.
- Chen, L., M. Zheng, S. Duan, W. Luo, and L. Yao. 2021. Underwater target recognition based on improved YOLOv4 neural network. *Electronics* 10 (14):1634. doi:10.3390/electronics10141634.
- Girshick, R. 2015. “Fast r-cnn.” In *Proceedings of the IEEE international conference on computer vision*, Santiago, Chile, 1440–48. <https://arxiv.org/abs/1504.08083>
- Girshick, R., J. Donahue, T. Darrell, and J. Malik. 2014. “Rich feature hierarchies for accurate object detection and semantic segmentation.” In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Columbus, USA, 580–87 <https://arxiv.org/abs/1311.2524>
- He, K., X. Zhang, S. Ren, and J. Sun. 2015. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37 (9):1904–16.
- He, K., X. Zhang, S. Ren, and J. Sun. 2016. “Deep residual learning for image recognition.” In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, USA, 770–78. <https://arxiv.org/abs/1512.03385>
- Hu, X., Y. Liu, Z. Zhao, J. Liu, X. Yang, C. Sun, S. Chen, B. Li, and C. Zhou. 2021. Real-time detection of uneaten feed pellets in underwater images for aquaculture using an improved YOLO-V4 network. *Computers and Electronics in Agriculture* 185:106135. doi:10.1016/j.compag.2021.106135.
- Jacobi, M., and D. Karimanzira. 2013. “Underwater pipeline and cable inspection using autonomous underwater vehicles.” In *2013 MTS/IEEE OCEANS- Bergen: Bergen, Norway*, IEEE, 1–6. <https://doi.org/10.1109/OCEANS-Bergen.2013.6608089>

- Jiang, H., and E. Learned-Miller. 2017. "Face detection with the faster R-CNN." In *2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017)*, Washington, USA: IEEE, 650–57. <https://arxiv.org/abs/5201606.03473>
- Lei, F., H. Zhu, F. Tang, and X. Wang. 2022. Drowning behavior detection in swimming pool based on deep learning. *Signal, Image and Video Processing* 16 (6):1–8. doi:10.1007/s11760-021-02124-9.
- Lin, T.-Y., M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. 2014. "Microsoft coco: Common objects in context." In *European conference on computer vision*: Zurich, Switzerland, Springer, 740–55. <https://arxiv.org/abs/1405.0312>
- Liu, S., L. Qi, H. Qin, J. Shi, and J. Jia. 2018. "Path aggregation network for instance segmentation." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Salt Lake City, USA, 8759–68. <https://arxiv.org/abs/1803.01534>
- Lu, H., Y. Li, Y. Zhang, M. Chen, S. Serikawa, and H. Kim. 2017. Underwater optical image processing: A comprehensive review. *Mobile Networks and Applications* 22 (6):1204–11. doi:10.1007/s11036-017-0863-4.
- Moniruzzaman, M., S. M. S. Islam, P. Lavery, and M. Bennamoun. 2019. "Faster r-cnn based deep learning for seagrass detection from underwater digital images." In *2019 Digital Image Computing: Techniques and Applications (DICTA)*, Perth, Australia: IEEE, 1–7. <https://doi.org/10.1109/DICTA47822.2019.8946048>
- Narimani, M., S. Nazem, and M. Loueipour. 2009. "Robotics vision-based system for an underwater pipeline and cable tracker." In *Oceans 2009-Europe*, Bremen, Germany: IEEE, 1–6. <https://doi.org/10.1109/OCEANSE.2009.5278327>
- Raza, K., and S. Hong. 2020. Fast and accurate fish detection design with improved YOLO-v3 model and transfer learning. *International Journal of Advanced Computer Science and Applications* 11 (2):7–16. doi:10.14569/IJACSA.2020.0110202.
- Redmon, J. 2013–2016. *Darknet: Open Source Neural Networks in C*. <http://pjreddie.com/darknet/>.
- Redmon, J., S. Divvala, R. Girshick, and A. Farhadi. 2016. "You only look once: Unified, real-time object detection." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, USA, 779–88. <https://arxiv.org/abs/1506.02640>
- Redmon, J., and A. Farhadi. 2017. "YOLO9000: Better, faster, stronger." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Honolulu, USA, 7263–71. <https://arxiv.org/abs/1612.0824>
- Redmon, J., and A. Farhadi. 2018. "Yolov3: An incremental improvement." *arXiv preprint arXiv:1804.02767*.
- Ren, S., K. He, R. Girshick, and J. Sun. 2015. Faster r-cnn: Towards realtime object detection with region proposal networks. *Advances in Neural Information Processing Systems* 28:91–99.
- Rosli, M. S. A. B., I. S. Isa, M. I. F. Maruzuki, S. N. Sulaiman, and I. Ahmad. 2021. "Underwater animal detection using YOLOV4." In *2021 11th IEEE International Conference on Control System, Computing and Engineering (ICCSC)*, Penang, Malaysia, IEEE, 158–63. <https://doi.org/10.1109/ICCSC52189.2021.5609530877>
- Schettini, R., and S. Corchs. 2010. Underwater image processing: State of the art of restoration and image enhancement methods. *EURASIP Journal on Advances in Signal Processing* 2010 (1):1–14. doi:10.1155/2010/746052.
- Shatnawi, A., G. Al-Bdour, R. Al-Qurran, and M. Al-Ayyoub. 2018. "A comparative study of open source deep learning frameworks." In *2018 9th international conference on information and communication systems (ICICS)*, Irbid, Jordan, IEEE, 72–77. <https://doi.org/10.1109/IACS.2018.8355444>

- Sung, M., S.-C. Yu, and Y. Girdhar. 2017. "Vision based real-time fish detection using convolutional neural network." In *OCEANS 2017-Aberdeen*, IEEE, 1–6. <https://doi.org/10.1109/OCEANSE.2017.8084889>
- Tian, M., X. Li, S. Kong, J. Yu, et al. 2021. "Pruning-Based YOLOv4 algorithm for underwater garbage detection." In *2021 40th Chinese Control Conference (CCC)*, Shanghai, China, IEEE, 4008–13. <https://doi.org/10.23919/CCC52363.2021.9550592>
- Uplavikar, P. M., Z. Wu, and Z. Wang. 2019. "All-in-One underwater image enhancement using domain-adversarial learning." In *CVPR Workshops*, Long Beach, USA, 1–8. <https://doi.org/10.48550/580ARXIV.1905.13342>
- Wang, C.-Y., A. Bochkovskiy, and H.-Y.M Liao. 2021. "Scaled-yolov4: Scaling cross stage partial network." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, USA, 13029–38. <https://doi.org/10.48550/ARXIV.2011.08036>
- Wang, C.-Y., H.-Y.M Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh. 2020. "Cspnet: A new backbone that can enhance learning capability of CNN." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, Seattle, USA., 390–91. <https://doi.org/10.48550/ARXIV.1911.11929>
- Wu, Y., A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. 2019. *Detectron2*. <https://github.com/facebookresearch/detectron2>.
- Xu, W., and S. Matzner. 2018. "Underwater fish detection using deep learning for water power applications." In *2018 International conference on computational science and computational intelligence (CSCI)*. Las Vegas, USA: IEEE, 313–18. <https://doi.org/10.48550/ARXIV.1811.01494>
- "YOLO-Label". 2019. *developer0hye.github.io*. Last accessed 11 January. 2022. <https://github.com/developer0hye/YoloLabel>.
- Yun, S., D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. 2019. "Cutmix: Regularization strategy to train strong classifiers with localizable features." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. Seoul, South Korea, 6023–32. <https://doi.org/10.48550/ARXIV.1905.04899>
- Zhang, M., S. Xu, W. Song, Q. He, and Q. Wei. 2021. Lightweight underwater object detection based on YOLO v4 and multi-scale attentional feature fusion. *Remote Sensing* 13 (22):4706. doi:10.3390/rs13224706.
- Zhao, X., W. Li, Y. Zhang, T. A. Gulliver, S. Chang, and Z. Feng. 2016. "A faster RCNN-based pedestrian detection system." In *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*. Montreal, Canada:IEEE, 1–5. <https://doi.org/10.1109/VTCFall.2016.7880852>
- Zhao, X., X. Wang, and Z. Du. 2020. "Research on detection method for the leakage of underwater pipeline by YOLOv3." In *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*. Beijing, China: IEEE, 637–42. <https://doi.org/10.1109/ICMA49215.2020.9233693>
- Zhao, Z.-Q., P. Zheng, S.-T. Xu, and X. Wu. 2019. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems* 30 (11):3212–32. doi:10.1109/TNNLS.2018.2876865.