JACOBS UNIVERSITY

Deep Learning for Detecting Amphoras in Ancient Shipwrecks

by

Tianyao Chen

Bachelor Thesis in Computer Science

Submission: May 14, 2021 Supervisor: Prof. Dr. Andreas Birk

Jacobs University Bremen | Department of Computer Science and Electrical Engineering

Statutory Declaration

Family Name, Given/First Name	Chen, Tianyao
Matriculation number	30001974
Kind of thesis submitted	Bachelor Thesis

English: Declaration of Authorship

I hereby declare that the thesis submitted was created and written solely by myself without any external support. Any sources, direct or indirect, are marked as such. I am aware of the fact that the contents of the thesis in digital form may be revised with regard to usage of unauthorized aid as well as whether the whole or parts of it may be identified as plagiarism. I do agree my work to be entered into a database for it to be compared with existing sources, where it will remain in order to enable further comparisons with future theses. This does not grant any rights of reproduction and usage, however.

This document was neither presented to any other examination board nor has it been published.

German: Erklärung der Autorenschaft (Urheberschaft)

Ich erkläre hiermit, dass die vorliegende Arbeit ohne fremde Hilfe ausschließlich von mir erstellt und geschrieben worden ist. Jedwede verwendeten Quellen, direkter oder indirekter Art, sind als solche kenntlich gemacht worden. Mir ist die Tatsache bewusst, dass der Inhalt der Thesis in digitaler Form geprüft werden kann im Hinblick darauf, ob es sich ganz oder in Teilen um ein Plagiat handelt. Ich bin damit einverstanden, dass meine Arbeit in einer Datenbank eingegeben werden kann, um mit bereits bestehenden Quellen verglichen zu werden und dort auch verbleibt, um mit zukünftigen Arbeiten verglichen werden zu können. Dies berechtigt jedoch nicht zur Verwendung oder Vervielfältigung.

Diese Arbeit wurde noch keiner anderen Prüfungsbehörde vorgelegt noch wurde sie bisher veröffentlicht.

. . . .

Date, Signature

Abstract

Amphoras have great archaeological significance and are often found in ancient shipwrecks. The increasingly abundant visual data and the advent of more advanced deep learning algorithms motivated us to automate the detection of underwater amphoras. We trained a convolutional neural network (CNN) based model called Single Shot Detector (SSD) with a 50-layer Residual Network (ResNet-50) backbone on a very small dataset of only 50 images. And we still managed to achieve a 0.238 MS COCO mean average precision (mAP_{coco}) and a 0.503 Pascal VOC $mAP@0.5$. Underwater amphora detection is a challenging task. We believe future studies can achieve better results by increasing the size and resolution of the dataset, training bleeding-edge object detectors like YOLOv5, and defining two separate classes for the head and the body of amphoras. The code and the trained model are available at https://github.com/tillchen/amphora_object_detection.

Contents

1 Introduction

1.1 Motivation

1.1.1 Amphoras

The name *amphora* is derived from the Greek word *amphoreus*, which means "two-handled" [1, 2, 3]. It is the combination of two linguistic roots: *amphi* (on both sides) and *phoreus* (bearer) [1, 2, 3]. The Greek and Roman empires used amphoras (or amphorae) from 1500 B.C.E. to 500 C.E. to ship products [2, 4].

Amphoras were designed to hold large quantities of liquid (wine, oil, and water) and dry products (grain, preserved fish, and nuts) [2, 5, 6]. Like many units of measurement named after the packages, amphoras were also a semi-standard unit for liquid [2]. The structurally solid egg-like shape and the high volume-to-weight ratio made amphoras very efficient containers [2, 4].

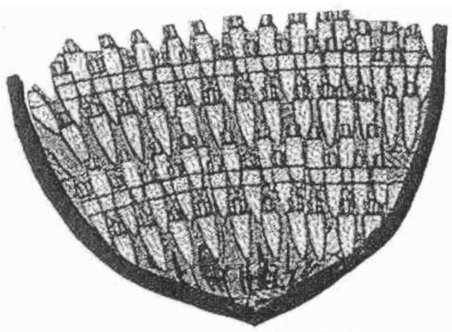

Figure 1: The egg-like shape allowed amphoras to interlock and minimize the waste of space on a ship. Source: [2].

Amphoras' various shapes and stamps - which changed by time, region, producer, contents, and brand identity - were used to identify the package status and the different products inside [2, 4].

Figure 2: Amphoras have various shapes. Source: [4].

Amphoras have great significance in archaeology. They can be used as evidence for the trade patterns throughout the Mediterranean [2]. As they have been found at trade destinations and shipwrecks, archaeologists use them to recreate the transit routes [2]. Furthermore, researchers can date ruins and shipwrecks by classifying different amphoras [2, 7].

1.1.2 Computer Vision for Underwater Archaeology

Computer vision is the science of perceiving and understanding the world through images and videos [8]. There have been multiple exciting applications of computer vision, including image classification [9], object detection and localization [10, 11], art generation (neural style transfer) [12], image creation with Generative Artificial Networks (GAN) [13], face recognition [14], human pose estimation [15], action and activity recognition [16], and image recommendation system [17].

However, there is still limited research for the usage of computer vision and machine learning in archaeology, especially underwater archaeology, compared to other domains [18, 19]. Instead of visual inspection, researchers can use computer vision to automate the detection, assessment, and classification of artifacts [18].

Underwater computer vision has proven to be challenging, mainly due to: 1) the distortion and attenuation caused by light propagation in water, 2) the free natural environment with the abundance of marine life and suspended particles, and 3) underwater objects are often broken [19, 20, 21, 22].

Despite the challenges, computer vision has lower costs [20] compared to sonar imagery [23] and laser scanning [24]. Plus, the increasingly abundant visual data obtained through unmanned underwater vehicles (UUVs), autonomous underwater vehicles (AUVs) [21, 25], and seafloor cabled observatories [19] motivate us to utilize deep learning.

The study of deep-water shipwrecks is in high demand as more and more threats to these sites are emerging [26]. Some new trawling methods have been destroying the surface of these sites [26]. Thus, it is crucial to implement efficient, accessible, and accurate techniques like deep learning based computer vision to study deep-water shipwrecks.

1.2 Deep Learning

Machine learning is the class of algorithms that allow computers to learn and improve from data instead of being explicitly programmed [27, 28]. And deep learning is the subfield of machine learning that builds artificial neural networks with more than one layer between the input and output layers [28, 29, 30]. Deep learning constructs complex representations by combining simpler ones from the previous layers [31].

1.2.1 Artifical Neural Networks (ANNs)

Inspired by the biological neuron, artificial neural networks (ANNs) were first introduced in 1943 using propositional logic [32]. The neuron activates its binary output when the number of active binary inputs reaches the activation threshold, making it possible to build networks that can perform any logical proposition [28, 32].

Perceptron was introduced in 1957, and it is based on a different artificial neuron called threshold logic unit (TLU) [33]. The input values are real numbers. TLU computes the weighted sum of the input and then applies a step function like the Heaviside function $heaviside(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x \geq 0 \end{cases}$ [28, 33].

A TLU can perform linear binary classification, whereas a layer of TLUs plus a bias neuron form a Perceptron capable of multi-output classification [28].

The output of a Perceptron is computed as follows, where $\mathbf{X}, \mathbf{W}, \mathbf{b}$, and ϕ are respectively the input matrix, weight matrix, bias vector, and activation function:

$$h_{\mathbf{W},\mathbf{b}}(\mathbf{X}) = \phi(\mathbf{X}\mathbf{W} + \mathbf{b})$$

Perceptron is trained using a variant of the Herbb's rule [34], which is famously summarized as "neurons wire together if they fire together" [35]. However, Perceptron can not learn complex patterns due to its linear decision boundary, and it does not output a class probability [28]. Thus, Multilayer Perceptron (MLP) was introduced by stacking multiple Perceptrons.

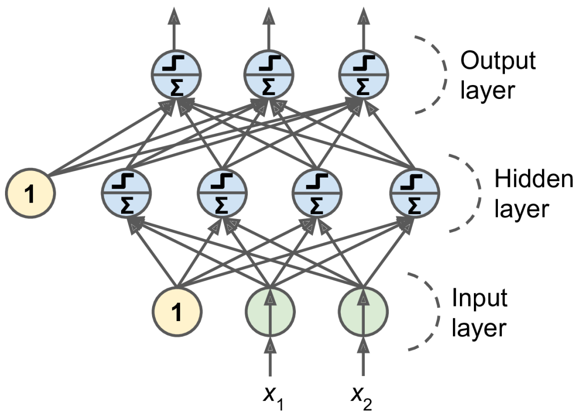

Figure 3: A Multilayer Perceptron with one hidden layer. Source: [28].

Backpropagation [36] is used to train the MLP, which first computes the loss, then measures each connection's loss contribution, and finally reduces the loss by adjusting the weights in the Gradient Descent [37] step. Activation functions like Rectified Linear Unit $ReLU(x) = max(0, x)$ are adopted to add nonlinearity, which gives deep neural networks the theoretical ability to approximate any continuous function [28].

1.2.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) [38] were inspired by the visual cortex, and they have been used in computer vision since the 1980s [28]. We can not simply use a deep neural network with fully connected layers for computer vision, as it breaks down for large images due to the immense number of parameters [28]. CNNs also have successful applications in other domains like recommender systems [39] and natural language processing (NLP) [40].

Hubel et al. [41, 42, 43] found that many biological neurons have small receptive fields, which means they only fire if a finite region of the visual field is stimulated. Some neurons only react to specific patterns, while the others with larger receptive fields respond to more complex patterns.

The filters or convolution kernels, which are learned during training, are neurons' weights presented as small images the size of receptive fields [28]. A layer of neurons with the same filter outputs a feature map, highlighting the parts of an image that maximize the activation of the filter [28].

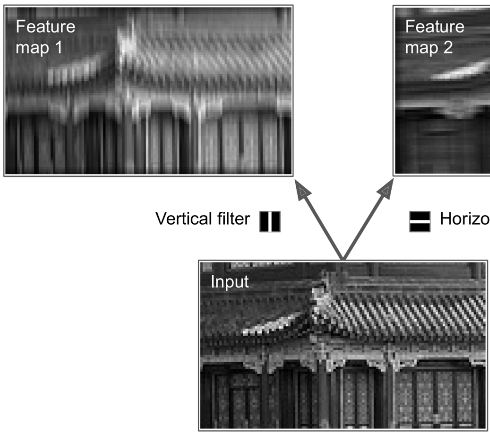

Figure 4: Two feature maps by applying two different filters. Source: [28].

The pooling layers subsample (i.e. shrink) the input image to make the network less computationally intensive [28]. Plus, pooling layers can bring some invariance to small translations, rotations, and scaling [28].

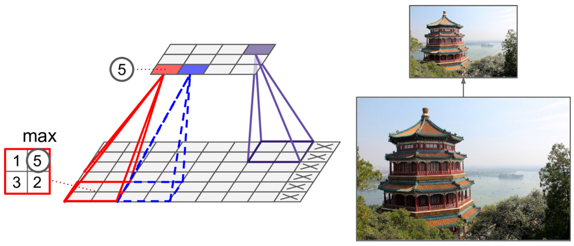

Figure 5: A max pooling layer with 2 x 2 as the kernel size and two as the stride. The max value from the receptive field passes to the next layer. Source: [28].

The typical CNN architecture involves the aforementioned convolutional layers, pooling layers, and fully connected layers. Some well-established CNN architectures are LeNet-5 [44], AlexNet [45], GoogLeNet [46], VGGNet [47], ResNet (Residual Network) [48], Xception (Extreme Inception) [49], MobileNet [50, 51], and SENet (Squeeze-and-Excitation Network) [52].

ResNet is the backbone network of the model used in this paper. ResNet's skip connections help speed up the training considerably, since: 1) the network preconditions the problem to be the identity function, which is often close to the target function, and 2) the network can start making progress even if some layers have not started learning yet [28, 48]. Batch normalization [53] is used after each convolution to reduce the vanishing gradient problem [54] and the need for other regularization techniques like dropout [55].

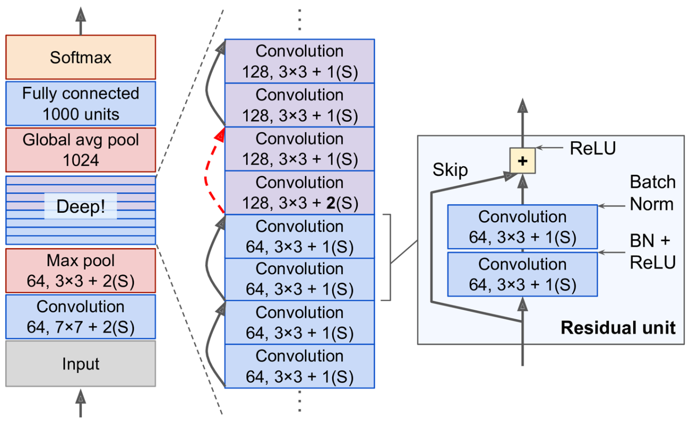

Figure 6: ResNet architecture. Source: [28].

1.3 Deep Learning vs. Traditional Computer Vision

The improvements of deep learning algorithms, computing power, image resolution, and visual data size have enabled deep learning to achieve state-of-the-art performance in many computer vision tasks, including image classification, object detection, and semantic segmentation [19, 56, 57].

Traditionally, computer vision requires the manual feature extraction step, which relies on domain knowledge to produce high-quality features [8, 10, 57]. These handcrafted features are further processed by a machine learning classifier like a support vector machine (SVM) [8, 10, 57]. However, manual feature extraction becomes more and more complex as the number of classes grows [57]. Besides, conventional machine learning algorithms' generalization ability saturates quickly as the size of training data expands [19].

Deep learning removes the manual feature extraction step and makes training end-to-end [8, 57]. Thus, deep learning requires less domain knowledge, and it is more flexible as we can re-train the models on custom datasets for specific tasks [57].

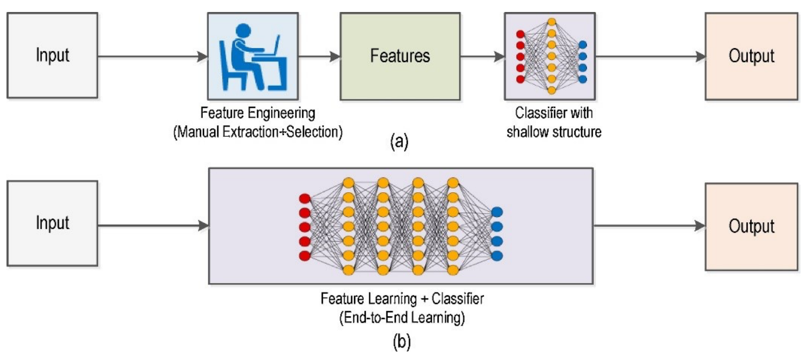

Figure 7: Traditional computer vision (a) vs. deep learning (b). Source: [57].

However, deep learning's performance depends on obtaining large datasets with high image resolution [57]. Some popular public datasets like PASCAL Visual Object Classes (VOC) [58], ImageNet [59], and Microsoft Common Objects in Context (COCO) [60] have respectively 11 thousand, 14 million, and 328 thousand images [11]. Having a deficient dataset for a task may cause deep learning to have inferior results to traditional computer vision.

1.4 Object Detection

Object detection is the computer vision task that localizes and classifies objects in an image [8, 10, 11, 28]. Object detection is one of the most challenging tasks in computer vision, as it can be considered both a regression task for bounding box prediction and a classification task [8, 28, 61]. Plus, the significant variations in viewpoints, poses, occlusions, and lighting conditions add extra difficulties to perform perfect object detection [10, 11].

The traditional sliding window approach trains a CNN to classify and locate a single object and then slides it across the image [28, 61, 62, 63]. This approach slides the CNN multiple times with various window sizes to detect objects at different scales, which causes it to be relatively slow [28].

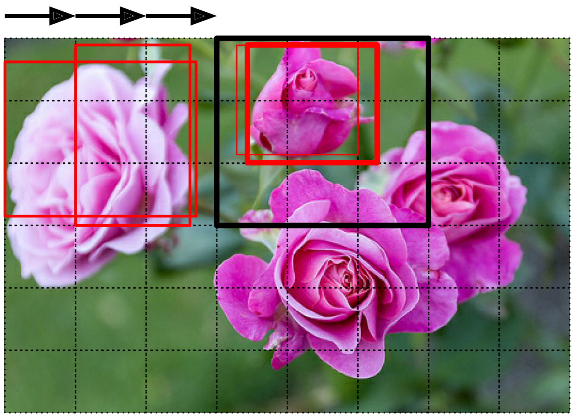

Figure 8: The sliding window approach for object detection. Source: [57].

Luckily, many novel object detectors have been introduced. Here we summarize three influential object detector families: Region-Based Convolutional Neural Networks (R-CNN) [61, 64, 65], Single Shot Detector (SSD) [66], and You Only Look Once (YOLO) [63, 67, 68, 69, 70].

1.4.1 General Object Detection Framework Components

Before we dive into specific object detectors, it's worth understanding the four high-level components of general object detection frameworks.

Region Proposal Region proposal finds regions of interest (RoIs) for further processing, which is achieved by discarding regions with a low objectness score. The objectness score indicates the probability of the region containing objects [71].

Backbone Network A pre-trained CNN is used for feature extraction and predictions. The bounding-box prediction is the tuple (x, y, w, h), respectively the coordinates of the center, the width, and the height [8, 28].

Non-Maximum Suppression (NMS) The backbone network typically produces multiple overlapping bounding boxes for one object, thus NMS finds the box with the maximum class probability and suppresses the rest [8]. The steps include [72]:

1. Sort the boxes based on the probability score.
2. Select the box with the maximum probability score.
3. Compute the overlap - intersection over union (IoU) - of the boxes and discard the ones with an IoU higher than the NMS threshold.
4. Repeat recursively on the remaining boxes.

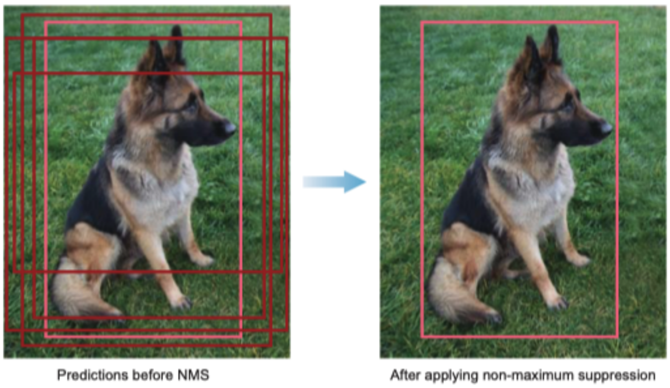

Figure 9: NMS. Source: [8].

Metrics The two primary metrics for object detection are frames per second (FPS) and mean average precision (mAP) [8, 11, 28, 73]. To understand mAP, we need to understand first the aforementioned intersection over union (IoU) and the precision-recall curve (PR curve). IoU is also known as the Jaccard index, which measures the similarity between two sets [73]. IoU can be mathematically formulated as follows [8, 73]:

$$IoU = \frac{B_{ground\ truth} \cap B_{prediction}}{B_{ground\ truth} \cup B_{prediction}}$$

We say that a prediction is a true positive (TP) if the classification is correct and the IoU is higher than the NMS threshold; otherwise, it is a false positive (FP) [8, 11, 73]. A false negative (FN) is a ground truth that does not have a prediction [73]. Precision and recall are defined as follows [29, 74]:

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

$$Recall = \frac{TP}{TP + FN}$$

There is a trade-off between precision and recall [8, 28, 29, 73]. We can obtain the average precision (AP) by drawing the PR curve and computing the area under the curve (AUC) [8, 73]. We then get the mAP by averaging the AP over all the classes [8, 28, 73]. The traditional Pascal VOC metric uses $mAP@0.5$ with a 0.5 IoU threshold [11, 58]. The new COCO metric $mAP_{coco} = mAP@[0.50 : 0.05 : 0.95]$ is averaged over various IoU thresholds from 0.5 to 0.95 in steps of 0.05, which rewards detectors with better localization [11, 75]. Note that AP is sometimes used in the COCO metric to implicitly mean mAP [75], although we use mAP consistently in this paper. COCO also introduced mAP for different scales: mAP^{small} for $area < 32^2$, mAP^{medium} for $32^2 < area < 96^2$, and mAP^{big} for $area > 96^2$ [11, 75]

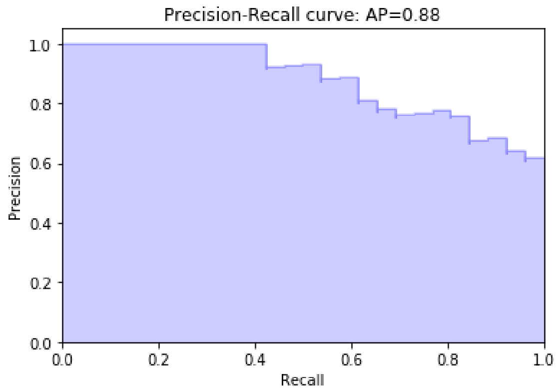

Figure 10: A PR curve with a 0.88 AUC/AP. Source: [73].

1.4.2 Region-Based Convolutional Neural Networks (R-CNN)

The evolution from the original R-CNN [61] to Fast R-CNN [64] and then to Faster R-CNN [65] builds up the R-CNN family.

R-CNN R-CNN has four components [8, 61]:

1. Region proposal with a greedy search algorithm called selective search, which finds RoIs by combining similar pixels into boxes.
2. A pre-trained CNN as the network backbone.
3. Classification with a linear SVM.
4. Localization with a bounding-box regressor.

R-CNN has the following disadvantages [8, 61, 64]:

- The FPS is very low. The selective search algorithm proposes about 2000 RoIs, which is computationally expensive as the CNN has to process each proposal separately.
- The training is multi-stage, inelegant, expensive, and not end-to-end. It involves training three components separately: the CNN, the SVM, and the bounding-box regressor.

Fast R-CNN Fast R-CNN made the following changes from R-CNN [8, 64]:

- The backbone CNN comes before region proposal so that the images only go through the CNN once instead of 2000 RoIs going through the CNN separately.
- Classification is performed by the softmax layer of the CNN instead of the SVM. And localization is also an output layer of the CNN.
- The RoI max pooling layer after region proposal reshapes the input size for the fully connected layers.

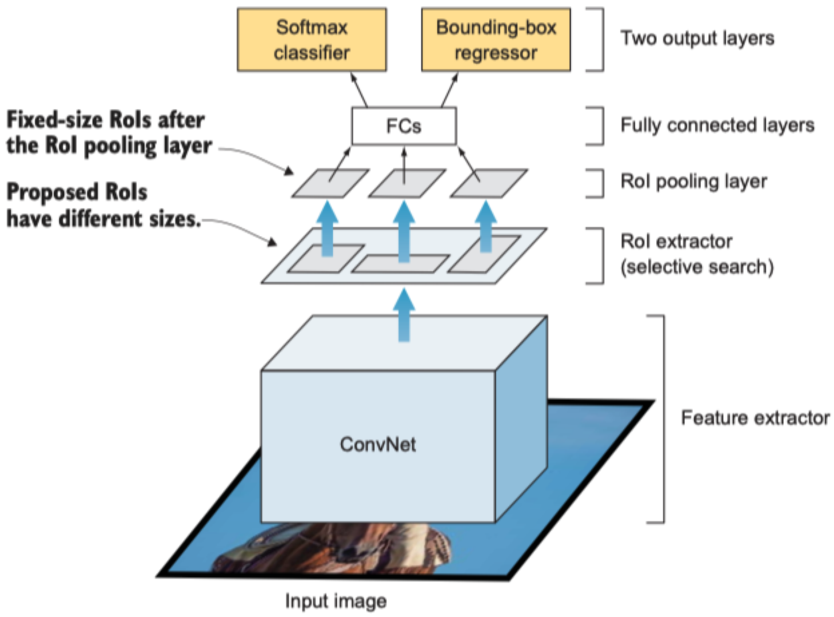

Figure 11: Fast R-CNN. Source: [8].

Fast R-CNN is significantly faster than R-CNN, although the selective search algorithm still exists as the bottleneck [8, 64, 65].

Faster R-CNN Faster R-CNN made the following improvements from Fast R-CNN [8, 65]:

- Region Proposal Network (RPN) or attention network replaced selective search, which reduced the number of proposals, speeded up the model, and made training end-to-end. RPN is a Fully Convolutional Network (FCN) [76] that outputs objectness scores and RoIs, and it can be used as a standalone network for single-class object detection. RPN adds minimal cost as the features are also shared with the detection network.
- Anchors were introduced as reference boxes at different scales and aspect ratios. Thus, the regression layer only needs to output the offsets of the coordinates, width, and height from the anchors. The anchors are created with the sliding-window approach. By default, nine anchors (three scales and three aspect ratios) are generated for each window.

To summarize, the R-CNN family are two-stage detectors that separate region proposal and detection [8, 11]. They are very computationally intensive and can not achieve real-

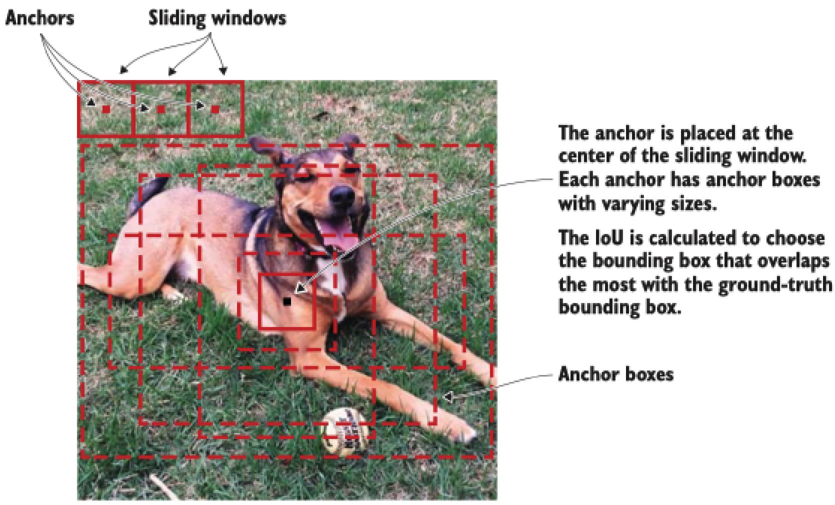

Figure 12: Anchors in Faster R-CNN. Source: [8].

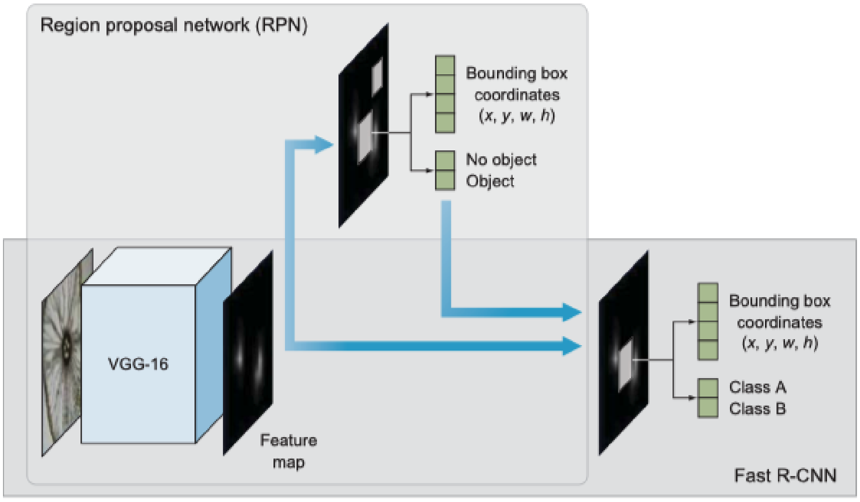

Figure 13: Faster R-CNN. Source: [8].

time detection [8, 66, 63]. One-stage detectors like Single Shot Detector (SSD) and You Only Look Once (YOLO) skip the region proposal to achieve real-time detection [8].

1.4.3 Single Shot Detector (SSD)

Single Shot Detector (SSD) makes both the objectness prediction and classification directly in one shot [8, 66]. It has three main components [8, 66]:

- The backbone network. It also uses anchors called priors like in Faster R-CNN. But the network sends the bounding box offsets and the class scores to NMS directly when it finds a box that contains the object features.
- Multi-scale feature layers, which are convolutional layers that decrease in size progressively to detect objects at various scales. The resolutions of feature maps drop as the CNN reduces the spatial dimension.

- NMS.

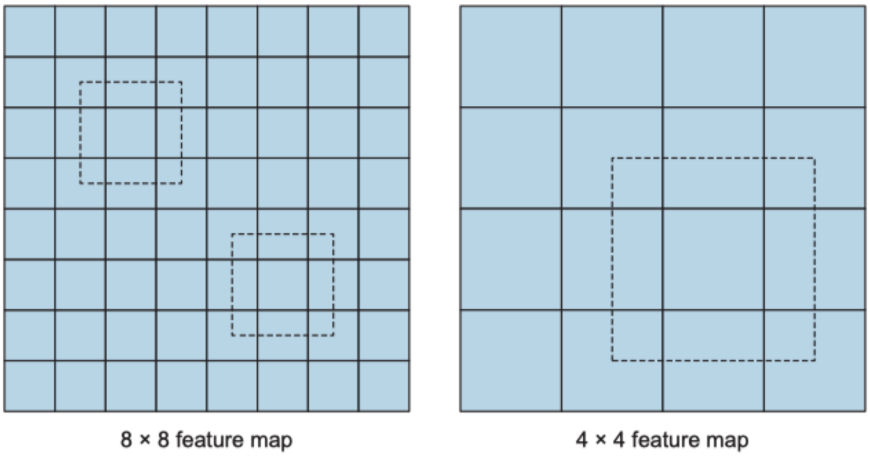

Figure 14: Multi-scale feature maps in SSD. Higher-resolution feature maps (left) detect smaller objects. Lower-resolution feature maps (right) detect bigger objects Source: [8].

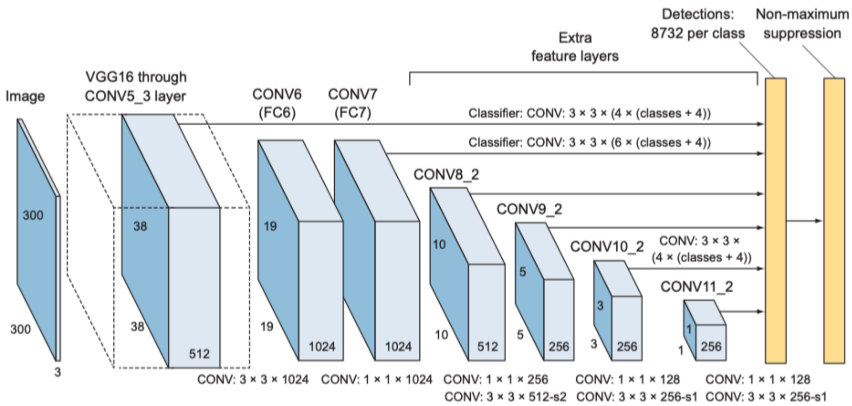

Figure 15: SSD. Source: [8].

1.4.4 You Only Look Once (YOLO)

YOLO is a one-stage real-time detector family similar to SSD. The YOLO family has also been going through a series of improvements since the original YOLO paper was published. YOLOv1 [63] introduced the general architecture; YOLOv2 [67] added anchors similar to Faster R-CNN and SSD; YOLOv3 [68] further refined the architecture.

YOLO divides the image into a grid, and the grid cell detects objects if the object's center is inside the cell [63]. The backbone network is called DarkNet, which is inspired by GoogLeNet [8, 63].

YOLOv4 [69], a bleeding-edge detector introduced in 2020, utilized numerous new features to improve the performance from YOLOv3, including DropBlock regularization [77], Mish activation [78], Self-Adversarial Training [69], etc.

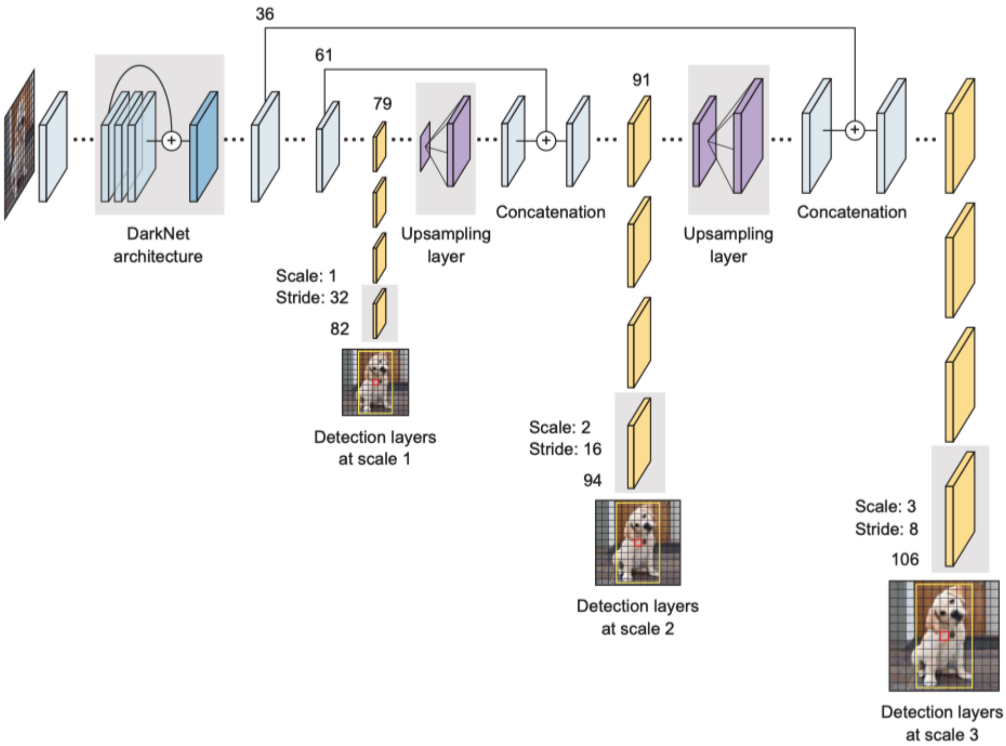

Figure 16: YOLOv3 architecture. YOLO performs detections at three different scales. Layer 79, 91, and 106 make grids of 13 x 13, 26 x 26, and 52 x 52 to detect large, medium, and small objects. Source: [8].

YOLOv5 [70] is under active development, and the authors have yet to publish a paper.

2 Related Work

As previously mentioned in section 1.1.2, there is limited research on deep learning for underwater archaeology and vision-based underwater object detection in general.

For fish detection, Qin et al. [19, 79], Zhang et al. [80], Villon et al. [81], Xu et al. [82], and Konovalov et al. [83] respectively used Fast R-CNN, a model similar to R-CNN, a sliding window with a CNN, YOLOv3, and Xception.

For crab detection, Cao et al. [84] proposed a detector called Faster MSSDLite, which is based on SSD with a MobileNetV2 backbone and Feature Pyramid Network (FPN) [85].

For planktons and corals, we only found research for classification but not object detection [19, 25].

For amphora detection, we only found the study from Pasquet et al. [22, 62] that used a sliding window with a CNN on a high-resolution orthophoto (i.e. aerial photo).

3 Data and Methods

3.1 Data

The dataset consists of 50 images (294 objects) in the training set and seven images (31 objects) in the validation set, which maintains a 90%: 10% ratio for the object count. Two additional images were used as the test set. All the images were obtained through various sources [86, 87, 88, 89, 90, 91, 92, 93, 94, 95] and were labeled with labelImg [96].

3.2 Model

Out of the three families of object detection frameworks we have discussed in section 1.4, we chose SSD due to the following reasons:

- Faster R-CNN is too computationally intensive and too slow for AUVs and UUVs.
- YOLOv5, the state-of-the-art model of the YOLO family, is under active development, and the authors have yet to publish a paper.

We used the TensorFlow Object Detection API [97, 98]. ssd_resnet_50_fpn_coco is the model name in the TensorFlow 1 Detection Model Zoo [99]. The backbone network is ResNet-50 (50-layer ResNet) (see section 1.2.2) instead of VGG-16 in the original SSD paper, as it improves mAP significantly [48]. Following the approach from Cao et al. [84] for crab detection, we adopted Feature Pyramid Network (FPN) to improve the detection at different scales [85].

3.3 Model Training

We used transfer learning [100] as the model was pre-trained on the COCO dataset, which means the weights and biases are restored instead of randomly initialized, and the model does not have to learn the low-level features from scratch [28]. num_classes, batch_size, and num_examples in eval_config were adjusted to one, eight, and seven respectively in the configuration file, while the rest of the fields were unchanged. The network reshaped the images to 640 x 640 with the fixed_shape_resizer. The activation function is ReLU6, a ReLU variant that caps the maximum value at six to encourage the model to learn sparse features earlier [101]. We used L_2 regularization [102] to reduce overfitting [103]. Data augmentation [45] - a method to artificially enrich the dataset by randomly flipping horizontally and cropping the images - was adopted as a regularization technique. We chose the momentum optimizer [104] as it converges faster than Stochastic Gradient Descent [105] by keeping accelerating to the optimum [28]. We used learning rate warmup and learning rate scheduling by cosine decay to speed up the convergence, which means the learning rate starts small, increases gradually, and then decreases after reaching the maximum [28, 106, 107, 108]. The model was trained on Google Colaboratory [109] with a GPU runtime, and the training took 2 hours and 40 minutes to reach the peak mAP result at step 8621.

4 Evaluation

The primary metric mAP_{coco} (see section 1.4.1) from our model is 0.238, while the traditional $mAP@0.5$ is 0.503. The documented mAP_{coco} in the TensorFlow 1 Detection Model Zoo for ssd_resnet_50_fpn_coco is 0.35, which was computed on the COCO validation set. Our model's performance is indeed lower than that in the model zoo. And it is somewhat difficult to perform a direct comparison with the result from Pasquet et al. [62], as they did not compute the mAP and only mentioned that they detected around 90.3 percent of amphoras. However, we can conclude that our model did not detect more than 90.3 percent based on the visual inspection of the validation images in figure 17.

Nevertheless, our model's performance is still impressive due to the following reasons:

- Our dataset is very limited. As previously mentioned in section 1.3, the COCO dataset has 328 thousand images compared to our 57 images plus two test images. For Pasquet et al., they split one 38000 x 15000 orthophoto into 400 x 400 images and used 25% of them for training, which means the training set alone was around 890 images.

- Our dataset is more diverse than that of Pasquet et al. Since our dataset images are obtained through numerous sources instead of from a single orthophoto, they reflect better the various shapes of amphoras (see figure 2). Plus, our dataset has both large and medium instances of amphoras that vary significantly in terms of scale. The diversity of our dataset made it harder to achieve a high mAP, although it encouraged the model to generalize better.

- Underwater object detection is more challenging than general object detection performed by the COCO dataset. As previously mentioned in section 1.1.2, the same challenges also apply to amphora detection. In fact, the undetected amphoras in figure 17 are almost all either broken, or partially buried in sand, or blocked by suspended particles. We only defined one class for all amphoras even though some of them are broken, while Pasquet et al. defined two separate classes for the head and the body to detect broken instances better.

- Our $mAP@0.5$ score is comparable with that from Xu et al. [82] for fish detection. They achieved a 0.5392 $mAP@0.5$ using YOLOv3, although they did not compute the mAP_{coco}.

We also ran the model on 2 test images shown in figure 18. To our surprise, the model managed to detect the amphora in the first image even though it is so broken that only half of it is present. As for the second image, none of the amphoras was detected. The second image's size is only 709 x 411, while it has hundreds of amphora instances. These instances are primarily small objects defined by the COCO metric, which our training set does not include. It is a known issue that small objects are significantly more difficult to detect than medium and large objects, mainly due to the low resolution and consequently the lack of features to learn [110, 111, 112]. Plus, the image can be described as a crowded and densely packed scene, which is also notoriously challenging for object detectors as the objects overlap with each other so strongly that it is even impossible for human experts to differentiate the instances [62, 113, 114, 115]. We tried to split the image and add parts of it to the training set. However, the inter-occlusion of the instances caused many labeling errors, and the resolution of each instance was so low that the training could not converge to a solution.

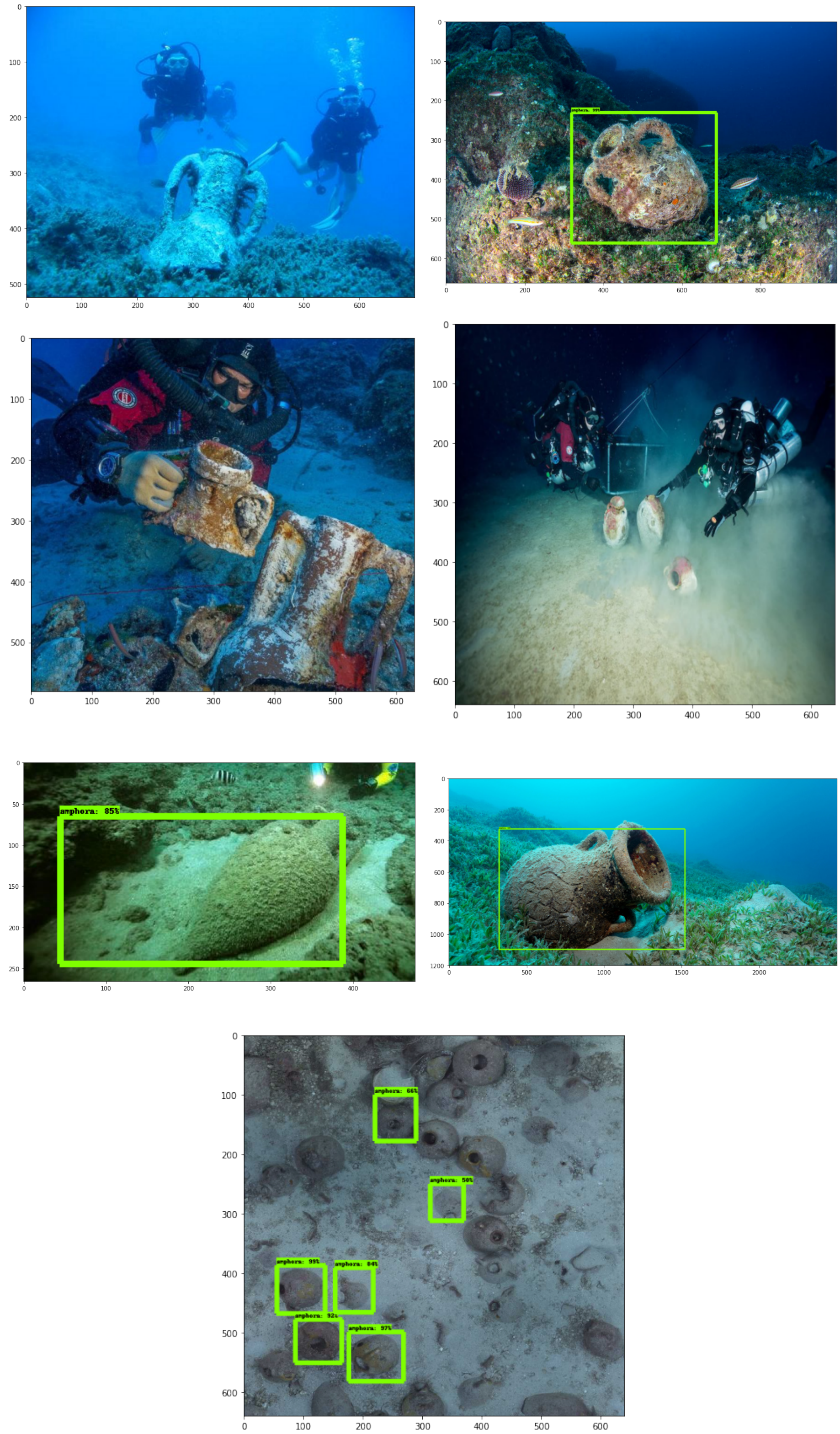

Figure 17: The validation images. Sources: [87, 88, 89, 90, 91, 92, 93].

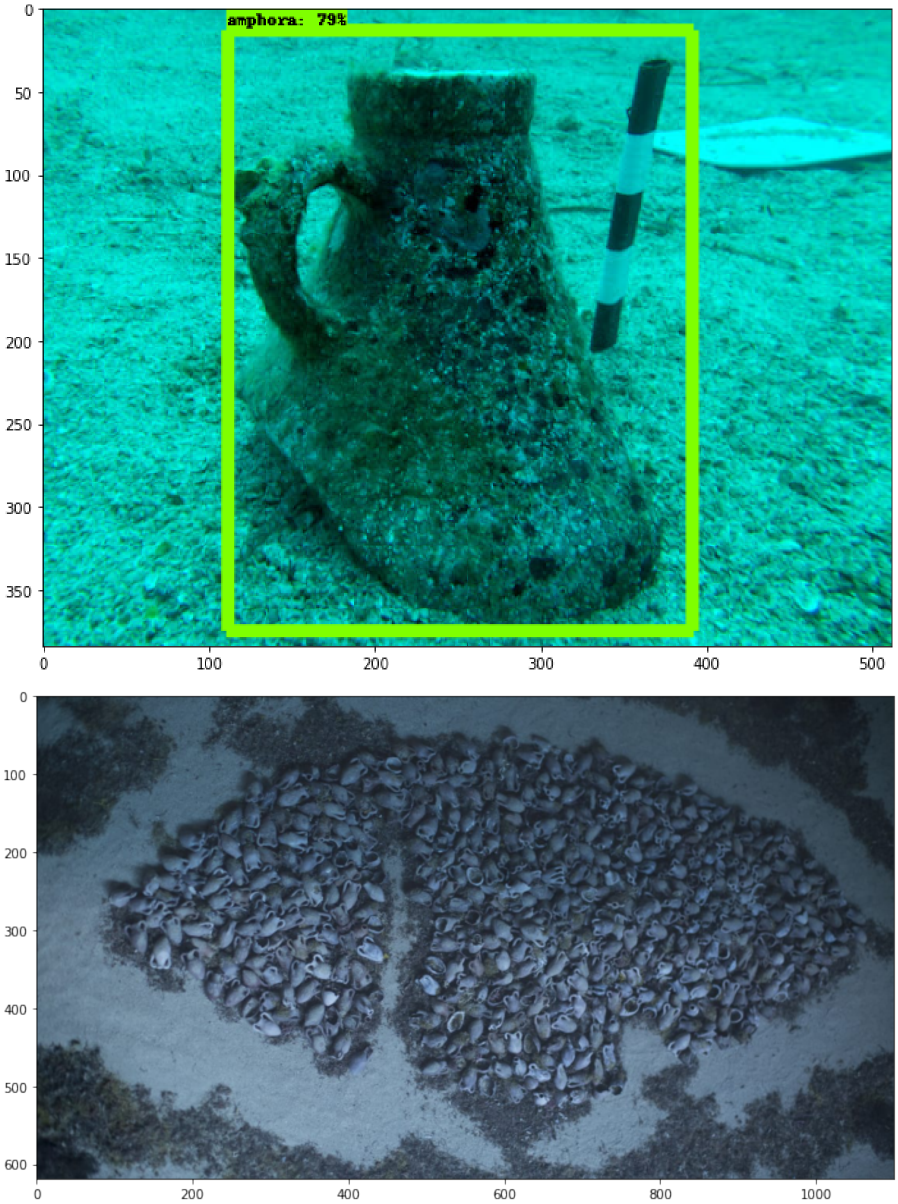

Figure 18: The test images. Sources: [94, 95].

5 Conclusion and Future Work

In this paper, we reviewed the relevance of amphoras, computer vision's potential in underwater archaeology, deep learning, and three influential object detector families. We then trained an SSD model with a ResNet-50 backbone on a very small and low-resolution dataset, and we still managed to achieve a 0.238 mAP_{coco} and a 0.5392 $mAP@0.5$. Detecting underwater amphoras is a challenging task. The object detectors have to overcome the same difficulties posed by general underwater object detection (see section 1.1.2), the considerable variations of the amphoras' shapes (see figure 2), and the crowdedness of the scenes in some shipwrecks (see figure 18).

We hope this paper can serve as a proof of concept for future studies since there are many aspects to improve upon:

- The small number and the low resolutions of the images in our dataset are the bottlenecks towards higher performance. There are indeed limited publicly available high-resolution images for underwater amphoras [62]. Future studies may focus on obtaining more amphora images through AUVs and UUVs to improve the mAP. Higher resolution images like orthophotos can address the issues of detecting small objects [110] and detecting overlapping objects by reducing labeling errors. Plus, more data augmentation techniques like randomly change to grayscale and randomly adjust brightness, contrast, and hue can be easily added in the data_augmentation_options from the TensorFlow Object Detection API to potentially further enrich the dataset.

- Many bleeding-edge object detectors with better performance are emerging. The SSD model in this study has a 0.35 mAP_{coco} and a 76 ms detection speed on the COCO validation set. YOLOv5 can achieve a 0.367 mAP_{coco} with a mere 2 ms detection speed or a 0.504 mAP_{coco} with a 6.1 ms detection speed for 640 x 640 images, and a 0.55 mAP_{coco} with a 70.8 ms detection speed for 1280 x 1280 images. It will be worth experimenting with YOLOv5 after its active development stabilizes. Some other models in the TensorFlow 2 Detection Model Zoo [116] are also interesting to explore. Note that we used the SSD model in the TensorFlow 1 Detection Model Zoo, as the TensorFlow 2 version requires two scripts to be running to train and evaluate simultaneously. It is unfortunately not possible to do so on Google Colaboratory. But one with access to GPUs can train an EfficientDet [117] D1 640 x 640 model with a 0.384 mAP_{coco} and a 54 ms detection speed or an EfficientDet D7 1536 x 1536 model with a 0.512 mAP_{coco} and a 325 ms detection speed from the TensorFlow 2 Detection Model Zoo. CenterNet [118], a novel keypoint-based detector that considers an object as a single center point and regresses to the object size without NMS (see section 1.4.1), is also available in the TensorFlow 2 Detection Model Zoo. CenterNet can achieve a 0.417 mAP_{coco} with a 6 ms detection speed or a 0.614 mAP_{coco} with a 76 ms detection speed for 512 x 512 images, and a 0.645 mAP_{coco} with a 211 ms detection speed for 1024 x 1024 images.

- Following the approach from Pasquet et al. [62], we can try to define two separate classes for the head and the body to better detect broken instances.

References

[1] Douglas Harper et al. "Online etymology dictionary". In: (2001).

[2] Diana Twede. "Commercial amphoras: the earliest consumer packages?" In: *Journal of Macromarketing* 22.1 (2002), pp. 98–108.

[3] Elizabeth Lyding Will. "The ancient commercial amphora". In: *Archaeology* 30.4 (1977), pp. 264–270.

[4] Mark Cartwright at World History Encyclopedia. *Amphora*. 2016. URL: https://www.worldhistory.org/Amphora/ (visited on 05/02/2021).

[5] Brendan P Foley et al. "Aspects of ancient Greek trade re-evaluated with amphora DNA evidence". In: *Journal of Archaeological Science* 39.2 (2012), pp. 389–398.

[6] Virginia Grace. *Amphoras and the ancient wine trade*. Vol. 6. ASCSA, 1979.

[7] Virginia Grace. "The Middle Stoa dated by amphora stamps". In: *Hesperia: The Journal of the American School of Classical Studies at Athens* 54.1 (1985), pp. 1–54.

[8] Mohamed Elgendy. *Deep Learning for Vision Systems*. Manning Publications, 2020.

[9] Waseem Rawat and Zenghui Wang. "Deep convolutional neural networks for image classification: A comprehensive review". In: *Neural computation* 29.9 (2017), pp. 2352–2449.

[10] Zhong-Qiu Zhao et al. "Object detection with deep learning: A review". In: *IEEE transactions on neural networks and learning systems* 30.11 (2019), pp. 3212–3232.

[11] Li Liu et al. "Deep learning for generic object detection: A survey". In: *International journal of computer vision* 128.2 (2020), pp. 261–318.

[12] Yongcheng Jing et al. "Neural style transfer: A review". In: *IEEE transactions on visualization and computer graphics* 26.11 (2019), pp. 3365–3385.

[13] Ian J Goodfellow et al. "Generative adversarial networks". In: *arXiv preprint arXiv:1406.2661* (2014).

[14] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. "Deep face recognition". In: (2015).

[15] Alexander Toshev and Christian Szegedy. "Deeppose: Human pose estimation via deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1653–1660.

[16] Ronald Poppe. "A survey on vision-based human action recognition". In: *Image and vision computing* 28.6 (2010), pp. 976–990.

[17] Wei Niu, James Caverlee, and Haokai Lu. "Neural personalized ranking for image recommendation". In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 2018, pp. 423–431.

[18] L van der Maaten et al. "Computer vision and machine learning for archaeology". In: (2007).

[19] Hongwei Qin et al. "When underwater imagery analysis meets deep learning: A solution at the age of big visual data". In: *OCEANS 2015-MTS/IEEE Washington*. IEEE. 2015, pp. 1–5.

[20] Dario Lodi Rizzini et al. "Investigation of vision-based underwater object detection with multiple datasets". In: *International Journal of Advanced Robotic Systems* 12.6 (2015), p. 77.

[21] Huimin Lu et al. "Underwater optical image processing: a comprehensive review". In: *Mobile networks and applications* 22.6 (2017), pp. 1204–1211.

[22] John K McCarthy et al. *3D Recording and Interpretation for Maritime Archaeology*. Springer Nature, 2019.

[23] Avi Abu and Roee Diamant. "A statistically-based method for the detection of underwater objects in sonar imagery". In: *IEEE Sensors Journal* 19.16 (2019), pp. 6858–6871.

[24] Alan Gordon. "Use of laser scanning system on mobile underwater platforms". In: *Proceedings of the 1992 Symposium on Autonomous Underwater Vehicle Technology*. IEEE. 1992, pp. 202–205.

[25] Md Moniruzzaman et al. "Deep learning on underwater marine object detection: A survey". In: *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer. 2017, pp. 150–160.

[26] Pierre Drap et al. "Underwater photogrammetry and object modeling: a case study of Xlendi Wreck in Malta". In: *Sensors* 15.12 (2015), pp. 30351–30384.

[27] Arthur L Samuel. "Some studies in machine learning using the game of checkers". In: *IBM Journal of research and development* 3.3 (1959), pp. 210–229.

[28] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[29] Andriy Burkov. *The hundred-page machine learning book*. Vol. 1. Andriy Burkov Canada, 2019.

[30] WJ Zhang et al. "On definition of deep learning". In: *2018 World automation congress (WAC)*. IEEE. 2018, pp. 1–5.

[31] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[32] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[33] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[34] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[35] Siegrid Lowel and Wolf Singer. "Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity". In: *Science* 255.5041 (1992), pp. 209–212.

[36] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[37] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[38] Yann LeCun et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551.

[39] Aäron Van Den Oord, Sander Dieleman, and Benjamin Schrauwen. “Deep content-based music recommendation”. In: *Neural Information Processing Systems Conference (NIPS 2013)*. Vol. 26. Neural Information Processing Systems Foundation (NIPS). 2013.

[40] Ronan Collobert and Jason Weston. “A unified architecture for natural language processing: Deep neural networks with multitask learning”. In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 160–167.

[41] David H Hubel. “Single unit activity in striate cortex of unrestrained cats”. In: *The Journal of physiology* 147.2 (1959), pp. 226–238.

[42] David H Hubel and Torsten N Wiesel. “Receptive fields of single neurones in the cat’s striate cortex”. In: *The Journal of physiology* 148.3 (1959), pp. 574–591.

[43] David H Hubel and Torsten N Wiesel. “Receptive fields and functional architecture of monkey striate cortex”. In: *The Journal of physiology* 195.1 (1968), pp. 215–243.

[44] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[45] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[46] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[47] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[48] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[49] François Chollet. “Xception: Deep learning with depthwise separable convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258.

[50] Andrew G Howard et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv preprint arXiv:1704.04861* (2017).

[51] Mark Sandler et al. “Mobilenetv2: Inverted residuals and linear bottlenecks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[52] Jie Hu, Li Shen, and Gang Sun. “Squeeze-and-excitation networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141.

[53] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[54] Sepp Hochreiter. “The vanishing gradient problem during learning recurrent neural nets and problem solutions”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 (1998), pp. 107–116.

[55] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[56] Athanasios Voulodimos et al. “Deep learning for computer vision: A brief review”. In: *Computational intelligence and neuroscience* 2018 (2018).

[57] Niall O’Mahony et al. “Deep learning vs. traditional computer vision”. In: *Science and Information Conference*. Springer. 2019, pp. 128–144.

[58] Mark Everingham et al. “The pascal visual object classes (voc) challenge”. In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[59] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[60] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[61] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[62] Jérôme Pasquet et al. “Amphora detection based on a gradient weighted error in a convolution neuronal network”. In: (2017).

[63] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[64] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[65] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *arXiv preprint arXiv:1506.01497* (2015).

[66] Wei Liu et al. “Ssd: Single shot multibox detector”. In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[67] Joseph Redmon and Ali Farhadi. “YOLO9000: better, faster, stronger”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[68] Joseph Redmon and Ali Farhadi. “Yolov3: An incremental improvement”. In: *arXiv preprint arXiv:1804.02767* (2018).

[69] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. “Yolov4: Optimal speed and accuracy of object detection”. In: *arXiv preprint arXiv:2004.10934* (2020).

[70] Ultralytics. *YOLOv5*. 2020. URL: https://github.com/ultralytics/yolov5 (visited on 04/24/2021).

[71] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. “Measuring the objectness of image windows”. In: *IEEE transactions on pattern analysis and machine intelligence* 34.11 (2012), pp. 2189–2202.

[72] Navaneeth Bodla et al. “Soft-NMS–improving object detection with one line of code”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5561–5569.

[73] Benjamin Planche and Eliot Andres. *Hands-On Computer Vision with TensorFlow 2: Leverage deep learning to create powerful image processing apps with TensorFlow 2.0 and Keras*. Packt Publishing Ltd, 2019.

[74] Jesse Davis and Mark Goadrich. “The relationship between Precision-Recall and ROC curves”. In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 233–240.

[75] Microsoft. *COCO detection evaluation metrics*. URL: https://cocodataset.org/#detection-eval (visited on 04/22/2021).

[76] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[77] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. “Dropblock: A regularization method for convolutional networks”. In: *arXiv preprint arXiv:1810.12890* (2018).

[78] Diganta Misra. “Mish: A self regularized non-monotonic neural activation function”. In: *arXiv preprint arXiv:1908.08681* 4 (2019).

[79] Xiu Li et al. “Fast accurate fish detection and recognition of underwater images with fast r-cnn”. In: *OCEANS 2015-MTS/IEEE Washington*. IEEE. 2015, pp. 1–5.

[80] David Zhang et al. “Unsupervised underwater fish detection fusing flow and objectiveness”. In: *2016 IEEE Winter Applications of Computer Vision Workshops (WACVW)*. IEEE. 2016, pp. 1–7.

[81] Sébastien Villon et al. “Coral reef fish detection and recognition in underwater videos by supervised machine learning: Comparison between Deep Learning and HOG+ SVM methods”. In: *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer. 2016, pp. 160–171.

[82] Wenwei Xu and Shari Matzner. “Underwater fish detection using deep learning for water power applications”. In: *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE. 2018, pp. 313–318.

[83] Dmitry A Konovalov et al. “Underwater fish detection with weak multi-domain supervision”. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, pp. 1–8.

[84] Shuo Cao et al. “Real-time robust detector for underwater live crabs based on deep learning”. In: *Computers and Electronics in Agriculture* 172 (2020), p. 105339.

[85] Tsung-Yi Lin et al. “Feature pyramid networks for object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[86] Google. *Google Images*. URL: https://images.google.com/.

[87] URL: https://scuba247.net/dive-sites/ (visited on 05/01/2021).

[88] Marko Radojević. *Discover the underwater winery on the Pelješac peninsula*. URL: https://www.itinari.com/discover-the-underwater-winery-on-the-peljesac-peninsula-1dsk (visited on 05/01/2021).

[89] Woods Hole Oceanographic Institute. *Artifacts Discovered on Return Expedition to Antikythera Shipwreck*. URL: https://www.whoi.edu/press-room/news-release/artifacts-discovered-on-return-expedition-to-antikythera-shipwreck/ (visited on 05/01/2021).

[90] University of Malta. *Phoenician Shipwreck Project*. URL: https://phoenicianshipwreck.org/ (visited on 04/28/2021).

[91] Kevin Deacon. *Scuba diver examining ancient amphora underwater, Egypt*. URL: https://images.auscape.com.au/objects/scuba-diver-examining-ancient-amphora-underwater-10091765.html (visited on 05/01/2021).

[92] Gemma Conroy. *Seagrass Safeguards Human History*. URL: https://www.hakaimagazine.com/news/seagrass-safeguards-human-history/ (visited on 05/01/2021).

[93] CNRS LSIS. *GROPLAN*. URL: http://www.lsis.org/groplan/ (visited on 04/28/2021).

[94] Ionian Aquarium. *The Fiscardo Roman Merchant Ship Wreck*. URL: https://www.ionian-aquarium.com/fiscardo-roman-wreck1.html (visited on 05/01/2021).

[95] Sanisera Archaeology Institute for International Field Schools. *Discover Amphora and Shipwrecks in the Underwater Port of Sanitja*. URL: http://archaeology.institute/013-discover-amphora-and-shipwrecks-in-sanitja.asp (visited on 05/01/2021).

[96] TuzTa Lin. *LabelImg*. URL: https://github.com/tzutalin/labelImg (visited on 04/29/2021).

[97] Jonathan Huang et al. "Speed/accuracy trade-offs for modern convolutional object detectors". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7310–7311.

[98] TensorFlow. *TensorFlow Object Detection API*. URL: https://github.com/tensorflow/models/tree/master/research/object_detection (visited on 04/29/2021).

[99] TensorFlow. *TensorFlow 1 Detection Model Zoo*. URL: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md (visited on 04/29/2021).

[100] Lisa Torrey and Jude Shavlik. "Transfer learning". In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[101] Alex Krizhevsky and Geoff Hinton. "Convolutional deep belief networks on cifar-10". In: *Unpublished manuscript* 40.7 (2010), pp. 1–9.

[102] Andrew Y Ng. "Feature selection, L 1 vs. L 2 regularization, and rotational invariance". In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 78.

[103] Douglas M Hawkins. "The problem of overfitting". In: *Journal of chemical information and computer sciences* 44.1 (2004), pp. 1–12.

[104] Boris T Polyak. "Some methods of speeding up the convergence of iteration methods". In: *Ussr computational mathematics and mathematical physics* 4.5 (1964), pp. 1–17.

[105] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[106] Priya Goyal et al. "Accurate, large minibatch sgd: Training imagenet in 1 hour". In: *arXiv preprint arXiv:1706.02677* (2017).

[107] Andrew Senior et al. “An empirical study of learning rates in deep neural networks for speech recognition”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 6724–6728.

[108] Ilya Loshchilov and Frank Hutter. “Sgdr: Stochastic gradient descent with warm restarts”. In: *arXiv preprint arXiv:1608.03983* (2016).

[109] Google. *Colaboratory*. URL: https://colab.research.google.com (visited on 04/29/2021).

[110] Mate Kisantal et al. “Augmentation for small object detection”. In: *arXiv preprint arXiv:1902.07296* (2019).

[111] Jianan Li et al. “Perceptual generative adversarial networks for small object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1222–1230.

[112] Christian Eggert et al. “A closer look: Small object detection in faster R-CNN”. In: *2017 IEEE international conference on multimedia and expo (ICME)*. IEEE. 2017, pp. 421–426.

[113] Eran Goldman et al. “Precise detection in densely packed scenes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5227–5236.

[114] Bastian Leibe, Edgar Seemann, and Bernt Schiele. “Pedestrian detection in crowded scenes”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 1. IEEE. 2005, pp. 878–885.

[115] Carlos Arteta et al. “Learning to detect partially overlapping instances”. In: *Proceedings of the IEEE Conference on Computer Vision And Pattern Recognition*. 2013, pp. 3230–3237.

[116] TensorFlow. *TensorFlow 2 Detection Model Zoo*. URL: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md (visited on 05/03/2021).

[117] Mingxing Tan, Ruoming Pang, and Quoc V Le. “Efficientdet: Scalable and efficient object detection”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10781–10790.

[118] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. “Objects as points”. In: *arXiv preprint arXiv:1904.07850* (2019).