Deep Learning for Detecting Amphoras in Ancient Shipwrecks

by

Tianyao Chen

Bachelor Thesis in Computer Science

Submission: May 2, 2021

Supervisor: Prof. Dr. Andreas Birk

Jacobs University Bremen | Department of Computer Science and Electrical Engineering

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

Consider this a separate document, although it is submitted together with the rest. The abstract aims at another audience than the rest of the proposal. It is directed at the final decision maker or generalist, who typically is not an expert at all in your field, but more a manager kind of person. Thus, don't go into any technical description in the abstract, but use it to motivate the work and to highlight the importance of your project.

(target size: 15-20 lines)

Contents

1 Introduction

1.1 Motivation

1.1.1 Relevance of Amphoras

The name *amphora* is derived from the Greek word *amphoreus*, which literally means ”two-handled” [1, 2]. It is the combination of two linguistic roots: *amphi* (on both sides) and *phoreus* (bearer) [1, 2]. Amphoras (or amphorae) were commercially used from 1500 B.C.E. to 500 C.E. to ship products throughout the Mediterranean, supplying the ancient Greek and Roman empires [2]. Amphoras were designed to ship large quantities of liquid (wine, olives, and oils) and dry products (grain, nuts, and salted fish) [2].

Like many measures that are named after the packages, amphoras were also a semi-standard unit of liquid measure [2]. A cargo ship's capacity was measured by the number of amphoras it could carry instead of by weight [2, 3].

The structurally strong egg-like shape and the high volume-to-weight ratio made amphoras very efficient packages [2]. Amphoras were by far the most common cargo type in Mediterranean shipwreck analysis; more than half of the ships only carried amphoras [2, 4].

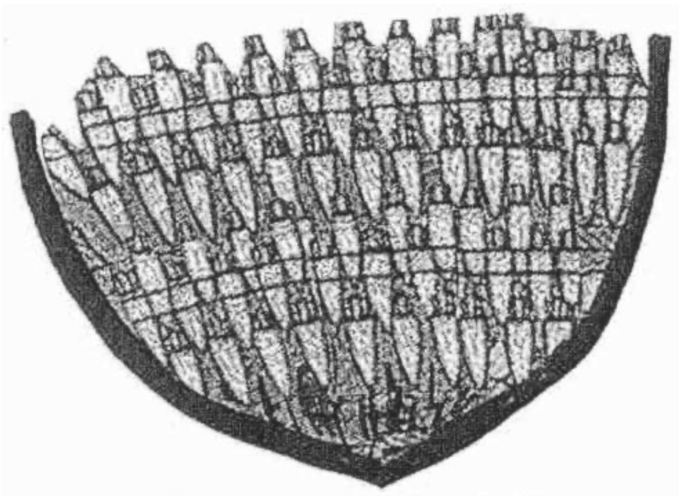

Figure 1: The egg-like shape enabled amphoras to interlock and minimize the waste of space on a ship. Source: [2].

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

Amphoras have great significance in archaeology. They can be used as evidence for the trade patterns throughout the Mediterranean [2]. As they were usually discarded at the destination of a trade and have been found in shipwrecks, archaeologists have been using them to recreate the transit routes [2]. Furthermore, researchers have been able to classify different amphoras, which also helps to date ruins and shipwrecks [2].

Figure 2: Amphoras have various shapes. Source: [5].

1.1.2 Computer Vision for Underwater Archaeology

Computer vision is the science of perceiving and understanding the world through images and videos [6]. There have been multiple exciting applications of computer vision, including image classification [7], object detection and localization [8, 9], art generation (neural style transfer) [10], image creation with Generative Artificial Networks (GAN) [11], face recognition [12], action and activity recognition [13], human pose estimation [14], and image recommendation system [15].

However, there is still limited research for the application of computer vision and machine learning in archaeology, especially underwater archaeology, compared to other domains [16, 17]. Computer vision, instead of visual inspection, could be used to automate the detection, assessment, and classification of artifacts [16].

Underwater computer vision has proven to be challenging, largely due to: 1) the distortion and attenuation caused by light propagation in water, 2) the unrestricted natural environment with the abundance of marine life and suspended particles, and 3) underwater objects are often broken [17, 18, 19, 20].

Despite the challenges, computer vision has lower cost [18] compared to sonar imagery [21] and laser scanning [22]. Plus, the increasingly abundant visual data obtained through autonomous underwater vehicles (AUVs), unmanned underwater vehicles (UUVs) [19, 23], and seafloor cabled observatories [17] enables us to utilize deep learning.

Furthermore, the research for deep-water shipwrecks is even more limited, mostly due to the lack of information and accessibility [24]. However, the need to study deep-water sites are in high demand, as the threats to these sites are increasing [24]. One major threat is the new forms of trawling that destroy the surface of these sites and interfere with the readability [24]. This means that many shipwrecks are likely to be damaged before they can be studied [24]. It is thus 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 [25, 26]. 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 [26, 27, 28]. Deep learning constructs complex representations by combining simpler ones from the previous layers [29].

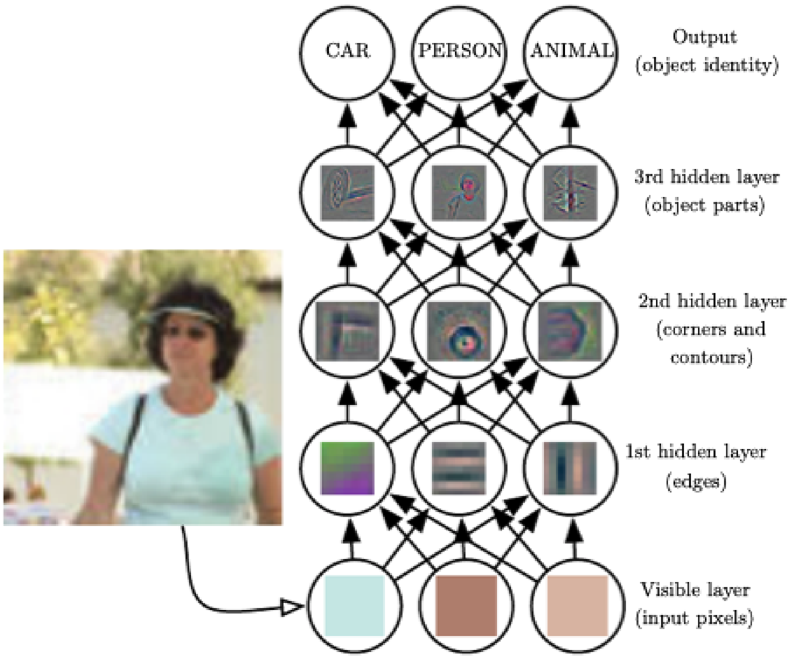

Figure 3: A deep learning system that learns the representations of a person. This is achieved by combining simpler features like corners and contours, which are further expressed by combining simpler features like edges. Source: [29].

1.2.1 Artifical Neural Networks (ANNs)

Inspired by the biological neuron, artificial neural networks (ANNs) were first introduced in 1943 using propositional logic [30]. The artificial neuron activates its single binary output when the number of active binary inputs reaches the activation threshold, which enables us to build networks that can perform any logical computation [26, 30].

Then the Perceptron was introduced in 1957, which is based on a different artificial neuron called threshold logic unit (TLU) or linear threshold unit (LTU) [31]. The inputs and outputs are numbers instead of binary values, and each input has a weight. TLU computes the weighted sum of the inputs and then applies a step function like the Heaviside

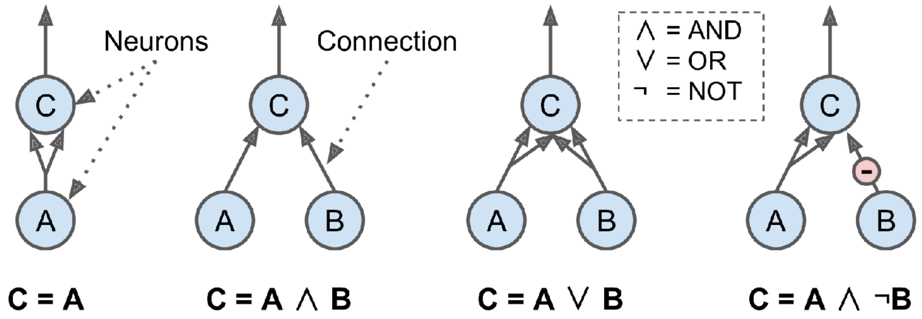

Figure 4: ANNs performing logical computations with the activation threshold of 2. Source: [26].

function $heaviside(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x \geq 0 \end{cases}$ [26, 31].

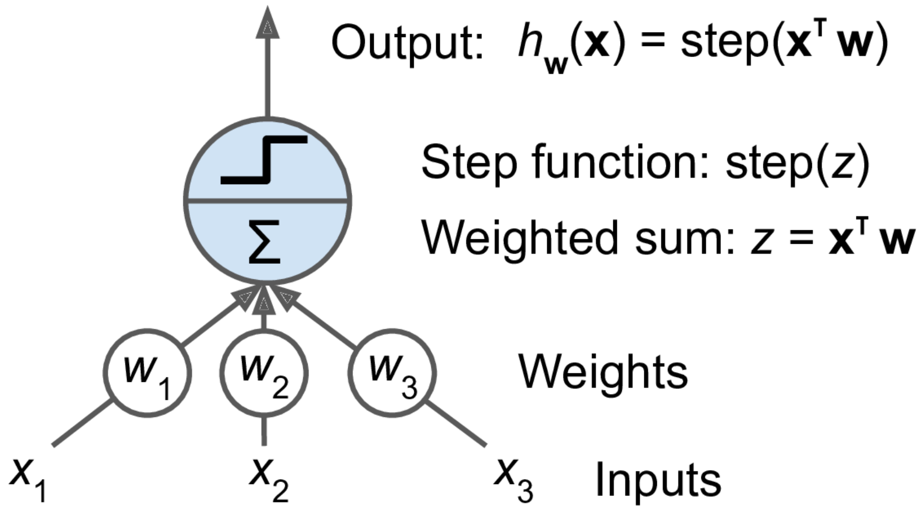

Figure 5: Threshold logic unit. Source: [26].

A single TLU can be used for simple linear binary classification, while a layer of TLUs plus a bias neuron form a Perceptron capable of multi-output classification [26].

The outputs of a fully connected layer 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:

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

The Perceptron is trained using a variant of the Herbb's rule [32], which is famously summarized as "neurons wire together if they fire together [33]" [26]. However, the Perceptron can not learn complex patterns due the linear decision boundary of the output neurons, and it can only make predictions based on a hard threshold instead of outputting a class probability [26]. To address these limitations, the Multilayer Perceptron (MLP) was introduced by stacking multiple Perceptrons.

Backpropagation [34] is used to train the MLP, which first makes a prediction and measures the error in the forward pass, then measures the error contribution from each connection in the reverse pass, and finally tweaks the connection weights to reduce teh

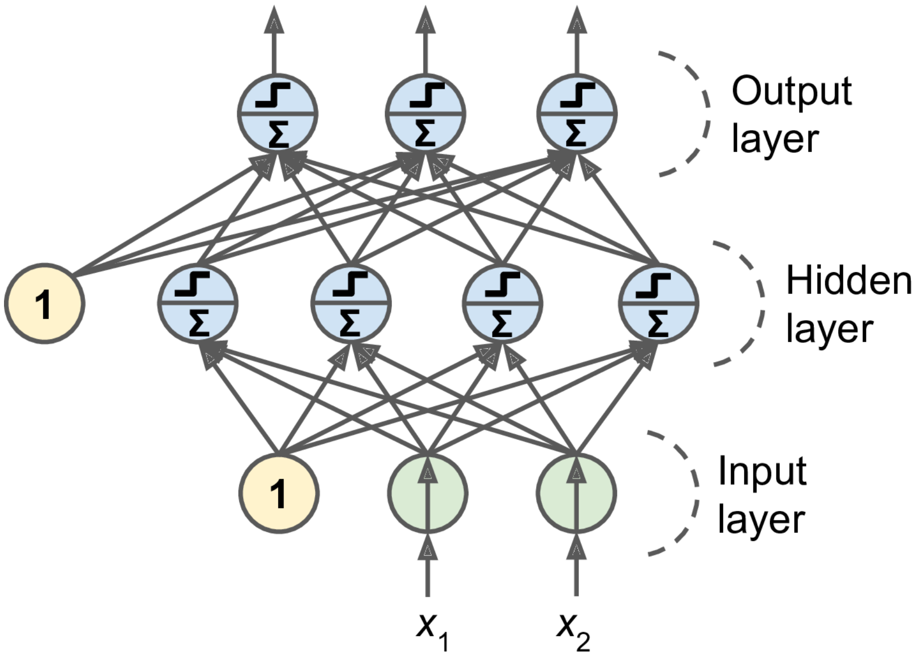

Figure 6: A Multilayer Perceptron with one hidden layer. Source: [26].

error in the Gradient Descent [35] step [26]. Activation functions like Rectified Linear Unit $ReLU(x) = max(0, x)$ are used to add nonlinearity, which theoretically gives a large enough deep neural network the ability to approximate any continuos function [26].

1.2.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) were inspired by the brain's visual cortex, and they have been used in computer vision since the 1980s [26]. 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 huge number of parameters it requires [26]. CNNs also have successful applications in other domains like recommender systems [36] and natural language processing (NLP) [37].

Hubel et al. [38, 39, 40] found that many biological neurons have a small receptive field, which means they only react to visual stimuli in a limited region of the visual field [26]. Some neurons only react to horizontal lines, while others only react to lines with different orientations [26]. Some neurons have larger receptive fields, and they react to more complex patterns formed by lower-level patterns [26].

Neurons in the first convolutional layer are only connected to pixels in their receptive fields, and neurons in the second convolutional layer are only connected to the neurons in a small receptive field in the first layer [26]. This allows the CNN to concentrate on lower-level features in the first hidden layer, then combine them into higher-level features in the second hidden layer, and so on [26].

The filters or convolution kernels, which are learned during training, are neurons' weights that can be presented as small images the size of receptive fields [26]. For example, a black sqaure with a horizontal white line in the middle (a matrix full of 0s except for the central row with 1s) is a filter that only reacts to the central row in the receptive field. A

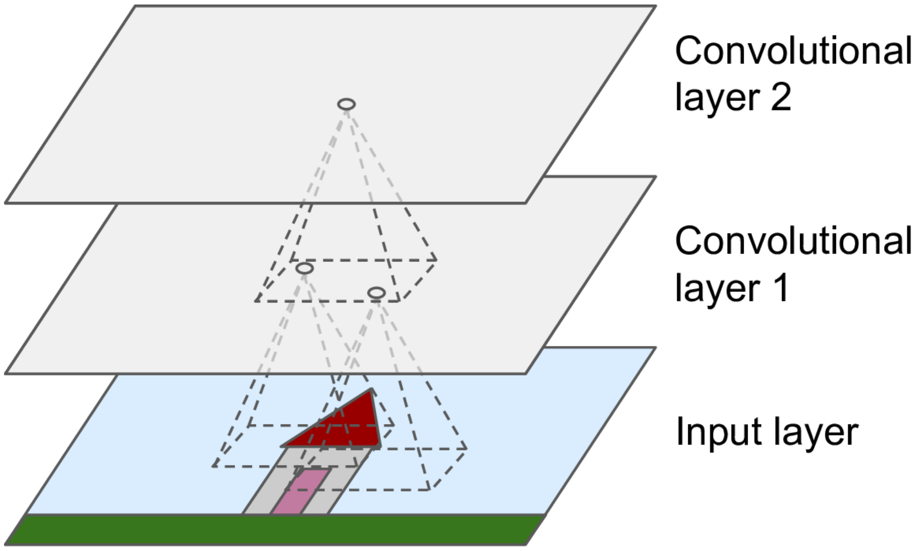

Figure 7: CNN layers with rectangular receptive fields. Source: [26].

layer of neurons with the same filter outputs a feature map, which highlights the parts of an image that activate the filter the most [26].

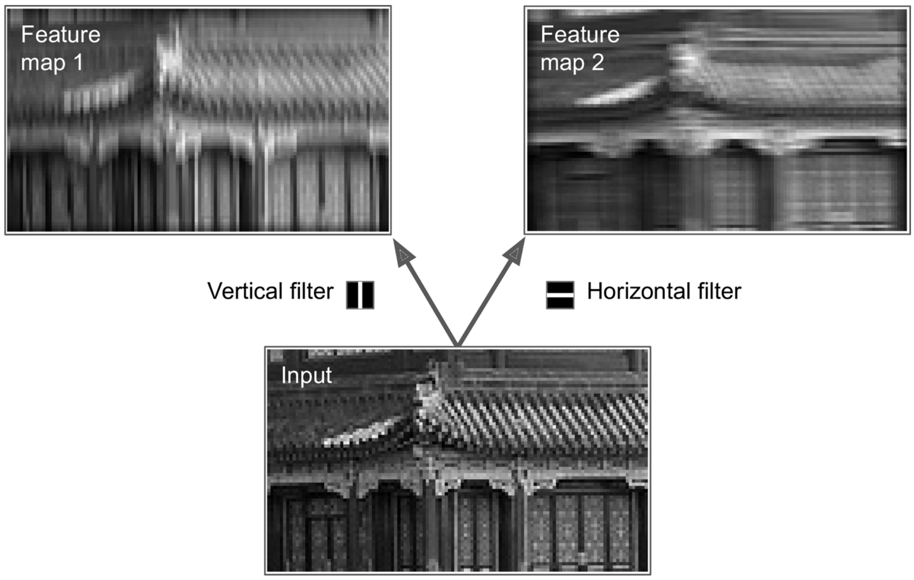

Figure 8: Two feature maps obtained by applying two different filters. Source: [26].

The pooling layers subsample (i.e. shrink) the input image to reduce the computational load and the number of paramters, which also reduces overfitting [26]. Plus, pool layers can bring some invariance to small translations, rotations, and scaling [26].

The typical CNN architecture involves the aforementioned convolutional layers, pooling layers, and fully connected layers. Some well-established CNN architecutres are LeNet-5 [41], AlexNet [42], GoogLeNet [43], VGGNet [44], ResNet (Residual Network) [45], Xception (Extreme Inception) [46], MobileNet [47, 48], and SENet (Squeeze-and-Excitation Network) [49].

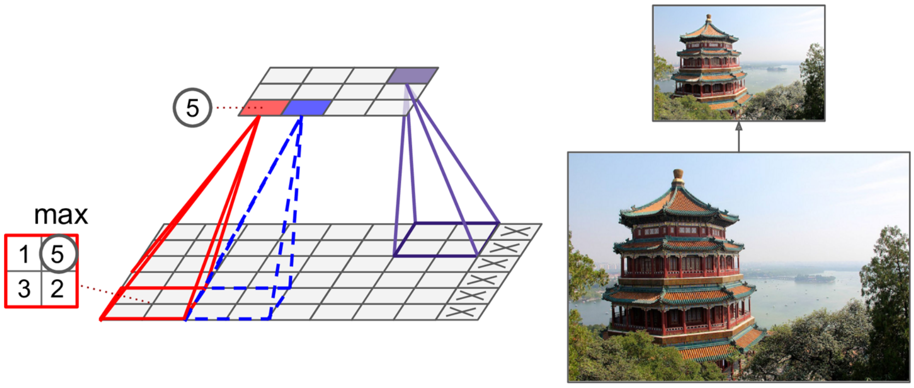

Figure 9: A max pooling layer with a 2 x 2 kernel and stride 2. Only the max value from each receptive field gets passed to the next layer. Source: [26].

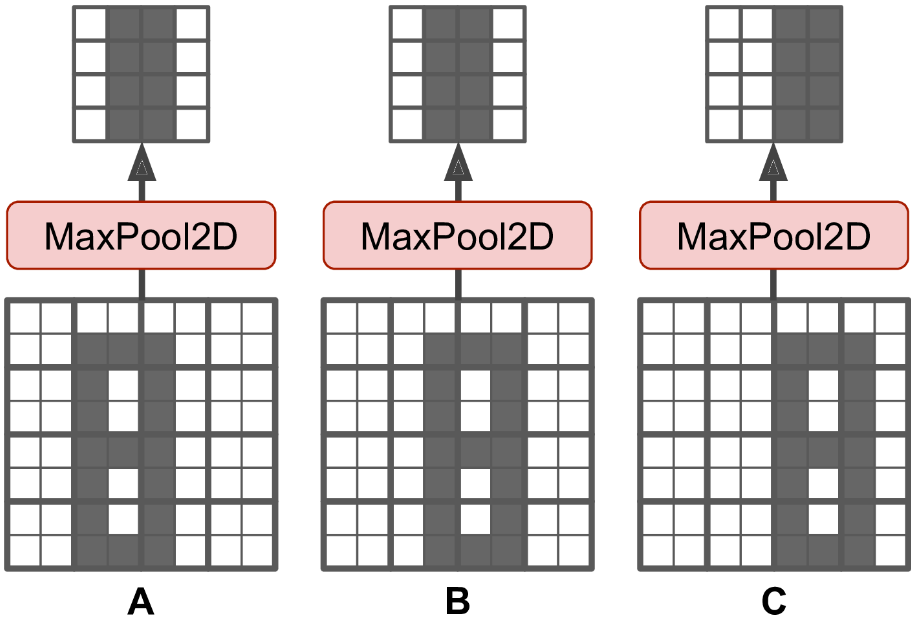

Figure 10: Max pooling layer's invariance to small translations. Source: [26].

Here we give ResNet a more in-depth introduction, as it's the backbone network of the model we use to detect amphoras. Resnet adds skip connections (or shortcut connections), which means the input signal of a layer is also added to the ouput of a higher up layer [26, 45]. Skip connections help speed up the training considerably, since: 1) the network preconditions the problem to be the identify 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 [26, 45]. Batch normalization [50] is used after each convolution, which zero-centers and normalizes each input to reduce the vanishing gradient problem [51] and the need for other regularization techniques like dropout [52] [26, 45]. The global average pooling layer at the end of the network is another type of pooling layer that gets the mean of the entire feature map [26, 45]. Softmax is used in the output layer to ensure that all probabilities add up to 1 [26, 45].

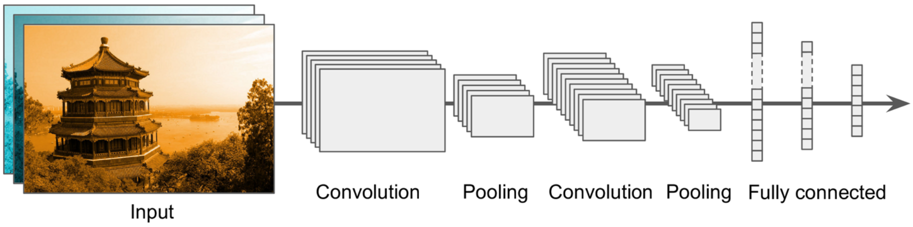

Figure 11: The typical CNN architecuture. Source: [26].

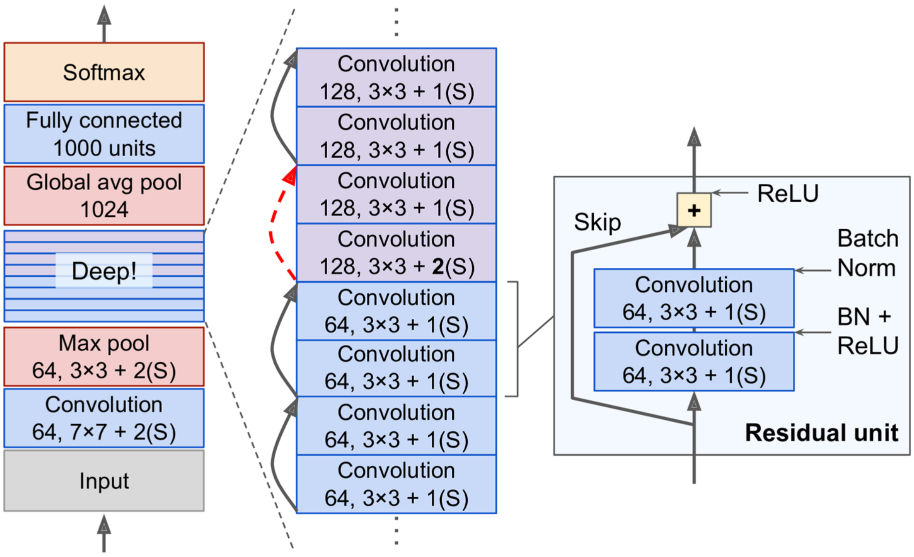

Figure 12: ResNet architecture. Source: [26].

1.3 Deep Learning vs. Traditional Computer Vision

The increasingly abundant visual data and the improvements of deep learning algorithms, computing power, and image resolution have enabled deep learning to achieve state-of-the-art performance in many computer vision tasks, including image classification, object detection, and semantic segmentation [17, 53, 54].

Traditionally, computer vision requires the manual feature selection and engineering step, which relies on domain knowledge to produce high-quality features [6, 8, 54]. These handcrafted features are further procesed by a machine learning classifier like a support vector machine (SVM) [6, 8, 54]. However, manual feature extraction becomes more and more complex as the number of classes increases [54]. Besides, conventional machine learning algorithms' ability to generalize saturates quickly as the size of training data grows [17].

In deep learning, the manual feature extraction step is no longer needed, and neural networks can be trained end-to-end as a feature extractor plus a classifier [6, 54]. Thus, deep learning requires less domain knowledge, it and provides more flexibility as the models can be re-trained with a custom dataset for any specific use case [54].

However, deep learning does not make traditional computer vision obsolete. Techniques like Scale Invariant Feature Transform (SIFT) [55], Speeded Up Robust Features (SURF) [56], and Features from Accelerated Segment Test (FAST) [57] are still useful in improving performance for computer vision tasks [54]. And deep learning's performance depends

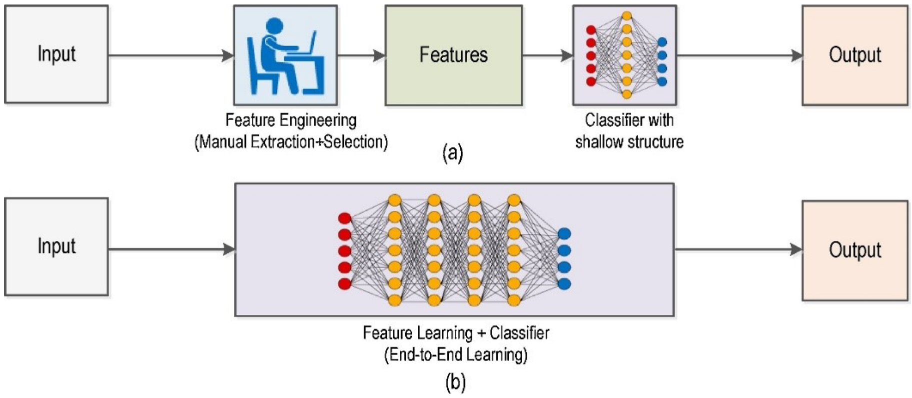

Figure 13: (a) Traditional computer vision workflow. (b) Deep learning workflow. Source: [54].

on obtaining large datasets with high image resolution [54]. Some popular public datasets like PASCAL Visual Object Classes (VOC) [58], ImageNet [59], and Microsoft Common Objects in Context (COCO) [60] have respectively 500 thousand, 14 million, and 328 thousand images.

1.4 Object Detection

Object detection is the computer vision task that localizes and classifies objects in an image [6, 8, 9, 26]. Object detection remains to be one of the most challenging problems in computer vision, as it can be considered as both a regression task (localization by predicting the bounding box) and a classification task (predicting the object class in each bounding box) [6, 61, 26]. Plus, the large variantions in viewpoints, poses, occlusions, and lighting conditions make it even more difficult to perform perfect object detection [8, 9].

The traditional sliding window approach is to train a CNN to classify and locate a single object, and then slide it across the image [26, 62, 61, 63]. This approach slides the CNN mulitple times with different window sizes to detect objects with various sizes, which causes it to be quite slow [26].

Luckily, many object detection frameworks based on fully convolutional network (FCN) [64] have been introduced. FCNs contain only convolutional layers and pooling layers and only need to process each image once, which means they are faster and the input images can have arbitrary sizes [6, 26, 64]. Here we summarize three well-established and influential object detection framework families: Region-Based Convolutional Neural Networks (R-CNN) [61, 65, 66], Single Shot Detector (SSD) [67], and You Only Look Once (YOLO) [63, 68, 69, 70, 71].

1.4.1 General Object Detection Framework Components

Before we dive into specific object detection frameworks, it's worth understanding the four high-level components of general object detection frameworks [6].

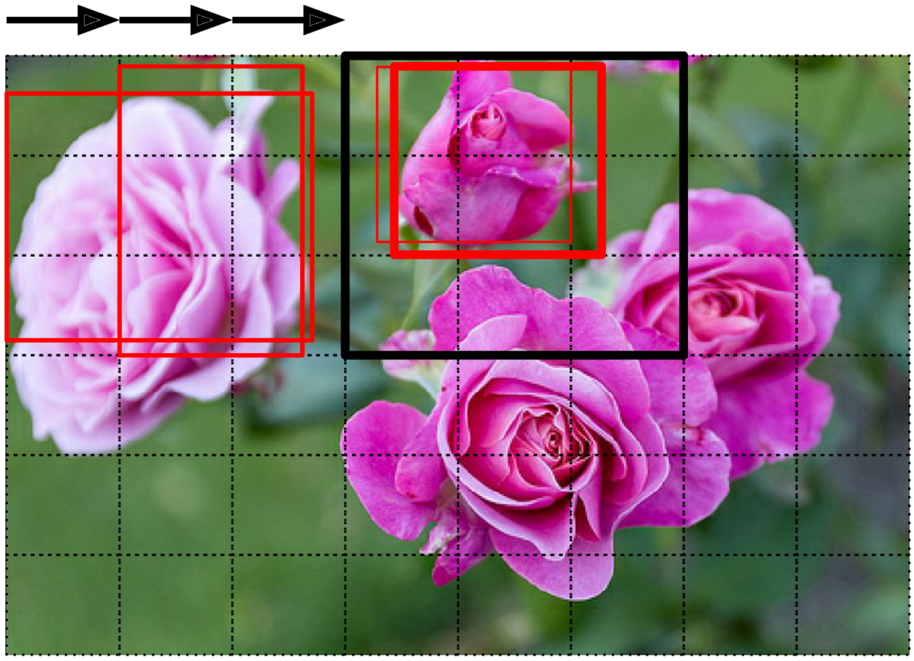

Figure 14: The sliding window approach for object detection. Source: [54].

Region Proposal The region proposal algorithm finds regions of interest (RoIs) that are further processed by the framework by discarding regions with low objectness score [6]. The objectness score indicates the probability of the region containing an object instead of the background, and it is the class probability for a single-class object detection task [6].

Network Predictions A pretrained CNN is used as the backbone for feature extraction and makes the bounding-box prediction and class prediction for each region [6]. The bounding-box prediction is the tuple (x, y, w, h), where x and y are the coordinates of the center and w, h are the width and the height [6].

Non-Maximum Suppression (NMS) The backbone network typically produces mulitple overlapping bounding boxes for one object, thus NMS finds the box with the maximum class probability and suppresses the rest [6]. The steps include [6]:

1. Discard boxes with predictions less than the tunable confidence threshold.
2. Select the box with the highest probability.
3. Compute the overlap - intersection over union (IoU) - of the boxes that have the same class prediction. Boxes with high IoU are averaged together.
4. Suppress boxes with an IoU less than the tunable NMS threshold.

Metrics The two main metrics for object detection are frames per second (FPS) and mean average precision (mAP) [6, 9, 26, 72]. To understand mAP, we need to understand first the aforementioned intersection over union (IoU) and the precision-recall curve (PR

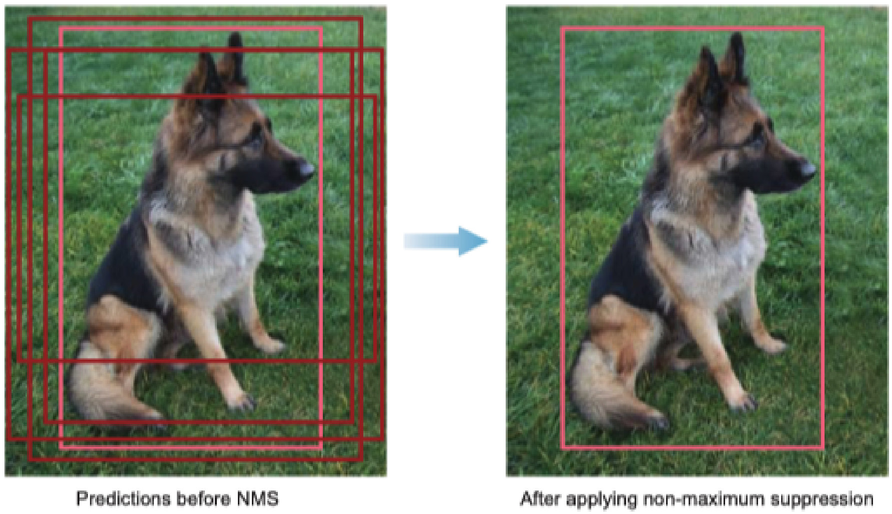

Figure 15: NMS. Source: [6].

curve). The IoU is also known as the Jaccard index, which is used to messaure similarity between two sets [72]. The IoU can be mathematically formulated as follows [6, 72]:

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

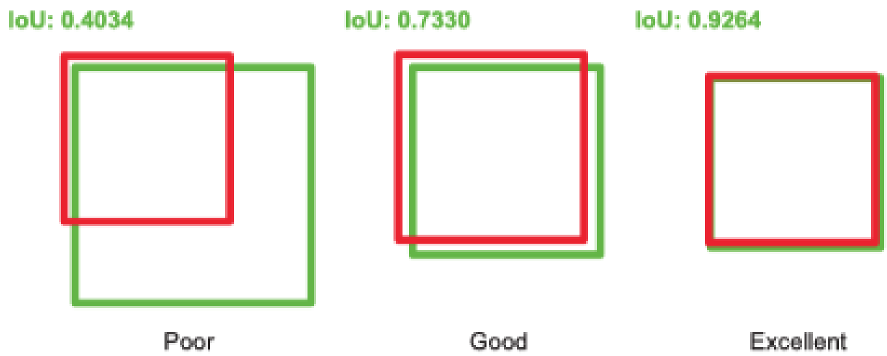

Figure 16: IOU. Source: [6].

We say that the prediction is a true positive (TP) if the predicted class is the same as the ground truth and the IoU value is more than the tunable threshold, otherwise it is a false positive (FP) [9, 6, 72]. A false negative (FN) is a ground truth that does not have a prediction [72]. Then we can define precision and recall as follows [27, 73]:

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

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

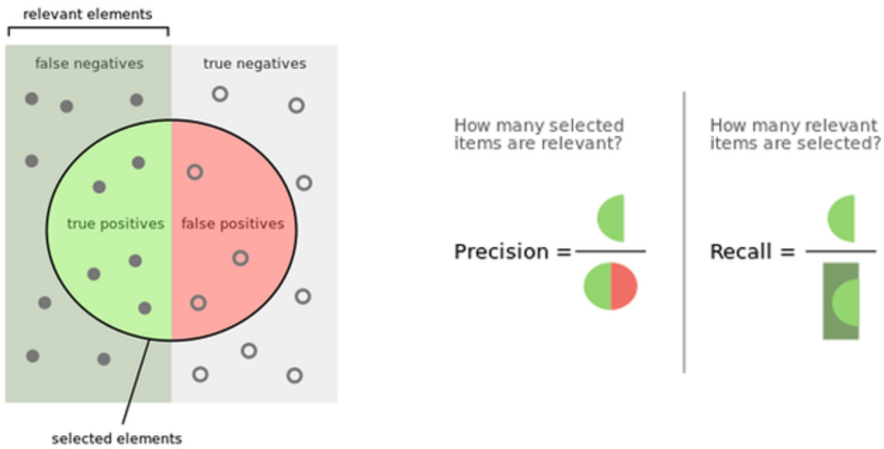

Figure 17: Precision and recall. Source: [74].

There is a trade-off between precision and recall [6, 26, 27, 72]. We can obtain the average precision (AP) by drawing the precision-recall curve (PR curve) and computing the area under the curve (AUC) [6, 72]. Finally, we get the mean average precision (mAP) by averaging the AP over all the classes [6, 26, 72]. Note that the AP and mAP are the same when it's a single-class object detetction task. The Pascal VOC metric uses mAP@0.5, which means the IOU threshold is 0.5 [9, 58, 72]. And the MS COCO metric - the one we are using for detecting amphoras - $mAP_{coco} = mAP@[0.50 : 0.05 : 0.95]$ is averaged with different IOU thresholds from 0.5 to 0.95 in steps of 0.05, which rewards detectors with better localization [9, 72, 75]. MS COCO also introduces mAP for different scales: mAP^{small} for small objects with area smaller than 32^2 pixels, mAP^{medium} for medium objects with area bigger than than 32^2 pixels and smaller than 96^2 pixels, and mAP^{big} for big objects with area bigger than 96^2 pixels [9, 75].

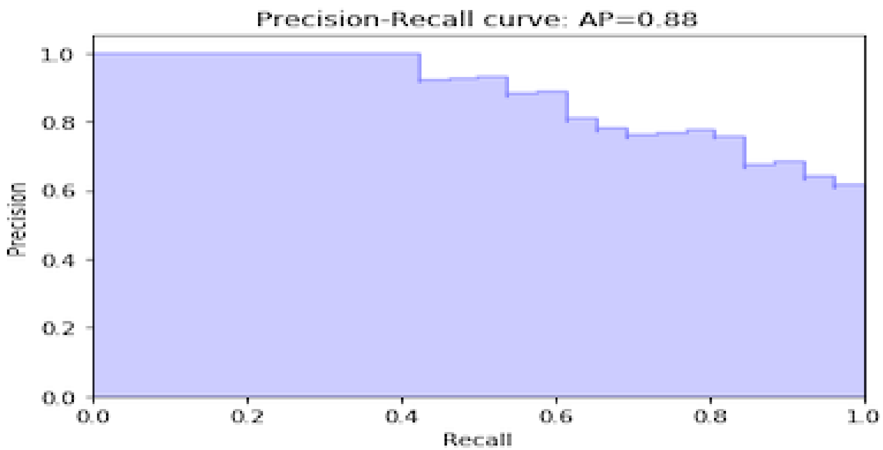

Figure 18: A PR curve with 0.88 AUC. Source: [72].

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

The evolution from the original R-CNN [61] to Fast R-CNN [65] and then to Faster R-CNN [66] builds up the R-CNN family.

R-CNN R-CNN has four components [6, 61]:

1. Region proposal with a greedy search algorithm called selective search, which finds RoIs by combining similar pixels into boxes.
2. Feature extractor with a pretrained CNN.
3. Classification with a linear SVM.
4. Localization with a bounding-box regressor.

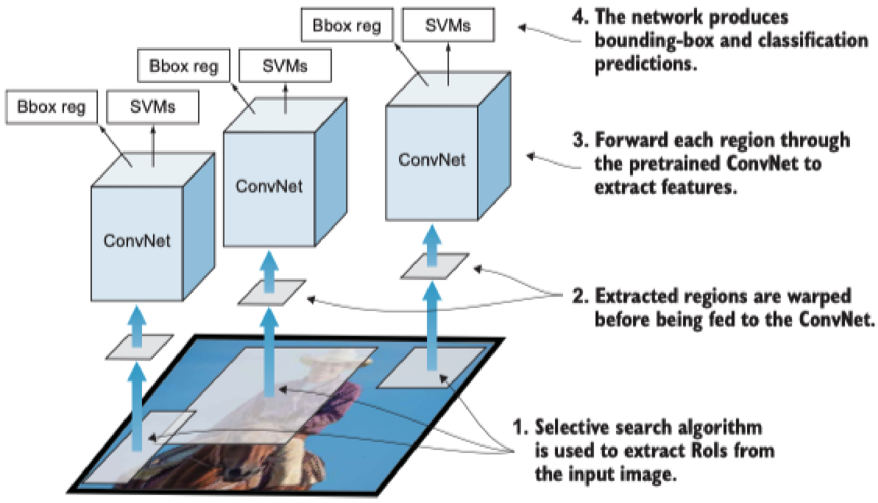

Figure 19: R-CNN. Source: [6].

R-CNN has the following disadvantages [6, 61, 65]:

- The FPS is very low. The selective search algorithm proposes about 2000 RoIs, which is every 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 makes the following changes from R-CNN [6, 65]:

- The CNN goes before region proposal instead of after, so that the image only goes through the CNN once instead of 2000 RoIs going through the CNN separately.
- Classification is done by the softmax layer of the CNN instead of the SVM. And localization is also an output layer of the CNN.
- A RoI max pooling layer is added after region proposal to fix the input size for the fully connected layers.
- A multi-task loss function is used.

Fast R-CNN is much faster than R-CNN, although the selective search algorithm still exists as the bottleneck [6, 65, 66].

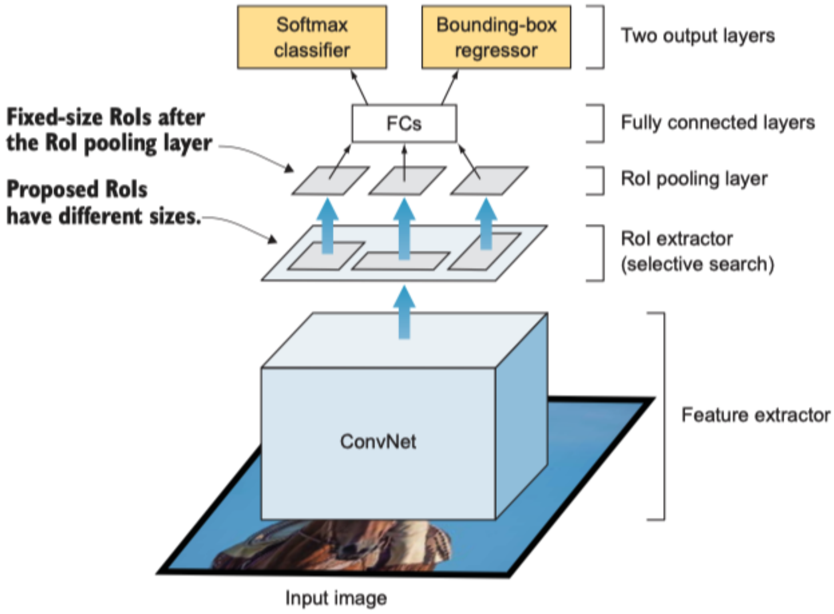

Figure 20: Fast R-CNN. Source: [6].

Faster R-CNN Faster R-CNN makes the following improvments from Fast R-CNN [6, 66]:

- Region Proposal Network (RPN) or attention network replaces selective search, which reduces the number of proposals, speeds up the model, and makes the model training end-to-end. RPN is a FCN that outputs objectness scores and RoIs, and it can be used as a standalone network for single-class object detection. It also shares the features with the detection network, which enables region proposal to be nearly cost-free.

- Anchors are introduced as reference boxes at different scales and aspect ratios. Thus the regression layer only needs to output the offsets of coordinates, width, and height from the anchors. The anchors are created using the sliding-window approach. By default 9 anchors (3 scales and 3 aspect ratios) centered at each sliding window are created for each window.

To summarize, the R-CNN family are two-stage detectors that separate region proposal and detection [6]. They can not achieve real-time detection (only 7 FPS), and they are too computationally intensive [6, 67, 63]. One-stage detectors like Single Shot Detector (SSD) and You Only Look Once (YOLO) skip the region proposal to achieve real-time detection speed [6]. In general, one-stage detectors sacrifices some accuracy for speed [6, 76].

1.4.3 Single Shot Detector (SSD)

Single Shot Detector (SSD) makes both the objectness and class probability predictions directly in one shot [6, 67]. It has three main components [6, 67]:

- The base network, which is VGG-16 in the original paper. It also uses anchors called priors like in Faster R-CNN. But the network sends the bounding box offsets

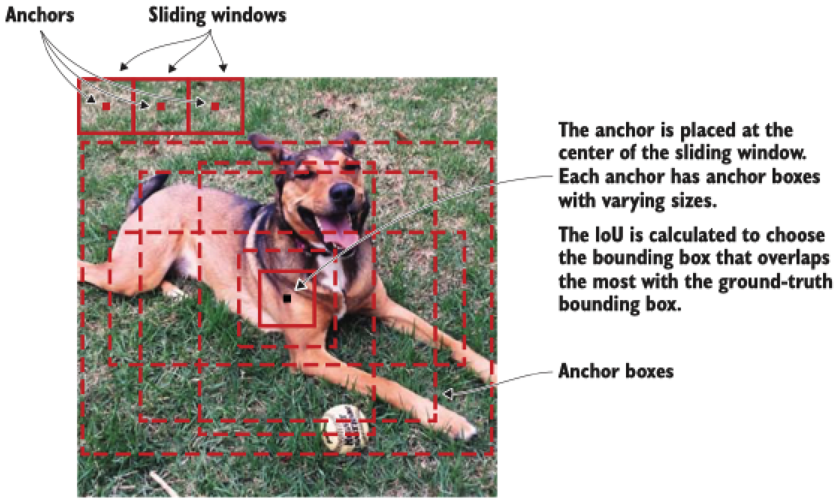

Figure 21: Anchors in Faster R-CNN. Source: [6].

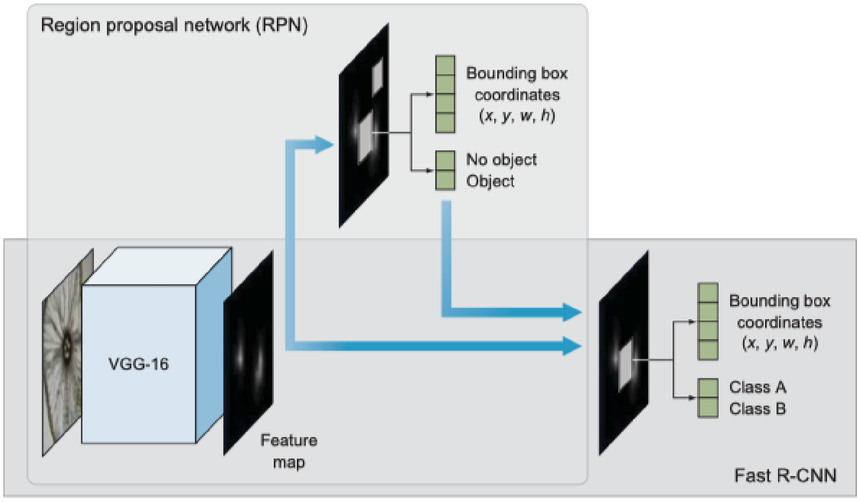

Figure 22: Faster R-CNN. Source: [6].

and the class scores to NMS directly when it finds a bounding box that contains the object features.

- Multi-scale feature layers, which are convolutional layers that decrease in size progressively to detect objects at multiple scales. The resolution of feature maps decreases as the CNN reduces the spatial dimension.
- NMS.

In short, SSD300 (300 x 300 input size) is able to achieve real-time detection with 59 FPS, while SSD512 outperforms Faster R-CNN in terms of mAP [67].

1.4.4 You Only Look Once (YOLO)

Like the R-CNN family, the YOLO family has been going through a series of improvements since the original YOLO paper was published. YOLO is also a one-stage real-time

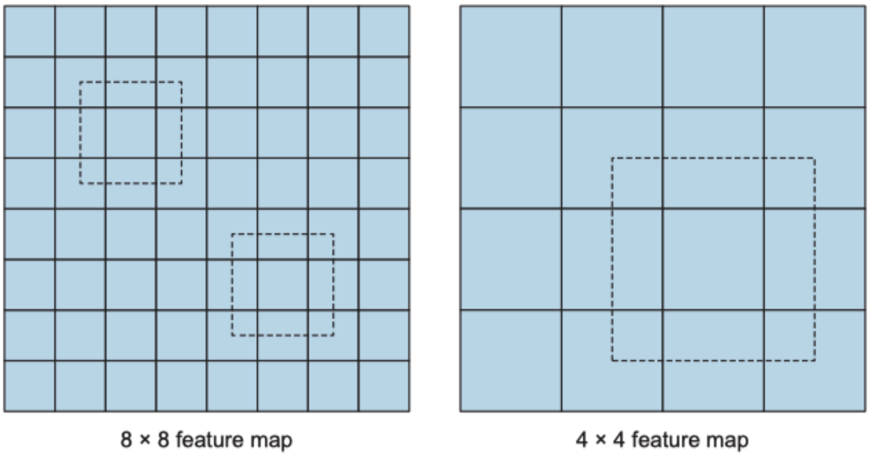

Figure 23: Multi-scale feature maps in SSD. Higher-resolution feature maps (left) detect smaller objects. Lower-resolution feature mpas (right) detect bigger objects Source: [6].

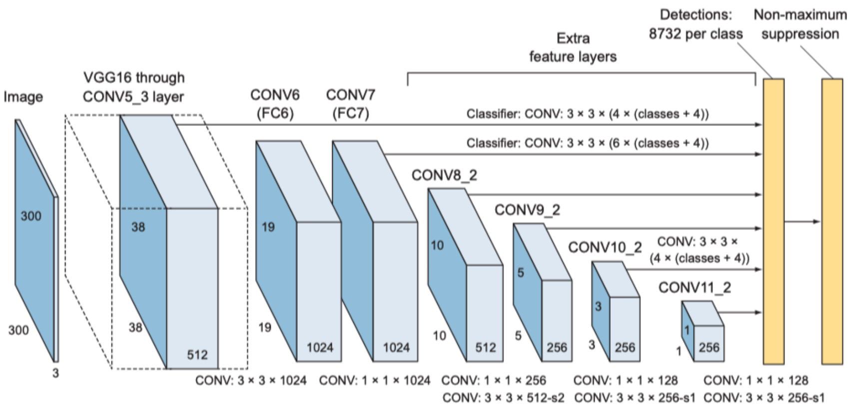

Figure 24: SSD. Source: [6].

detector simlar to SSD. YOLOv1 [63] introduces the general architecture; YOLOv2 [68] adds anchors similar to Faster R-CNN and SSD; YOLOv3 [69] further refines the architecture and the training process; YOLOv4 [70] makes use of some universal object detector features called ”bag of freebies” and ”bag of specials”; YOLOv5 [71] is under active development and authors have yet to publish a paper.

YOLO divides the image into a grid, and a grid cell is responsible for detecting an object if the center of the object is inside the cell [6, 63]. The backbone network is called DarkNet, which is inspired by GoogLeNet [6, 63].

YOLOv4, a bleeding-edge detector introduced in 2020, utilizes numerous new features to improve the performance from YOLOv3, including DropBlock regulariztion [77], Mish activation [78], Self-Adversarial Training [70], etc.

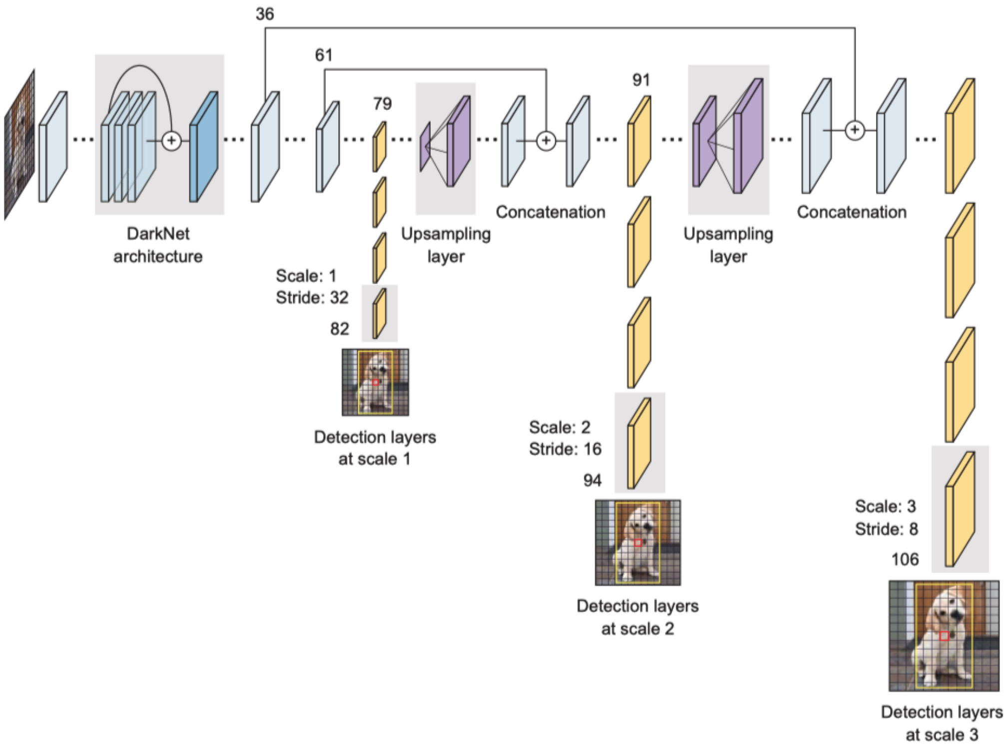

Figure 25: YOLOv3 architecture. YOLO performs detections at 3 different scales. Layer 79 makes a grid of 13 x 13 to detect large objects. Layer 91 makes a grid of 26 x 26 to detect medium objects. And finally layer 106 makes a grid of 52 x 52 to detect small objects. Source: [6].

2 Related Work

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

For fish detection, Qin et al. [17, 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, sliding window with a CNN, YOLOv3, and Xception.

For crab detection, Can 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 have only found research for classification but not object detection [17, 23].

For amphora detection, Pasquet et al. [20, 62] used sliding window with a CNN on high-resolution orthophotos (aerial photos). And this is the only study we have found on amphora detection with deep learning.

3 Data and Methods

3.1 Data

The dataset consists of 50 images (294 objects) in the training set and 7 images (31 objects) in the validation set, which maintains the 90% : 10% ratio for the object count. Two additional images are used as the test set. All the images are obtained through various sources [86, 87, 88, 89, 90, 91, 92, 93, 94, 95] and are annotated with LabelImg [96].

3.2 Model

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

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

The TensorFlow Object Detection API [97, 98] is used to build the model. The model name is ssd_resnet_50_fpn_coco in the TensorFlow Detection Model Zoo [99]. The backbone network is changed from VGG-16 in the original SSD paper to ResNet-50 (50-layer ResNet) (see section 1.2.2), as it improves the mAP significantly [45]. Similar to the approach from Can et al. [84] for crab detection, Feature Pyramid Network (FPN) is added to improve the detection at different scales [85].

3.3 Model Training

Transfer learning [100] is used as the model is pretrained on the COCO dataset, which means the weights and biases are restored instead of being randomly intialized and the model does not have to learn the low-level features from scratch [26]. num_classes, batch_size, and num_examples in eval_config are adjusted to 1, 8, and 7 respectively in the configuration file, while the rest of fields stay unchanged. The images are reshaped to 640 x 640 with the fixed_shape_resizer. The activation function is ReLU6, a modified version of ReLU that caps the maximum value at 6 and thus encourages the model to learn sparse features earlier [101]. We use L_2 regularization [102] to reduce overfitting [103]. Data augmentation [26, 42] - a method to artificially enrich the dataset by randomly flipping horizontally and cropping the images - is also used as a regularization technique. We choose the momentum optimizer [104] as it converges faster than Stochastic Gradient Descent [105] by keeping accelerating to the optimum [26]. Learning rate warmup and learning rate scheduling by cosine decay are also added to speed up the convergence, so that the learning rate starts small, increases gradually, and then decreases after reaching the maximum [26, 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 main metric mAP_{coco} (see section 1.4.1) from our model is 0.238, while the traditional mAP_{50} is 0.503. The documented mAP_{coco} in the TensorFlow Detection Model Zoo for ssd_resnet_50_fpn_coco is 0.35, which is measured 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 does not detect more than 90.3 percent based on the visual inspection of the validation images in figure 26.

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

- Our dataset is very small. As we have mentioned before in section 1.3, the COCO dataset has 328 thousand images compared to our 57 images plus 2 test images. For Pasquet et al., they split one orthophoto (38000 x 15000 pixels) into 400 x 400 images and used 25% of them for training, which means the training set alone is around 890 images.
- Our dataset is a lot 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 the scale. The diversity of our dataset makes it harder to achieve a high mAP, but it also encourages the model to generalize better.
- Underwater object detection is more challenging than general object detection performed by the COCO dataset. As we have mentioned before in section 1.1.2, the same challenges also apply for amphora detection. In fact, the undetected amphoras in figure 26 are almost all either broken, or partially buried in sand, or blocked by suspended particles. We only define one class for all amphoras even if they are broken, while Pasquet et al. defined 2 separate classes for the head and the body to better detect broken instances.

We also ran the model on 2 test images shown in figure 27. To our surprise, the model managed to detect the amphora in the second image even though it is so broken that only half of it is present. As for the first image, none of the amphoras was detected as expected. The size of the first image is only 709 x 411, while it has hundreds of amphora instances. This means that they are mostly small instances defined by the COCO metric, which our training set does not include. It is a known issue that small objects are significantly harder to detect than medium and large objects, mainly due to the low resolution and thus 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 difficult for object detectors since the objects overlap with each other so strongly that it is even impossible for human experts to differentiate the instances [113, 114, 115]. In fact, we tried to split the image and add parts of it to the training set. However, the inter-occlusion of the intances caused many labeling errors, and the resolution of each instance was so low that the training could not converge to a solution.

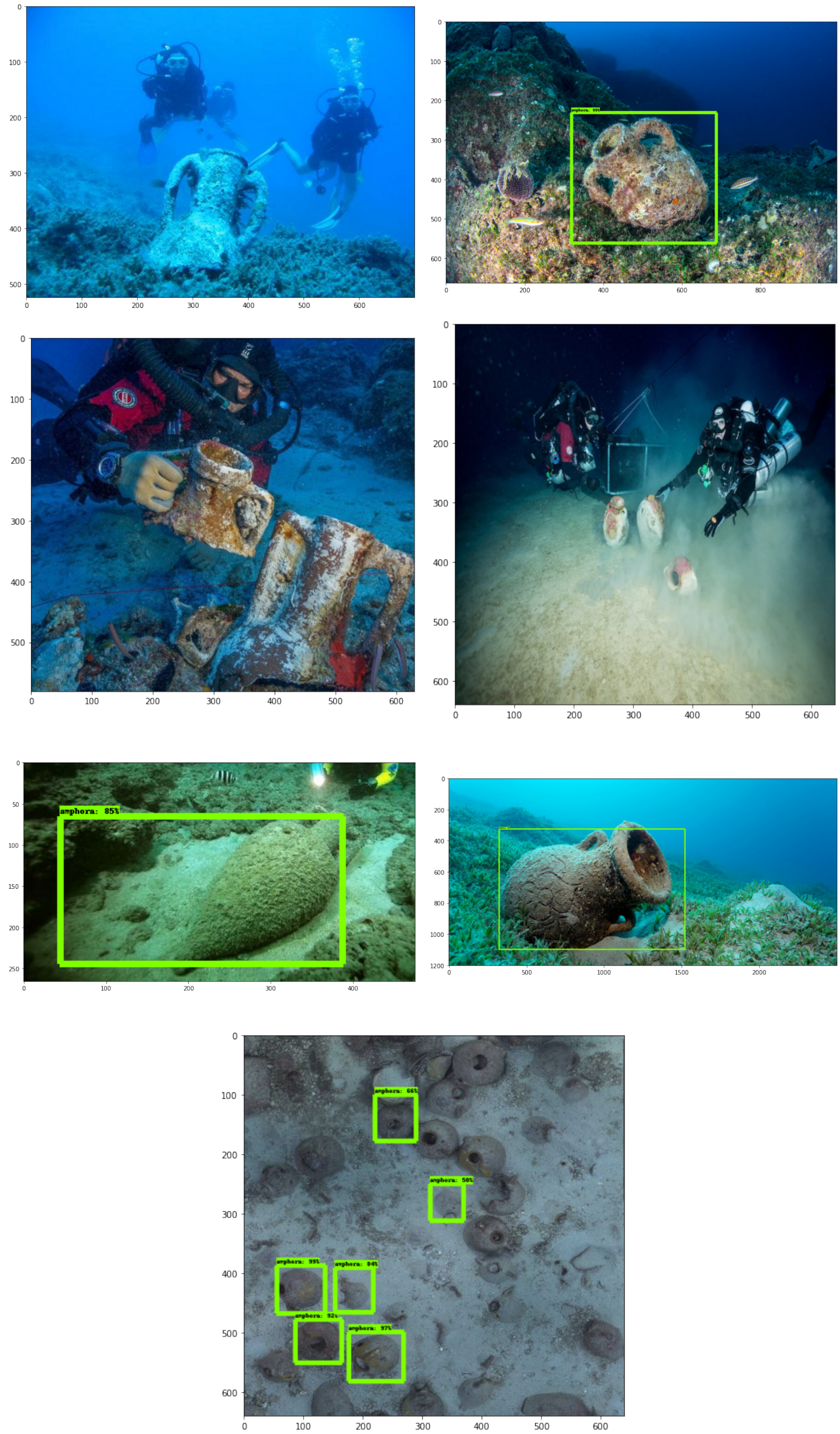

Figure 26: The validation images. Source: [87, 88, 89, 90, 91, 92, 93].

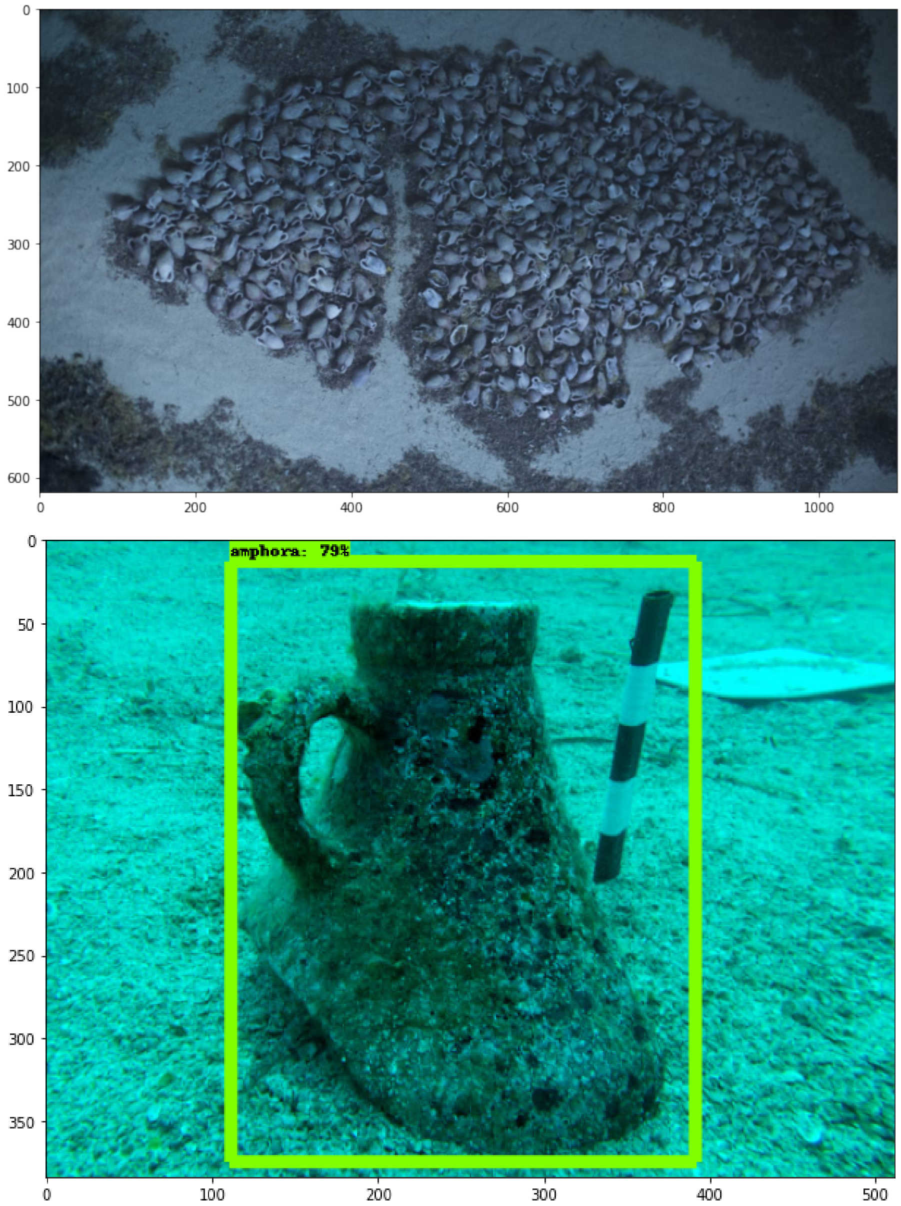

Figure 27: The test images. Source: [94, 95].

5 Conclusions

6 Future Work

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] Jacques Yves Cousteau. "Fish men discover a 2,200-year-old Greek ship". In: *National geographic* 105.1 (1954), pp. 1–36.

[4] Anthony J Parker. "Shipwrecks and ancient trade in the Mediterranean". In: (1984).

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

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

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

[8] 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.

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

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

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

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

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

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

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

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

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

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

[30] 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.

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

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

[33] 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.

[34] 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.

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

[36] 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.

[37] 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.

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

[39] 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.

[40] 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.

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

[42] 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.

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

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

[45] 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.

[46] 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.

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

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

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

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

[55] Ebrahim Karami, Mohamed Shehata, and Andrew Smith. “Image identification using SIFT algorithm: Performance analysis against different image deformations”. In: *arXiv preprint arXiv:1710.02728* (2017).

[56] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “Surf: Speeded up robust features”. In: *European conference on computer vision*. Springer. 2006, pp. 404–417.

[57] Edward Rosten and Tom Drummond. “Machine learning for high-speed corner detection”. In: *European conference on computer vision*. Springer. 2006, pp. 430–443.

[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] 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.

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

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

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

[68] 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.

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

[70] 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).

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

[72] 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.

[73] 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.

[74] Christopher Riggio. *What’s the deal with Accuracy, Precision, Recall and F1?* 2019. URL: https://towardsdatascience.com/whats-the-deal-with-accuracy-precision-recall-and-f1-f5d8b4db1021 (visited on 04/22/2021).

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

[76] Tsung-Yi Lin et al. “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[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.