FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Comparative Study of Machine Learning Techniques for Underwater Visual Object Recognition

António Pedro Oliva Afonso

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Andry Maykol Gomes Pinto

Second Supervisor: Aníbal Castilho Coimbra de Matos

July 16, 2019

Resumo

Esta dissertação pretende avaliar o desempenho de algortimos de aprendizagem computacional aplicados à classificação de imagem quando aplicados em ambiente subaquático, após terem sido treinados exclusivamente com informação adquirida em meio aéreo. Dois tipos de algoritmo de aprendizagem computacional foram testados: métodos mais convencionais, tais como redes neuronais e Support Vector Machines e métodos modernos, com redes neuronais convolucionais, especificamente a arquitetura Inception-V3.

Um novo dataset, HEIMDACA, composto por 9600 imagens de 8 objectos em dois meios distintos, aéreo e aquático é também apresentado.

Primeiramente foi estabelecido um patamar de comparação na fração aérea do dataset, resultando num True Positive Rate (rácio de acerto) para os métodos convencionais de 63.1% e de 98.4% para a rede neuronal convolucional. Seguidamente, estes mesmos modelos foram validados no meio aquático. Enquanto que o True Positive Rate das redes neuronais convolucionais sofreu apenas uma queda de 15.54%, os métodos mais convencionais caíram em 40.86%. Para verificar em quanto os modelos poderiam ser melhoras com o acréscimo de informação adquirida no meio aquático, procedeu-se a um treino incremental desses modelos, o que resultou em um aumento para 40.3% nos métodos convencionais e para 98.6% na rede neuronal convolucional.

Dos resultados pode-se concluir que as redes neuronais convolucionais, com as suas camadas de aprendizagem da representação (Representation Learning), supera completamente os algortimos mais tradicionais, tendo um decréscimo menor quando alterada do meio aéreo para o meio aquático e um maior desempenho no geral.

Abstract

This dissertation aims to evaluate the performance of machine learning image classification techniques when applied on the aquatic domain, after being trained exclusively with images from the aerial domain. Two types of machine learning algorithms are tested: Conventional Machine Learning, which comprises Neural Networks and Support Vector Machines, and Convolutional Networks, the Inception-V3 architecture, in specific.

A novel dataset, HEIMDACA, composed of 9600 images of 8 objects in two different domains, aerial and aquatic, is also presented.

A baseline for both methods was first established for the aerial fraction of the dataset, resulting in a best True Positive Rate of 63.1%, for the Conventional algorithms and 98.4% for the Convolutional Network. Those same models were then validated on the aquatic domain. While the True Positive Rate of Convolutional Network decreased only 15.54%, the Conventional methods suffered a drop of 40.86% in the True Positive Rate. To verify how much the models could improve by adding information from the aquatic domain, the classifiers were incrementally trained with images from the aquatic part of the dataset, increasing the True Positive Rate of Conventional methods to 40.3% and the Convolutional Network to 98.6%.

From the results it can be concluded that the Convolutional Network, with its Representation Learning layers, completely outperformed the more conventional machine learning algorithms, having a smaller decrease in True Positive Rate from the aerial to the aquatic domain and a greater accuracy overall.

Agradecimentos

Ao meu Orientador, Andry Pinto

Aos meus pais

Aos meus avós

Ao meu irmão

Aos meus amigos

À Tuna de Engenharia da Universidade do Porto

Pedro

"When you're fundraising, it's AI.
When you're hiring, it's ML.
When you're implementing, it's logistic regression.
When you're debugging, it's printf()"

Baron Schwartz (@xaprb)

Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
AUV	Autonomous Underwater Vehicle
BoVW	Bag-of-Visual-Words
BoW	Bag-of-Words
ConvNet	Convolutional Neural Network
GPU	Graphics Processing Unit
NN	Neural Network
ROS	Robot Operating System
ROV	Remotely Operated (Underwater) Vehicle
SIFT	Scale-invariant feature transform
SGD	Stochastic Gradient Descent
SURF	Speeded-Up Robust Features
SVM	Support Vector Machine

Chapter 1

Introduction

1.1 Motivation and context

In the past decade, machine learning has been seen by some as a panacea, a solution for all problems, and by others as a curse, for fear of the increasing automation of jobs, rendering some professions useless and unnecessary. What can be agreed on is that words such as "machine learning" or "artificial intelligence" have been attracting more and more attention, being used by businesses as their defining characteristic and differentiating factor, be it due to truly innovative advances and applications or just as a marketing strategy to attract investors.

One branch of this more modern machine learning is its application for visual detection and classification tasks, as seen on Tesla's Autopilot[1] or BMW's Intelligent Vision [2], cars which detect road signs and display them in a Heads-Up Display inside the car, available on products already in commercialization. Another use of this technology is in the *Amazon Go*[3] stores, a chain of partially automated convenience stores operated by Amazon, where there are no cashiers or checkout stations, the customer just grabs the needed items and their value is added to their virtual checkout cart on a smart-phone application, which is then charged when the customer leaves the store.

Exploration of the seas would greatly benefit from an increased degree of autonomy. For example, state-of-the-art marine gravity models [4] can be useful to analyze underwater geological structures (fractures, rifts, abyssal hills) with a minimum of 6km of extension and, if a more detailed analysis is required, a vessel with proper imaging equipment [5] has to be on site, involving great cost and human effort. Also, marine life studies could be done in shorter time and with less manpower. For example, Edgar and Stuart-Smith (2015) [6] present a reef fish population assessment made all over the world. The data was collected through direct visual analysis and required a team of over 100 scuba divers. A network of Autonomous Underwater Vehicles (UAV) network would have made this study even more complete, due to the large area the vehicles could cover and their allowed dive time, which would be higher than that of an human diver. However, the models these UAV could use require some *a priori* knowledge of things to detect and classify, they need

[1] https://www.tesla.com/autopilot

[2] https://www.bmw.co.uk/bmw-ownership/connecteddrive/driver-assistance/intelligent-vision

[3] https://www.amazon.com/b?ie=UTF8&node=16008589011

to know how the objects or animals look. Modern machine learning algorithms require a great amount of data to be effective and robust, data which, in an underwater context, is costly and time-consuming to collect. If classification models with an underwater purpose could be trained with information collected on land, it would be advantageous both from a human effort and economic point of view.

Thus, this dissertation aims to analyze and compare several visual object classification models trained with images of objects collected on land and then evaluate their performance when applied to object classification in an underwater environment.

1.2 Objectives

This work aims to:

- Present a novel dataset, composed of 4800 images of 8 different objects in two different environments, in and out of water, suitable for the comparison of image classification techniques when used in those two different environments. The dataset also includes segmentation masks and bounding boxes of the objects contained in the images, amplifying the dataset's usefulness for object detection and segmentation tasks.

- Test both Neural Networks and Support Vector Machines (conventional machine learning techniques) using images from the aerial part of the proposed dataset followed by a verification of how those same models behave in an underwater environment when trained exclusively with images taken out of water.

- Measure the benefits of incrementally training the models with underwater images. By adding images taken in the same environment, it is expected the models will improve considerably, as the imaging conditions where the images were take are roughly the same regarding lighting and blur amount (which is an important factor to take into account in underwater imaging).

- Repeat this analysis using a Convolutional Neural Networks architecture from the state-of-the-art in image classification.

- Make a comparison between the reliability of air-trained Conventional Machine Learning techniques and Convolutional Neural Networks when applied to an underwater classification task.

1.3 Scientific Contributions

On the context of this thesis, an abstract for a paper titled *"Underwater Object Recognition: A Domain-Adaption Methodology of Machine Learning Classifiers"* was submitted to the OCEANS

[4]https://seattle19.oceansconference.org/

2019 Seattle conference [4]. As of June 2019, notice of acceptance is yet to be received, being expected before the end of July.

1.4 Structure of this document

Chapter 2 gives some context on the particularities of imaging on the underwater environment, highlighting some of the issues that might affect visibility and consequently image classification methods. A theoretical overview is given on the techniques used, both Conventional Machine Learning as well as Deep Learning and Convolutional Neural Networks.

Chapter 3 introduces the novel dataset collected, mentioning how its collection proceeded and giving some information on the equipment used and its characteristics.

Chapter 4 explains the procedure taken for evaluation of the Conventional Machine Learning techniques and the influence of the underwater environment on both Neural Networks and Support Vector Machines is quantified

Chapter 5 explores a specific architecture of the state-of-the-art in image classification, evaluating its performance in both aerial and aquatic environments.

Chapter 6 concludes this work, giving an overview of the work done and lessons learned.

Chapter 2

State of the Art

This chapter will give context to the problem, introducing some challenges that the underwater environment poses. An overview of machine learning classification methods will be given, which facilitate the comprehension of following chapters, focusing on both Conventional Machine Learning (Neural Networks, Support Vector Machines) and Deep Learning (Convolutional Neural Networks) methods. The chapter ends with examples of Machine Learning applied in visual classification tasks in the underwater environment and a critical analysis of the state-of-the-art.

2.1 Marine Vision Characteristics

2.1.1 Light propagation

Light propagation in water is affected by two physical phenomena (Figure 2.1): absorption (loss of power, attenuating light and thus reducing the visibility at a distance) and scattering. Absorption, according to the Lambert-Beer law [1], is different depending on the material through which the light is propagating and increases exponentially with the distance to the light source, causing objects to be less illuminated as distance is increased, limiting the illumination range. Scattering can be divided into forward scattering (light component that is deviated on its way to the camera, resulting in blurring of image features) and backward scattering (light component that doesn't reach the object and is reflected directly to the camera by water or undesired objects, thus limiting contrast of images; this phenomenon can be seen when taking photos with flash of objects too close to the camera, the foreground gets too bright and the background too dark).

Artificial lighting can increase visibility range, but can also aggravate scattering due to the presence of floating particles called "marine snow"[1], causing bright spots on an image. Artificial lighting also tends to illuminate the scene in a non uniform fashion, causing edges to be darker than the center of the image.

Color also suffers an impact from water propagation [1]. Different wavelengths have different absorption rates 2.2, resulting in some colors only being able to propagate up to a certain distance

[1] Image taken from http://thescientificfisherman.com/fish-senses-1-fish-sight/

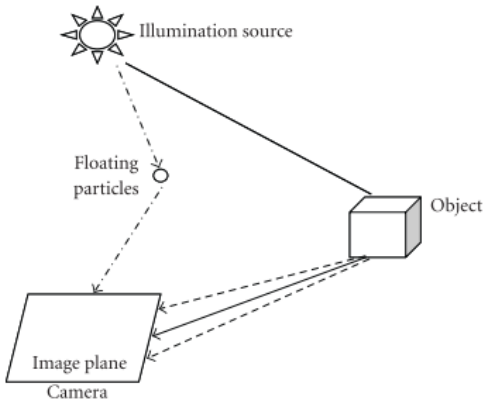


Figure 2.1: The three components of underwater optical imaging. Source [1]

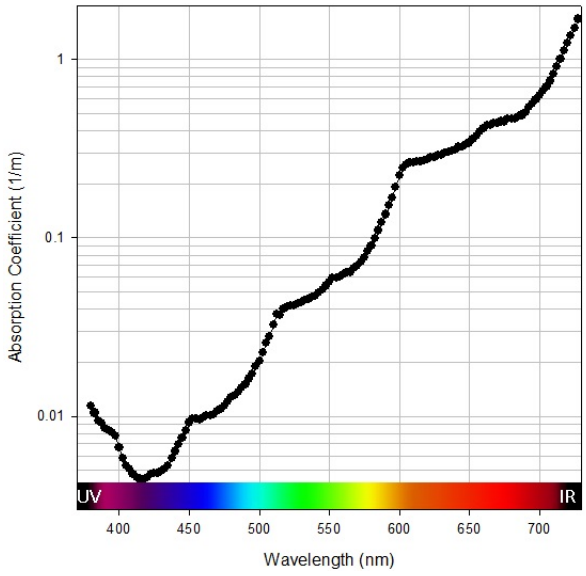


Figure 2.2: Absorption coefficient of water throughout the visible spectrum.[1]

[1]. From Figure 2.2 it can be seen blue has the lowest absorption coefficient, thus it is able to propagate further than red, for example, which has a higher coefficient.

2.1.2 Improving Underwater Vision

To correct these defects, according to Schettini and Chorchs [1], research is divided in two approaches: image restoration and image enhancement and color correction.

Image enhancement makes no use of *a priori* knowledge of the environment, not taking into account any light propagation model or noise functions. Usually faster and simpler than image restoration techniques.

One way of dealing with loss of contrast resulting from backscattering is by using Contrast Limited Adaptive Histogram Equalization (**CLAHE**) [7], which divides the image into several blocks, finds the optimal contrast for each region and avoids overamplification of contrast that might result in excessive noise. However, it can't correct losses in visual quality from the high scattering component in high turbidity waters [8].

Image restoration techniques try to recover the original scene from the captured image and take advantage of explicit knowledge of noise and degradation functions, usually following the Jaffe-McGlamery [9] image formation model. According to the model, the image can be formed by the linear superposition of three components [1]: direct component, forward scattered component and back scattered component.

It usually involves correcting images with empirical information as presented by Liu et al. (2001)[10], for example, where the authors measure the Point Spread Functon (PSF) and Modulation Transfer Function (MTF) of salt water and use the measurements with a Wiener filter to deblur images captured through water paths of different lengths. This method gives satisfactory result, but the authors only tested it in images of simple geometry (rectangular slits) and a single image of fish.

Garcia et al. (2002)[11] present a method to remove lighting inhomogeneities originated from artificial light sources by estimating and then subtracting the illumination field. Their method has been shown to work in a satisfactory way when simulating deep sea conditions (no natural illumination) but solar rays common in shallow waters cause artifacts in the image that couldn't be corrected.

2.2 Conventional Machine Learning

2.2.1 Feature extractors

In theory, an image classifier can be created by using the raw pixel values of an image on a Neural Network or other classifier. However, that wouldn't be too effective or efficient, as there

is a great amount of input data, and not all of it might be relevant to the training of the classifier. The classifier models can also become prohibitively large, resulting in a time-consuming training process. Feature extraction takes the initial set of data and transforms it so as to generate a smaller set of non-redundant and more representative features.

When real (as opposed to synthetically generated) images are used, various aspects have to be taken into account when processing the image since two pictures of the same object can have variations in the lighting conditions, object pose or one of them be affected by noise, which will influence the output of the feature extractor. The more robust the extractor is to these variations, the best the performance of classification tasks will be. Many complex image feature detector have as their basis simpler features such as edges (boundary between two image regions, a set of points with a strong gradient), corners (point-like features) and blobs (wider regions).

Over the past years, different feature extractors have been proposed in the literature, such as SIFT[12], SURF[13] and ORB[14], to name a few. Only SIFT will be presented, as the others aren't used in this thesis.

SIFT Created in 2004 by Lowe [12], SIFT is a feature extractor (Fig. 2.3) and descriptor that was designed having in mind the need for features invariant to several conditions. SIFT features are invariant to image scale and rotation, having robustness to affine distortion, changes in viewpoint and illumination. At the time of its creation the state-of-the-art feature detector for image classification tasks, it withstood the test of time up until 2012, when other feature descriptors took its place, albeit at the cost of efficiency [15].

Despite being a great candidate for use in classification tasks, it's unsuitable for real-time operation, as it is composed of computationally heavy operations, one of them being the evaluation of a single image for interest points at multiple scales, in a progressively blurrier state. Due to that disadvantage, subsequently developed feature detectors, such as the ones mentioned above, aimed at speeding up the feature detection and description process, while maintaining the same performance as SIFT.

2.2.2 Bag-of-Visual-Words

Previous to the BoVW method, feature-based image classification/matching was made by comparing each and every keypoint in an image with a database of keypoints, a computationally expensive task, since images can have hundreds or even thousands of keypoints.

The Bag-of-Keypoints[16] or Bag-of-Visual-Words (BoVW) was inspired in the Bag-of-Words method used in Natural Language Processing where a text is represented as an histogram of the words it contains.

Applying the BoVW method requires a vocabulary to be generated. First, image keypoints (Fig. 2.4) are extracted from a set of images using SIFT or other feature extractor. Using the K-Means algorithm, features are then clustered, aggregating features according to their similarity. The number of clusters, N, of the K-Means algorithm is predefined by the user, corresponding to the number of "visual words" in the vocabulary. As the authors of the original BoVW method [16]

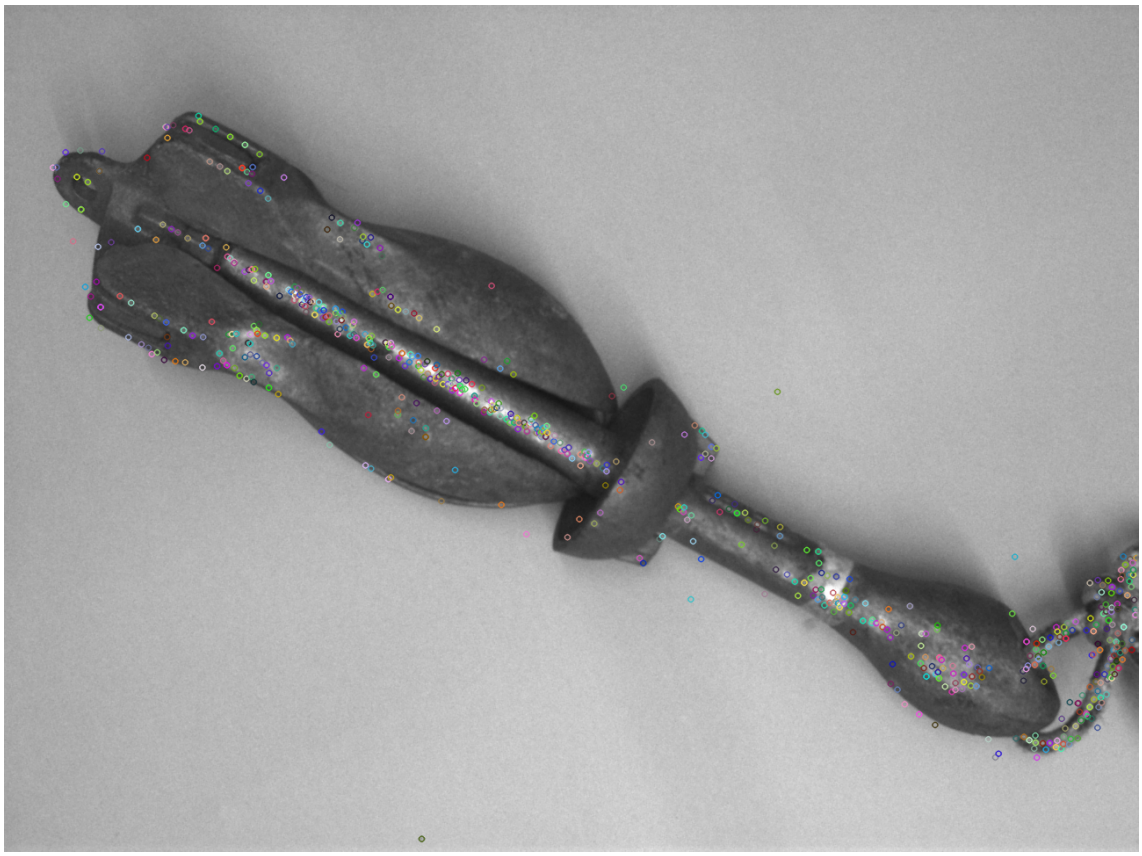


*Figure 2.3: Example of SIFT keypoints on the **anchor** class*

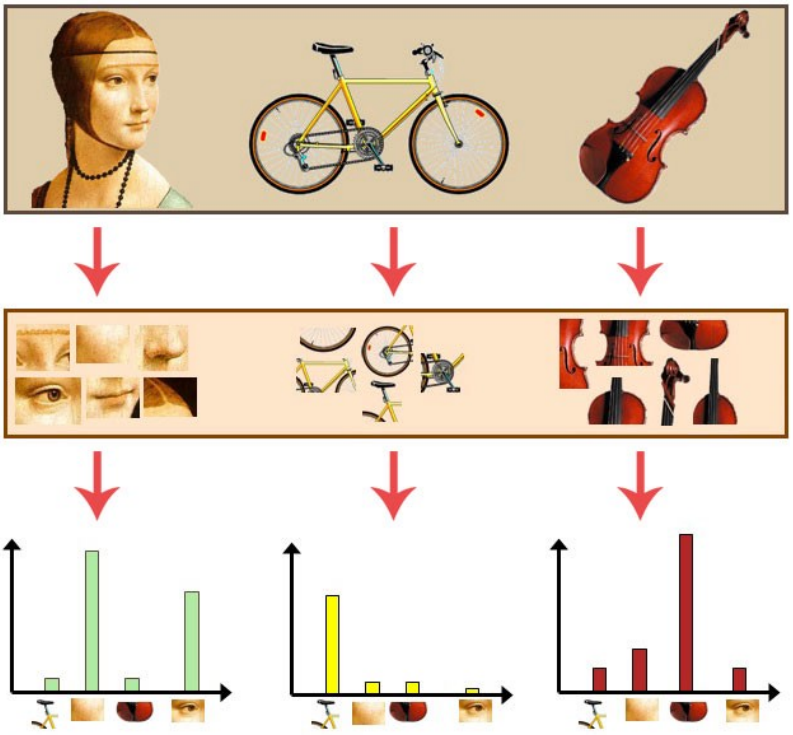


Figure 2.4: Simplified but intuitive visual explanation of BoVW [2]

recommend the vocabulary size "*should be large enough to distinguish relevant changes in image parts, but not so large as to distinguish irrelevant variations such as noise*". However, no further guidelines are given on the choice of vocabulary size.

Next, for the example of image classification, in order to train features are extracted from the training images using SIFT or a similar algorithm. Then, each keypoint is assigned a cluster (word), creating a N-dimensional feature vector with the number of occurrences of each visual word. This is done for every image in the training data, and the resulting feature vectors are used as input data to train the classifiers.

The introduction of a fixed-size feature-vector makes possible the use of general purpose classifiers, as well as reducing the amount of data involved in the classification task. his step reduces the dimensionality of the data at the input of the classifier, resulting in a fixed-sized input vector and in a lesser computational complexity of the classification task, while at the same time bringing some robustness to keypoints not favorable to the task (e.g, background clutter).

2.2.3 Neural Network

The concept of Neural Network had its origins in a 1958 paper by Frank Rosenblatt [17], where he presents a simplified mathematical model of how neurons in the brain work. This neuron, the Perceptron (Fig. 2.5), takes multiple inputs which are multiplied by a weight and sums them. If

[2] Image taken from https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb

this sum is greater than a certain threshold, the output is 1 (neuron firing) and if it's below the threshold the output is 0 (neuron doesn't fire).

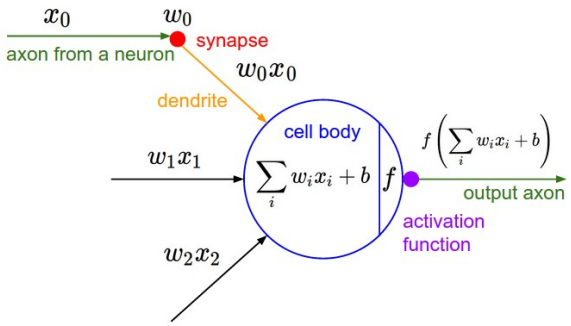


Figure 2.5: Rosenblatt's perceptron. x represent the inputs and w the weights[3]

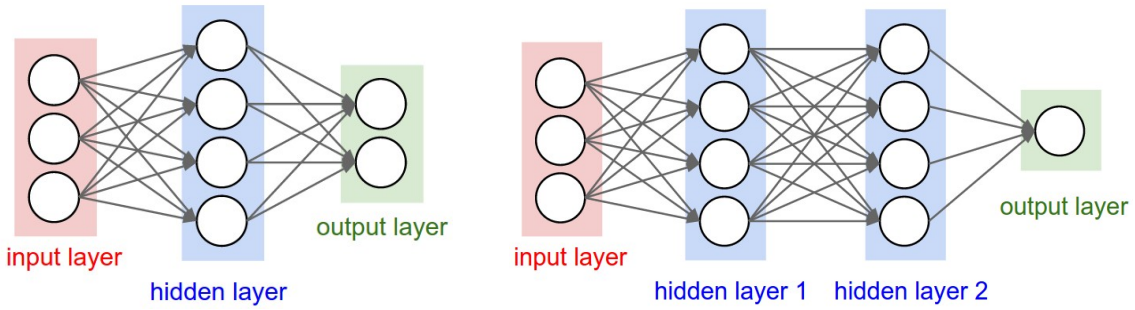
While the Perceptron can mimic (or learn) some simples boolean functions, such as the AND and OR, it is limited to patterns that are linearly separable, being incapable of learning the XOR function, for example [18]. However, by combining several of these simple Perceptrons in a single or multiple layers, it is possible to create what is known today as a Multilayer Perceptron or Neural Network.

Fig. 2.6 shows examples of *Fully Connected* Neural Networks (NN) which means a NN where neurons from a layer are connected to all nodes of the previous and the next layers, the exception being the input and output layers, due to being the first and last layer, respectively.

Typical structure of a Neural Network

- ***Input Layer:*** Has a neuron (unit) for each input value to be used. E.g. If 1000 features are being used for a certain task, the input layer will have 1000 units.

[3]Image taken from: cs231n.github.io/neural-networks-1/
[4]http://cs231n.github.io/neural-networks-1/



(a) 1 hidden layer with 4 neurons (units); 2 output units

(b) 2 hidden layers with 4 neurons (units) each; 1 output unit

Figure 2.6: Examples of Neural Networks[4]

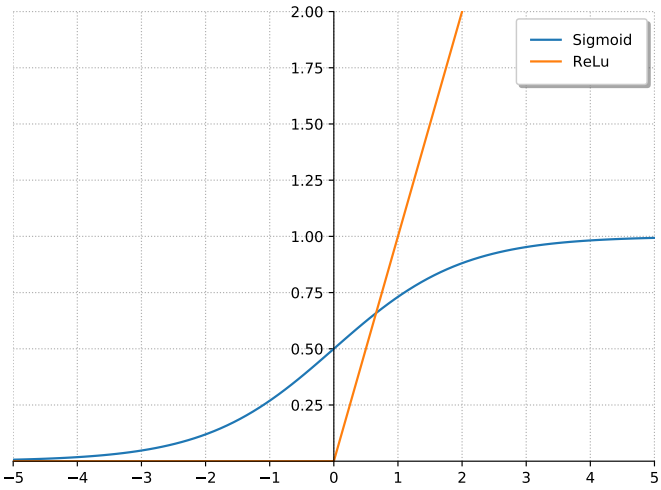


Figure 2.7: Plot of two Activation Functions

- ***Hidden Layer:*** Layers that are situated between the input and output layers. A NN can have 0 hidden layers, making it a 1-layer Neural Network (output layer).
- ***Output Layer:*** Last layer of the NN which, in a classification problem, consists of a number of neurons equal to the number of objects it has to identify, or a single neuron, in a binary classification problem.

Activation Functions Associated with each neuron is a function that takes the result of the dot product of the inputs and the neuron's weights and applies a fixed mathematical operation, the Activation Function (Fig. 2.7). Several activation functions have been used in the literature, each having its advantages and disadvantages.

Sigmoid A special case of the logistic function, has seen frequent use as an abstraction of the behaviour of a biological neuron due to its output having a limited range: from 0, representing a neuron not firing, to 1, the neuron firing at its maximum rate.

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

However, it has one major drawbacks: the weight update (learning) process takes into account the derivative of the output of the activation function in two sequential iterations [19]. If the sigmoid saturates at either 1 or 0, the derivative will be very small, resulting in a minimal change of the weights. That saturation effect can propagate to the whole network, which in turn will stop the learning process. This is called the "Vanishing Gradient Problem".

ReLU Popularized by Krizhevsky et al. (2012)[20] in his image classification task using Convolutional Neural Networks, the non-saturating non-linearity in Equation 2.2 doesn't involve expensive operations such as the exponential which in turn accelerates the network training process up to a factor of 6 compared to the sigmoid activation function (Eq. 2.1). However, the rectifying behavior of this function can bring the "death" of a neuron. If the learning rate is set too high, a

large gradient can update the neuron weight in such a way that the neuron becomes irreversibly unusable, it "dies". Neurons with this activation function are called Rectified Linear Units (ReLU).

$$f(x) = max(0,x) \tag{2.2}$$

This activation function has several variations that try to correct this "dying ReLU" drawback, namely the *Leaky ReLU* [21], in which there is a small slope in the negative input side of the ReLU 2.3 and the Parametric ReLU (PReLU) [22], where the coefficient of leakage (0.01 in the Leaky ReLU) is also a parameter to be learned by the network 2.4.

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases} \tag{2.3}$$

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{otherwise} \end{cases} \tag{2.4}$$

Learning Process Two strategies used to train Neural Networks are ***Batch Learning*** and ***On-line Learning*** [18]. In ***Batch Learning***, weights are only updated after presenting the number of samples that make a batch. For example, a *batch size* of 32 means that 32 samples of the training data will be shown to the network and only after averaging the error of those 32 samples the weights will be update. An *Epoch* has passed when all the training examples have been used to train the network. The network is then trained for several epochs, shuffling the training data between each iteration, for a maximum number of epochs or until the loss function doesn't vary more than a given tolerance. With ***On-line Learning***, weights are updated after each given example in the training data, making the parallelization of the training process impossible, resulting in a slower method, comparing to batch learning.

The learning process of Neural Networks is constituted by the minimization of a loss function through the successive calculation of gradients. The example given will be that of ***Supervised Learning***, when the true label of the training data is known. For a specific training occurrence, the network calculates a prediction which is then used to calculate the value of the loss function, by taking into account the predicted value and the real value. The resulting error is then propagated through all the layers of the network, using ***Backpropagation***, updating the weights of the network through gradient optimization algorithms, such as ***Stochastic Gradient Descent*** and ***ADAM***.

Stochastic Gradient Descent Equation 2.5 presents a simplification of the formula used by the Stochastic Gradient Descent (SGD)[18] optimizer, where i is the number of the iteration, α the learning rate and ∇J_i the gradient of the loss function at iteration i. The learning rate is a tunable parameter which determines how much to update the weights with respect to the gradient of the loss function. If α is too small, the network will take a long time to converge to its point of minimum gradient. If α is too high, the weights of the network might oscillate, not reaching

convergence.

$$W_{i+1} = W_i - \alpha \nabla J_i \tag{2.5}$$

ADAM In order to solve some of the convergence issues that result from a bad choice of learning rate α, some adaptive algorithms have been proposed. ADAM (Adaptive Moment Estimation)[23] combines the best of two other algorithms, having different learning rates for each weight (AdaGrad) instead of a single global learning rate like SGD and using the first and second moments of gradient to adapt the learning rate (RMSProp).

2.2.4 Support Vector Machines

Support Vector Machines (SVM)[24] are a popular machine learning method, being used successfully [25] in image classification applications on a large-scale (1000 classes), with a great amount of data (over 1.2M data samples), as well as regression and other tasks. Originally a binary classifier, following the idea of mapping the input-vectors to a very high-dimension feature space and then constructing a linear decision surface that separates the two classes while minimizing an error function (soft margins).

A variety of different classifiers can be obtained from SVMs by performing what is called a "kernel trick" [26], changing the linear decision surface to a non-linear one, with the most common kernels being the Polynomial Kernel and the Radial Basis Function (RBF), equations 2.6 and 2.7, respectively, where x and y are vectors on the input space, c and γ tunable parameters and d the polynomial order.

$$K(x,y) = (x^T y + c)^d \tag{2.6}$$

$$K(x,y) = e^{-\gamma \|x-y\|^2} \tag{2.7}$$

2.3 Deep Learning

Conventional machine learning techniques required considerable feature engineering [27] to be effectively used, as using data in its raw form (pixels, in the image example) was inefficient and brought unsatisfactory results and choosing the right kind of feature also required some expertise and intuition in both the task and nature of the data.

Instead of hand-engineering features, researchers aimed at making feature creation the initial step of the whole learning process, feeding a Neural Network, for example, with the raw data, which would then learn in the first few layers the features to be used deeper into the network. This

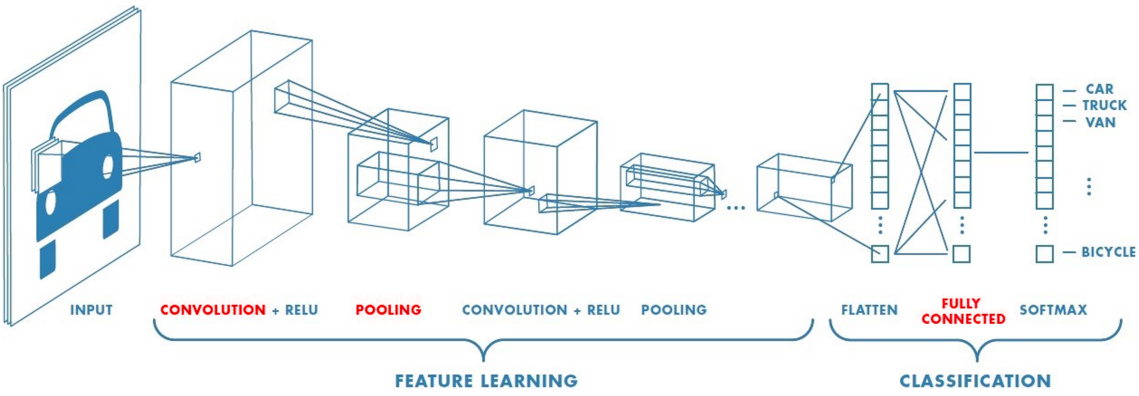


Figure 2.8: Simplified representation of a ConvNets[5]

is called ***Representation Learning*** and was found to happen in (feedforward) Neural Networks with many hidden layers, creating a so-called **deep-learning architecture** [27].

These deep neural networks, despite their ability to obtain good results in some tasks, due to their fully-connected nature and great amount of trainable parameters, often overfitted the training data, thus lacking generalization ability[28]. One particular type of deep feedforward network, however, was found to be much easier to train due to its lack of full connectivity and had better generalization capabilities, the Convolutional Neural Network (ConvNet) [28] (Fig. 2.8).

2.3.1 ConvNet layers

ConvNet architectures are composed of several types of building blocks, the three main ones being:

Convolutional: Its parameters are a set of learnable filters, of number and size chosen by the designer of the network. To generate the input for the following layer, each filter is convolved across the width and height of the input image, with a predefined stride (separation between filter applications), and along its depth (in the case of RGB images), creating an activation map of the filter response. The activation maps for all filters responses are then stacked, creating a block of data. The learned filters are the features, corresponding to the representation (feature) learning part of deep-learning in ConvNets (Fig. 2.9).

Pooling: Pooling layers are commonly introduced between Convolutional Layers in order to progressively reduce the size of the representation and consequently improve computational efficiency. For example, in Fig. 2.10 a max pool layer with size 2x2 and stride 2 is applied to a 4x4 slice, resulting in a $(4-2)/2+1=2$ sided area. Pooling layers can perform other functions such as average pooling or L2-norm pooling.

[5]Image taken from: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

[6]Image taken from: https://anhvnn.wordpress.com/2018/02/01/deep-learning-computer-vision-and-convolutional-neural-networks/

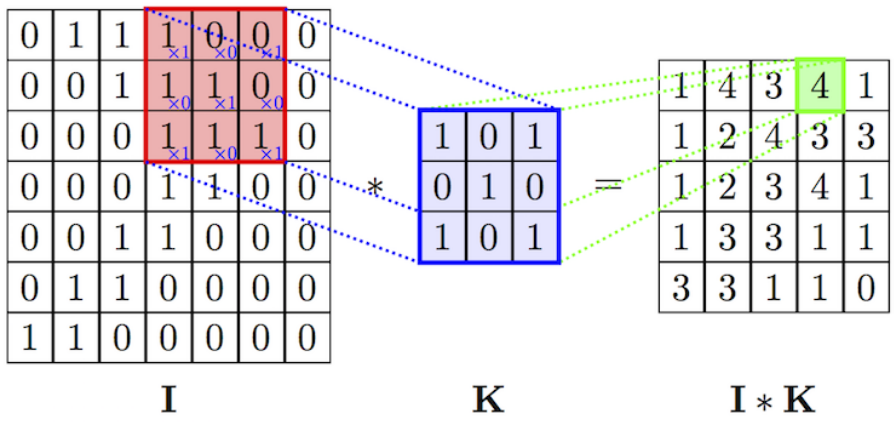


Figure 2.9: Convolution between a layer and a filter[6]

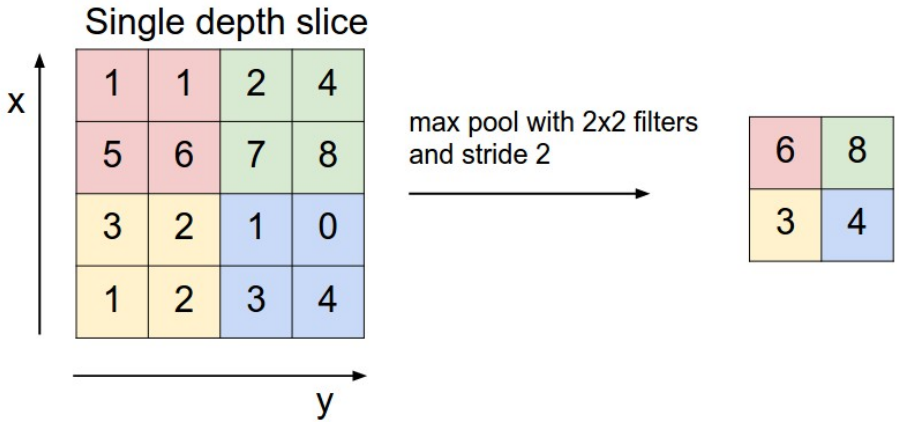


Figure 2.10: Max Pooling layer with a 2x2 size and stride 2[7]

Fully-Connected (dense): Usually used in the deeper stages of networks, it's identical to the hidden layers used in Multilayer Perceptrons, connected to all nodes of the previous and following layers.

2.3.2 Examples of Deep Architectures

All given examples will be of architectures used for the purpose of image classification, due to the nature of this thesis. A good benchmark for this task was the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC), where various research groups competed, presenting their unique approach to image classification, with hopes of being the new state-of-the-art. The last edition of the challenge was in 2017[8], having since 2016 scores surpassed human performance on that specific dataset, consisting of images of different objects in varying contexts for a total of 1000 classes to be identified. There's also an ever-increasing dataset from where images used in this challenge were taken, called ImageNet, which will be introduced further ahead 3.1.1.

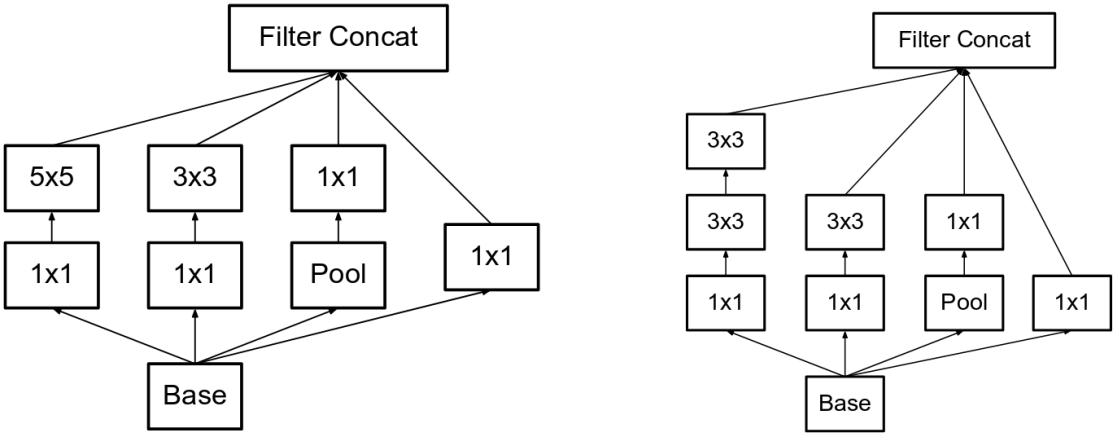
Up until 2012, winners of the challenge were approaches based on the state-of-the-art in image classification of that time: classification with SVMs using SIFT features combined with different means of representation such as Local Binary Pattern or Fisher Vectors [29]. The winner of the 2012 edition, however, with the use of a Convolutional Neural Network trained using 2 GPUs, presented an architecture that greatly surpassed the state-of-the-art of that time, achieving a top-5 error of 15.3%, compared to the 26.3% of the second-best entry. Called AlexNet, the network brought ConvNets to the computer vision research community's attention, and winners of subsequent editions of the challenge (thus the state-of-the-art in image classification) were all ConvNets, each iteration bringing new ways of making networks more efficient and with better performance.

A small historical overview of the evolution of Convolutional Neural Networks will be presented, not meant as an exhaustive analysis, but to give some context.

LeNet Historically, the first convolutional neural network for image an classification application [28], used for numerical handwriting recognition. At the time it was published, in 1990, the machine learning community had its skepticism regarding the capabilities of neural networks, resulting in this network not receiving its due attention.

GoogLeNet/Inception-v1 Winner of the 2014 edition of ILSVRC, achieved a top-5 error rate of 6.67%, very close to the human performance of 5.1% [9]. Consisting of 22 layers of operations, its convolutional layers are of very small size, resulting in an architecture with 4 million parameters to be learned, in comparison with AlexNet's 60 million parameter. The authors also introduced the concept of Inception module (Fig. 2.11a), and GoogLeNet and other iterations of this network were based in it.

[7] https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/
[8] http://image-net.org/challenges/beyond_ilsvrc.php
[9] http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/



(a) GoogLeNet (Inception-V1) inception module

(b) Inception-V3 inception module with layer factorization

Figure 2.11: One of the improvements of Inception-V3 over the original architecture. Source [2]

VGGNet The second-best entry of the 2014 editions of ILSVRC, brought attention to the community because of its uniform structure [30]. However, consisting of 138 million parameters, took 2 to 3 weeks of constant training in 4 GPUs, making it a very computationally heavy and unoptimized ConvNets. Other iterations have been studied by varying the number of layers, but performance still falls short compared to GoogLeNet.

Inception-v3 The third iteration [2] of the GoogLeNet architecture. Despite scaling up a network in computational cost and size brought immediate quality gains, efficiency and parameter count were still an enabling factor for some applications like mobile vision and big data. While the VGGNet can be considered a bruteforce solution (performance can be obtained by adding more layers), the Inception-v3 followed the paradigm of efficiency. The authors introduced the concept of layer factorization e.g., a $5x5$ filter is $25/9 = 2.78$ more computationally expensive than a $3x3$ filter so they replace a $5x5$ convolutional layer with 2 $3x3$ layers in series, resulting in a 28% reduction in complexity. This architecture didn't compete in the ILSVRC but when tested with the same dataset it achieved an error rate of 3.58%.

2.4 Machine Learning in Underwater Robotics

Autonomous and automatic systems can replace direct human intervention in underwater applications, reducing both costs and risks of sending humans to possibly hostile environments. They also allow for cheaper and safer exploration of more extreme benthic zones of the ocean, where pressures are too high for a scuba diver or a prolonged dive in atmospheric diving suits [31].

A skilled diver requires training, support equipment and personnel (boat plus crew) and for safety reasons divers should never be sent alone, being 3 the recommended number of divers for underwater missions [32]. Health is also another concern, from short-time conditions such as decompression sickness (gases come out of solution in body fluid, forming bubbles that lodge

themselves mainly in joints, causing pain; caused by uncontrolled ascent) and nitrogen narcosis (decrement in mental functions resulting from increased concentration of nitrogen in body tissues) to more permanent cognitive issues and hearing loss. When diving in contaminated water there should also be increased precautions such as a decontamination process before removing the diving suit due to the diver's health. Divers also have to pay attention to their dive time at certain depths, having to take breaks from time to time, and being each subsequent dive of shorter duration.

Two categories of unmanned underwater robotic systems can be identified: Autonomous Underwater Vehicles (AUV) and Remotely Operated (Underwater) vehicles (ROV). Although ROVs are unmanned, they still rely on an human operator controlling the vehicle, usually from a support boat on the sea surface. Being dependent on a communication cable, its range is limited by the cable length, while also having some constraints regarding maneuverability and accessibility to more remote locations, as the cable can get entangled in debris. On the other hand, due to being untethered, AUVs have a higher degree of mobility, being able to navigate through more complex locations and has the advantage of not needing an human operator controlling its movement, as such, can perform tasks that would otherwise be too time-consuming or demanding for humans [33].

Facilities in or associated with the sea (harbors, oil rigs) have to be inspected with some regularity, be it to assess the damage a boat collision with a pier or wall might have caused or even a routine assessment of corrosion [34]. An autonomous vehicle would be have to perform this task routinely without the human effort these inspection tasks entail. An AUV is presented by Jacobi in 2015[34] nicknamed "SeaCat" for this purpose. For general navigation and positioning it uses a laser gyroscope combined with a Doppler Velocity Log, a navigation method that uses the reflections of an acoustic pulse directed at the bottom of the sea to estimate the vessel's current position in relation to a previous one (dead-reckoning)[10]. It also features a sensor for inspection on its bow (front; aft - rear) with a pan-tilt unit, consisting of a video imaging system with laser illumination [11] and a multi beam echo sounder (MBES)[12].

AUVs have great potential for **Search and Rescue** operations, for example, in the recovery of black-boxes from crashed planes. An autonomous vehicle would have a longer operating time, since it isn't constrained by the the negative effects diving has on humans divers, being able to sweep a wider area. It wouldn't also need the same type of support boat and crew, the AUV could be launched from any general purpose aquatic vehicle.

There are some works in the literature dedicated to this topic. For example, Bonin-Font et al. (2015)[35] resort to a color-based algorithm to detect and track (using Histogram Back-Projection [36]) a (bright orange) black-box in an underwater environment, claiming that objects which have a dominant representative color significantly different from the background can be easily detected, both during the day and night (resorting to external light sources). They also test a feature-based

[10] https://www.nortekgroup.com/insight/nortek-wiki/new-to-subsea-navigation
[11] https://www.stemmer-imaging.com/en-se/knowledge-base/laser-illumination/
[12] https://www3.mbari.org/data/mbsystem/sonarfunction/SeaBeamMultibeamTheoryOperation.pdf

approach at object detection using the ORB feature detector [14], which has a lower detection score, since the feature detector also detects some keypoints in the environment surrounding the objects, due to the texture of the algae.

Fatan et al. (2006)[37] present a new approach for detection of an underwater cable, giving an AUV the capability of automatically tracking a cable. First, image edges were extracted using Canny Edge Detection [38] on a grayscale image. Then, texture feature extraction was applied to generate feature vectors to train two classifiers, a Multilayer Perceptron (Neural Network) and an SVM, in order to classify the resulting edges as "cable" and "not cable", to filter the edges so as to facilitate the next step, where the authors apply a Hough (Line) Transform [39] to detect the two lines belonging to the cable. From this work it can be retained that, in spite of the SVM being an efficient classification method, it doesn't always outperform Neural Networks. The authors only tested this method in a simulated environment, using images of the cable in a relatively regular background to train the classifier. It is expected this approach wouldn't hold up if tested in a real situation/environment.

Lee et al. (2012) [40] evaluate vision-based navigation techniques in underwater environments, resorting to real-time detection and tracking of underwater artificial landmarks, designed so as to have little view point change while still allowing the study of underwater color degradation. two approaches are tested: a feature based approach using SIFT, which, as expected, was unsuccessful, since the objects chosen were simple and featureless. The best approach was template-based, using normalized cross-correlation template matching (NCC)[41]. Another purpose of the experiment was to compare matching with templates obtained in-water against templates obtained in-air. The authors concluded that better results can be achieved with a template generated from images obtained in the actual underwater environment. However, in a practical situation in-air templates are more easily obtainable. This approach would have questionable effectiveness in a real tracking situation since NCC-based template matching isn't robust to scale, rotation and large perspective changes [41]. Robustness to changes on these factors could be improved by matching with different templates taken in different conditions. That would increase the detection and tracking complexity, and possibly not allowing the system to be used in real-time.

Kumar et al. (2018)[42] develop the concept of a AUV for Vision Based Tracking of marine species. A feature-based approach is used, resorting to ORB, which is found to be more appropriate for real-time applications since frame processing takes 19 times less than with a SIFT feature detector (0.22 vs. 4.2 seconds), despite accuracy being lower (63% vs. 76 %). No comparison between multiple animal species is made nor the algorithms' performance tested outside a controlled, relatively clutter-free environment, where the features of the target are clearly identifiable.

Spampinato et al. (2016)[43] tackle the difficult problem of fine-grained classification of fish species in low-quality visual data captured in real-life settings. While research of the last decades on object recognition has been mainly focused in distinguishing very dissimilar classes, there are many applications which require specialized domain knowledge that can benefit from automated fine-grained classification, where classes have a much higher degree of similarity betweenn them, such as bio-diversity studies both in the sea and on the ground. Their work contributes with two

fine-grained classification approaches, for still images and video, the curation and release of a dataset[13] for the purpose of those tasks.

2.5 Critical Analysis

Robotic underwater applications found in the literature[35] which use visual classification don't have it as the main focus, dedicating a single section to the visual classification problem in a research paper about an entire robotic platform, for example, reducing the depth at which the authors could explore the problem.

A wide variety of oceanographic research works that use visual classification techniques focus on the detection and classification of fauna and flora, which gives a good indication of what methods work in the underwater environment.

Very limited research has been found on quantifying the impact of the underwater environment on image classification methods. An example found where the authors approach that topic is the work by Lee et al. (2012)[40] but only as a side comparison.

Despite visual image classification being a vast field, with an ever-evolving state-of-the-art, research on the influence of the aquatic domain on air-trained object classification techniques remains unexplored, no research was found where the authors compared the same image classification techniques to the same objects both in an aerial and aquatic domain.

[13] Available on: http://perceive.dieei.unict.it/datasets.php

Chapter 3

Dataset & Methodology

In this section several datasets for object recognition benchmarking are presented, although they aren't suitable for the environment comparison objective of this thesis, hence the need to collect a new dataset.

The new dataset is then presented, starting with its construction workflow, where it is analyzed in detail. The choice of objects is justified, followed by the image collection setup conditions and camera specifications, concluding with the data augmentation techniques and post-processing applied to the collected images.

3.1 Useful datasets in the literature

Parallel to the progress in the visual perception niche of the machine learning field there has also been an increase of publicly available datasets, aiming to benchmark algorithms from pedestrian [44] and road sign [45] detection to human body pose estimation [46]. This thesis falls in the object recognition category, thus some datasets for that purpose are analyzed in this section.

3.1.1 ImageNet

A on-going database of 14,197,122 labelled images[1], following the semantic hierarchy of WordNet [47] that can be useful for a variety of machine learning tasks such as object recognition, fine-grained classification and automatic object clustering [48]. Composed of several *subtrees*, each having higher degrees of specificity, for example, in Fig. 3.1 are represented the *"mammal"* and *"vehicle"* subtrees, which have as leafs (the end of the subtree) a specific type of mammal and vehicle.

Due to its enormous variety of types of objects and number of images, ImageNet is used to pre-train Convolutional Neural Networks for several other visual-based applications which are then fine-tuned for the task in specific [29], from colorectal cancer nymph node classification [49] to plankton classification [50].

[1] http://image-net.org/

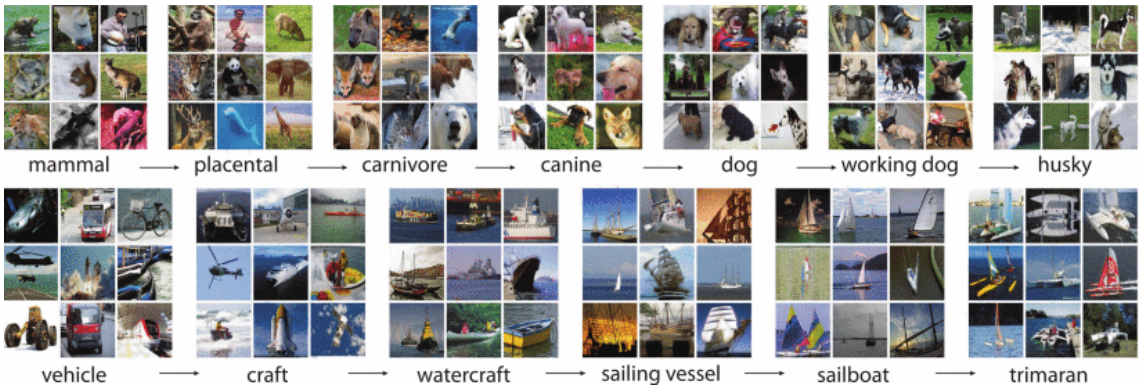


Figure 3.1: ImageNet semantic hierarchy example with images from the database

3.1.2 Microsoft COCO

Microsoft COCO: Common Objects in Context [3] is another large-scale (over 330 thousand labelled images) that, in contrast with ImageNet has fewer (91) categories but more instances of those classes. In total, 2.5 million instances are labelled among those images. The dataset addresses three problems in scene understanding research: detection of non-canonical views (most datasets present the objects in clear, non-obstructed view; this dataset aims to have some images with partially occluded objects in a scene where they might be found), contextual reasoning between objects and precise 2D localization, being the labels a more or less precise segmentation mask.

(a) Iconic views *(b) Non-iconic views*

Figure 3.2: Microsoft COCO image examples[3]

3.1.3 BENTHOZ-2015

This dataset [51] was collected as part of the Integrated Marine Observing System (IMOS), a marine research program of the Australia Government and consist of thousands of expert-labelled images of the seafloor around Australia's coast. This dataset is of interest to researchers studying

benthic habitats and organisms present there, while also being useful for the creation of 3D maps and development or test of Visual SLAM algorithm since this data is georeferenced (each images has associated the GPS coordinates where it was taken) as well as having other information from sensors, such as depth, altitude, temperature and salinity. The dataset is freely available in the Internet [2], and associated with it is an annotation tool useful for this kind of data, *Squidle*, *a tool for managing, exploring and annotating images, video and large-scale mosaics*.

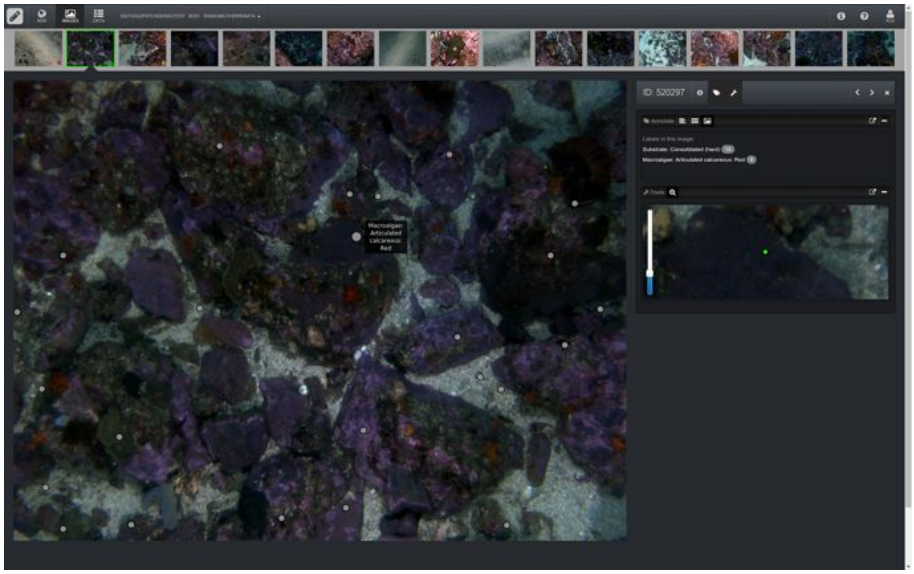


Figure 3.3: Example of annotated image from the BENTHOZ-2015 dataset

3.1.4 Marine Underwater Environment Database (MUED)

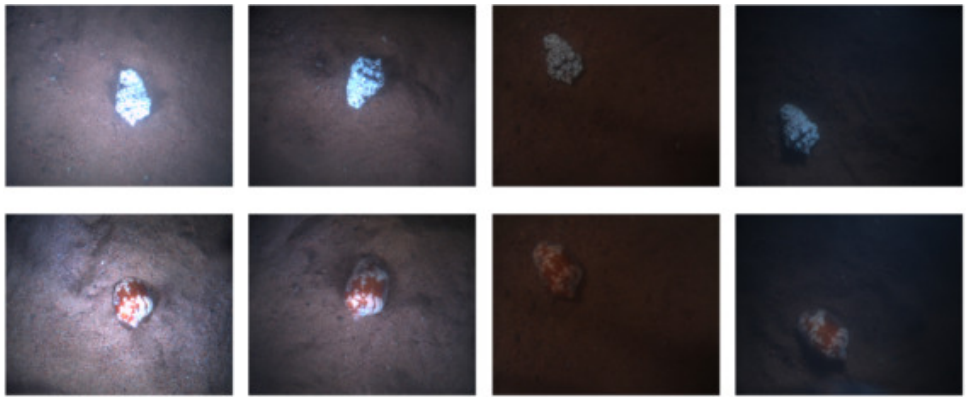


Figure 3.4: Object from the MUED dataset in different lighting conditions

Jian et al. (2019) [52] present a dataset with 8600 images of 430 different underwater objects, each image with one or more objects in a complex background, with variations in object pose, turbidity and illumination. The authors used this dataset exclusively to benchmark salient-object

[2] http://squidle.org/

detection algorithms concluding that most state-of-the-art algorithms do not adapt well to complex underwater environment. Unfortunately, this dataset cannot be used to compare performance in an object classification task between two environments, since it doesn't have image of the objects out of water.

3.2 The proposed dataset: HEIMDACA

The purpose of this thesis is not to perform image classification at a large scale, since there is already extensive work on the area, but to evaluate what is the difference in performance between the same image classification method applied both underwater and above water, using in both environments the same objects or classes. No dataset was found for this purpose, thus the need of collecting images of several objects in two different conditions:

The custom dataset, HEIMDACA [3] consists of pictures of 8 objects (classes) in two different environments: underwater (***aquatic***) and out of water (***aerial***).

The classes that compose the dataset are:

- *anchor* - small-sized anchor;
- *chain* - metal chain;
- *box* - square metal box;
- *float* - floater used in pool lane separators;
- *lead* - circular lead disk;
- *mark* - artificial marker presented by Figueiredo et al.; (2016)[53] for navigation and localization of UAVs;
- *weight* - pill-shaped heavy object used as ballast in small water vehicles;
- *ballast* - round epoxy object with a ring on top, also used as ballast.

[3] **H**ybrid **E**nvironment **Im**age **Da**taset for **C**lassification **A**pplications

(a) Anchor *(b) Box* *(c) Chain*

(d) Float *(e) Lead* *(f) Mark*

(g) Ballast *(h) Weight*

*Figure 3.5: Examples of objects collected in the **aerial** domain*

The underwater part of this dataset was collected in a tank and the aerial images in a laboratory of CRAS, both located in the Department of Electrical Engineering and Computers at Faculdade de Engenharia da Universidade do Porto. All images were taken from a top-down perspective at a fixed camera-object distance per set, resorting to an L-shaped structure (see Fig.3.7 and Table 3.1).

Two sets of images were taken on each environment (Fig. 3.6)

- ***Aerial***: *With Background* (Fig. 3.5), images taken with a green piece of cardboard, which facilitates the creation of segmentation masks [54] and *Without background*.
- ***Aquatic***: *Clear background*, images were captured in a section of the tank with its bottom relatively undamaged. *Rough Background*, images captured in a more textured part of the tank.

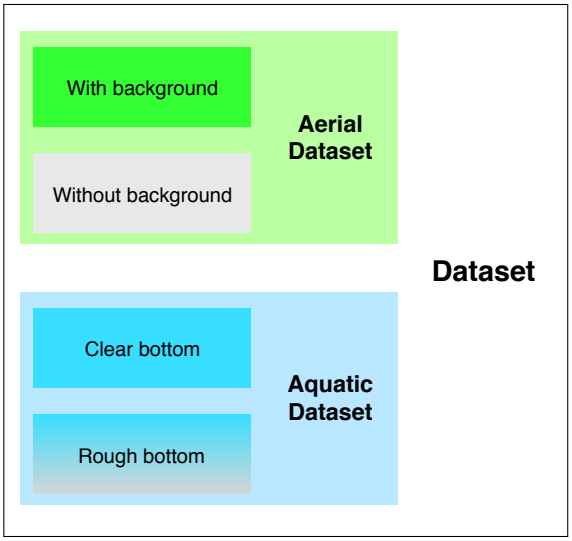


Figure 3.6: Dataset division

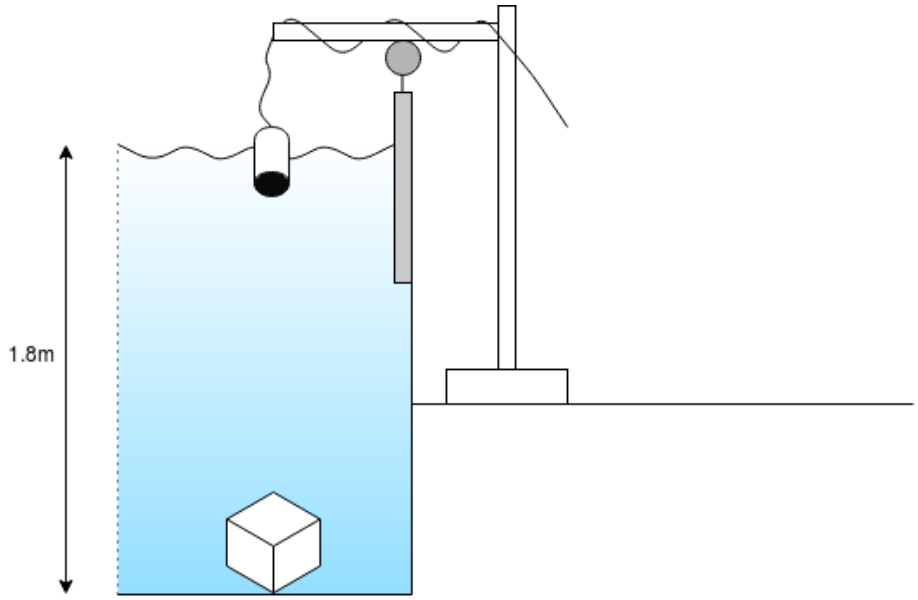


Figure 3.7: Underwater image capture setup

A MAKO G-125C camera from Allied Vision [4] was used to collect both aerial and aquatic datasets, the only difference being a plastic enclosure to make the camera waterproof, for safe use in the underwater environment. The enclosure had a flat-pane window, which might have introduced distortion due to the water-glass interface [55] but after inspection of the captured frames, no significant distortion degradation was found.

Images were captured at a 1292 by 964 pixels resolution, with RGB8Packed pixel format. Exposure, gain and white balance settings were auto-set using the camera's inbuilt algorithm while running the camera in the chosen environment, with no object in its field-of-view. After the desired

[4] https://www.alliedvision.com/en/products/cameras/detail/Mako 20G/G-125.html

Table 3.1: Camera and environment characteristics

	Aerial		**Aquatic**	
	W/Background	***W/o Background***	***Clear***	***Rough***
Distance to object	0.5 m	1.0 m	0.9 m	1.8m
Exposure	26.5 ms	18.0 ms	28.7 ms	25.5 ms
Gain	13 dB	9 dB	0 dB	0 dB
White Balance	Red: 1.60 Blue: 2.70	Red: 1.60 Blue: 2.70	Red: 3.0 Blue: 3.0	Red: 3.0 Blue: 3.0

values stabilized, they were kept constant for each set of conditions and are summarized in Table 3.1.

To guarantee camera settings were kept constant between frames, images were captured with the help of the *avt-vimba-camera* [5] plugin for the Robot Operating System (ROS). Cameras were calibrated using a 9 by 11 checkerboard pattern and Zhang's algorithm[56].

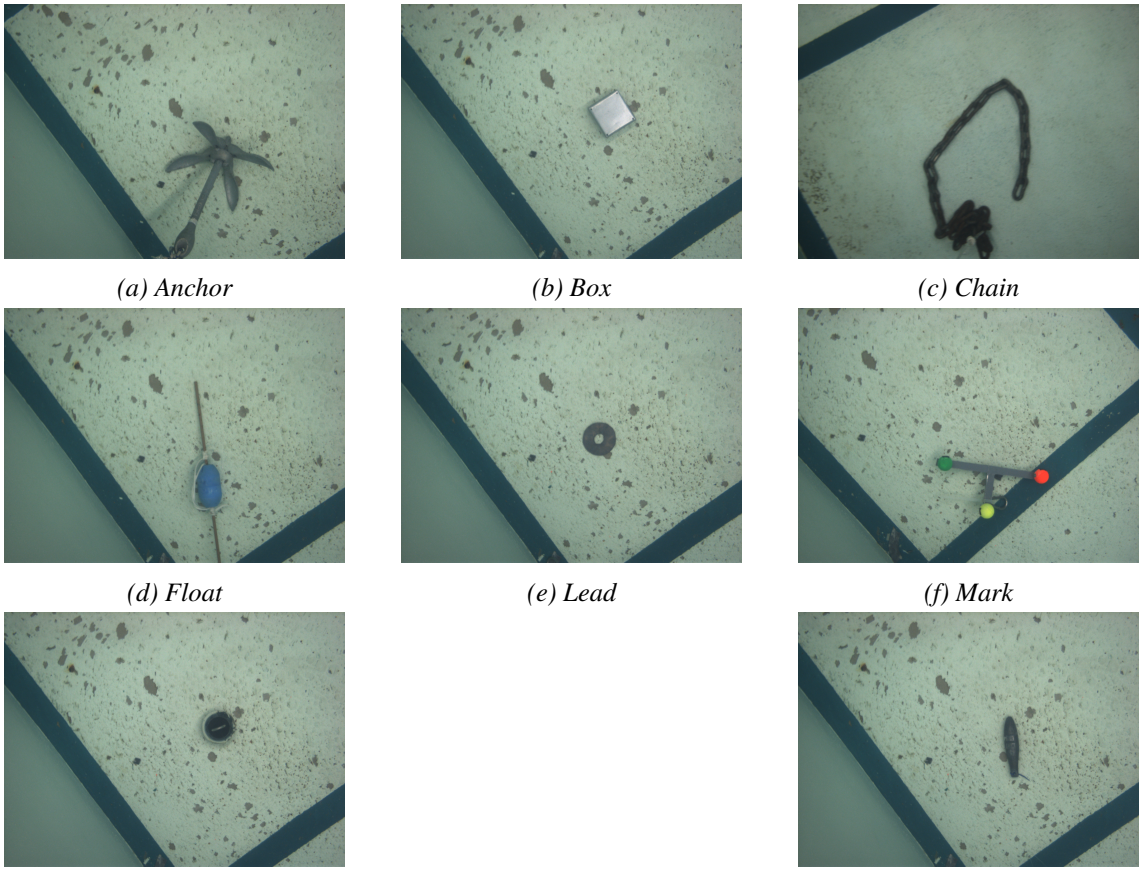
For the ***aerial*** and ***aquatic*** images, after the capture of several images, 10 frames were selected per object. The images chosen tried to show the object from multiple perspectives. Then, black and white segmentation masks were generated using ***Fiji*** [57], a *software* that allows for quick prototyping and test of image processing tools. The masks for the ***aerial*** images were created resorting to color segmentation due to their uniform background. First, image were undersegmented (Fig. 3.9b) and then morphological dilation was applied (Fig. 3.9c), generating a mask than encompassed the entire object. Masks for the ***aquatic*** images were created manually, using a "Polygonal Select" tool.

Data augmentation was performed by applying random combination of horizontal and vertical mirroring, scale variations (from 50% to 100% of the original image size) and rotations, 450 images were created per object from the 10 chosen frames. This technique was used to create diversity in the training sets while saving time in the data acquisition process as well as avoid overfitting and lack of generalization of the models [58]. Perspective transformation was applied to ***Aquatic*** images to simulate changes in the camera viewpoint and test the models' robustness to that variation (Table 3.2).

Table 3.2: Dataset summary

	Aerial		**Aquatic**	
	W/Background	***W/o Background***	***Clear***	***Rough***
# of images per class	400	200	400	200
Bounding Box	✓	✓	✓	✓
Segmentation Mask	✓	✓	✓	✓
Data Augmentation Type	Vert./Hor. Mirror Rotation Scale (0.5-1)	None	Vert./Hor. Mirror Perspective	Vert./ Hor. Mirror Perspective

[5] https://wiki.ros.org/avt_vimba_camera



(a) Anchor *(b) Box* *(c) Chain*

(d) Float *(e) Lead* *(f) Mark*

(g) Ballast *(h) Weight*

*Figure 3.8: Examples of objects collected in the **aquatic** domain, **Rough Background***

(a) Original Image

(b) Undersegmented Image *(c) Image after morphological dilation*

*Figure 3.9: **Aerial With Background** mask creation*

3.3 Conclusion

This chapter presented a custom dataset purposely made for this thesis. Composed of 8 objects, in two different environments, its aim isn't large scale classification, but rather to help evaluate and measure the effect of the underwater environment on the performance of image classification techniques. Data augmentation was performed in a subset of the images to introduce more variety in the data.

The following chapters will present the evaluation of machine learning techniques applied for image classification and some remarks about this dataset will be made in the end of this document.

Chapter 4

Conventional Machine Learning

This chapter aims to evaluate how conventional machine learning methods adapt from the aerial to the aquatic domain on an image classification task.

Neural Networks and Support Vector Machines will be used, combining feature extraction with a Bag-of-Words approach for generation of the input data. First, the classifiers will be trained and evaluated on the aerial domain, establishing a baseline for further comparison and discussion. Then, those air-trained classifiers will be evaluated on the aquatic dataset, measuring the loss those methods suffer from changing the domain. Finally, the air-trained models will be incrementally trained with images acquired in the aquatic domain, followed by an evaluation on that same domain.

All tests will be performed with an increasing number of classes. An initial prediction is that, due to the intermediate Bag-of-Visual-Words step, performance will decrease with the increase in the number of objects. The BoVW approach resorts to K-Means, an unsupervised learning algorithm, as the vocabulary forming method, which might cluster different "visual words" from different objects in the same cluster, losing descriptiveness in the process.

4.1 Evaluation Metrics

Before a comparison can be made, it is first necessary to define the metrics being evaluated. Fawcett (2006)[59] present an introduction to a set of metrics that has seen increased use in machine learning and data mining research: ROC[1] graphs, which represent visually some metrics extracted from the Confusion Matrix.

The Confusion Matrix (Fig. 4.1) summarizes the four possible outcomes of a binary classifier: a classifier can predict an instance as positive, and if its true value is positive it counts as *True Positive (TP)* or, if negative a *False Positive (FP)*. If the classifier predict the instance as negative and its true value is negative, it's a *True Negative (TN)* or if it's positive, *False Negative (FN)*. From this matrix some commonly used metrics can be derived, such as *Precision* or *specificity*, but the ones relevant for this thesis are the *True Positive Rate (TPR)*, also known as *Recall*, and the

[1] Receiver Operating Characteristics

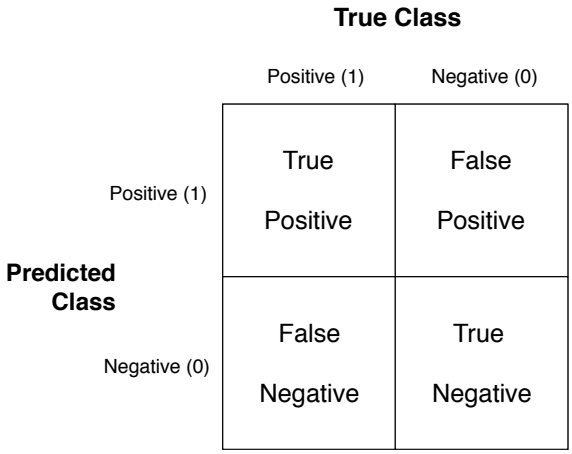


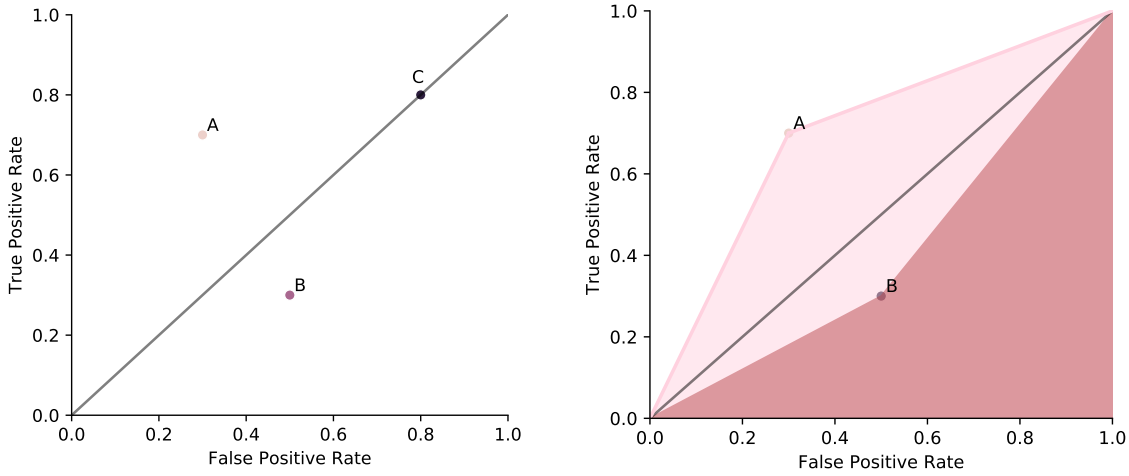
Figure 4.1: Confusion Matrix

False Positive Rate (FPR), or *Miss Rate*, Equations 4.1 and 4.2, respectively. A perfect classifier would have 100% *TPR* and 0% *FPR*.

$$TPR = \frac{TP}{TP+FN} \tag{4.1}$$

$$FPR = \frac{FP}{FP+TN} \tag{4.2}$$

A ROC graph, Fig. 4.2a, is a two-dimensional plot where the *TPR* is plotted on the Y axis and the *FPR* on the X axis. The classifiers used in this thesis will be discrete[59], producing a single point on the ROC graph. The diagonal $y = x$ line represents the behaviour of a classifier with no discrimination capacity, meaning the result of the classifier can be considered random. A classifier in the lower triangle (classifier B) is worse than random guessing and in the upper triangle the opposite (classifier A). The (0,1) point represents perfect classification and (0,0) means the classifier only predicts occurrences as negative.



(a) ROC graph

(b) AUC of classifier A and B

Figure 4.2: ROC graph and curve formed

Two different classifiers can be compared using a single value derived from ROC graph, the **A**rea **u**nder the ROC **c**urve (AUC) which, since the graph is a unit square, will always be between 0 and 1. On a discrete classifier, the AUC is the area of the trapezoid of vertex the points (0,0), (1,1), (1,0) and (*FPR,TPR*). Fig. 4.2b shows examples of two *AUC*, classifier A has $AUC = 0.7$ while B has $AUC = 0.4$. A random classifier would have $AUC = 0.5$.

Multiclass classifiers can be composed of multiple binary classifiers. In this case, the AUC is also a means of comparison between multiclass classifiers. Assuming a multiclass classifier composed of N binary classifiers, each of the (sub-)classifiers has its own AUC and if the number of occurrences in each class is the same, all the AUC can be averaged, resulting in the total AUC of the multiclass classifier.

$$AUC_{total} = \sum_{i=1}^{N} AUC_i \cdot \frac{1}{N} \tag{4.3}$$

4.2 Procedure

Figure 4.3 shows the intermediate steps taken to validate the image classification models. Figure 4.4 details the sub-stages in the training and validation operations. Those stages will be analyzed, mentioning the algorithms used at each stage. The example given will be that of the air-trained models, the combined training models followed the same principles.

[2]Using the SIFT implementation of OpenCV 4.1

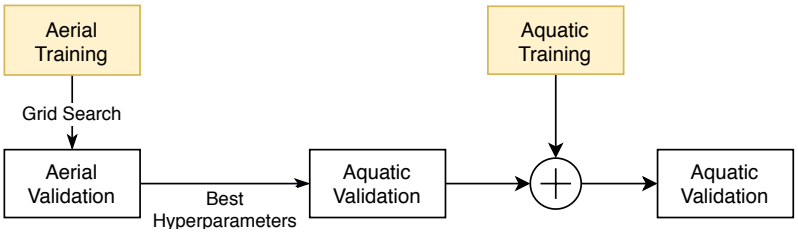


Figure 4.3: Training and validation procedure

4.2.1 Feature Extraction and Bag-of-Words

Before extracting features, images were converted to grayscale, as (regular) SIFT [12] takes a single image channel as input. Then, keypoint detection and description was performed on each of the aerial images[2] with the SIFT parameters suggested by Lowe in the SIFT paper [12], as the measure of the influence of those parameters on classification accuracy was decided to be out of the scope of this thesis. The choice of the feature extractor was made taking into account the influence the underwater environment can have on the imaging process (See Section 2.1), thus the need of a keypoint descriptor that is robust and invariant to those changes. Despite SIFT being rather computationally heavy in comparison to other feature extractors, its robustness should bring some advantages regarding classification accuracy.

After extracting features from the aerial images for all classes, the Bag-of-Words vocabulary was generated, using the method described by Csurka et al. (2004) [16]. Considering all features, the K-Means clustering algorithms is used to generate a vocabulary, aggregating similar features in clusters and forming "visual words". Three different vocabularies were generated, with ***100***, ***500*** and ***1000*** visual words. As mentioned before, and as claimed by the authors, this step reduces the dimensionality of the data at the input of the classifier, resulting in a fixed-sized input vector and a lesser computational complexity of the classification task, while at the same time bringing some robustness to keypoints not favorable to the task (e.g, background clutter), which might prove to be beneficial on the underwater environment, due to marine snow, for example.

4.2.2 Training strategy

Exclusively One-Vs-All (OVA) classifiers will be presented on this chapter, being this choice backed up by the literature[60] and preliminary testing. OVA classifiers are composed of multiple

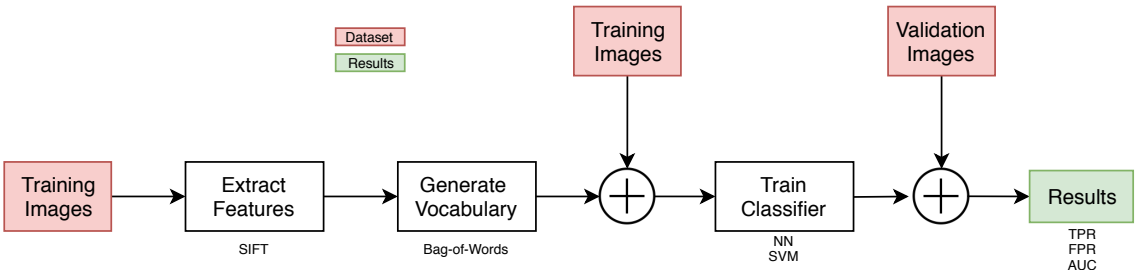


Figure 4.4: Sub-stages of the training and validation process

independent binary classifiers that identify an occurrence (image) as belonging to a *class* (positive) or *not class*. For example, in a task with 8 classes, there will be 8 binary classifiers. The *TPR* of the entire classifier can be calculated by averaging the *TPR* of the different binary (sub-)classifiers Rifkin and Klautau present [60] a fair comparison between OVA and other more complex multi-class classifiers that are claimed to outperform OVA. Disagreeing with a large body of published work on multiclass classification, they conclude when the binary classifiers are properly tuned, OVA performance matches that of the other classifiers. However, one great disadvantage of OVA is that, assuming a classification task with N classes, each image is tested by N binary classifiers.

Aerial Training was performed in two stages:

1. Following a wide range of parameters (to be presented further ahead), each classifiers was first trained with 400 images from the ***With Background*** part of the dataset, followed by an evaluation in 50 images from the ***Without Background*** part. The model with the highest AUC_{total} (Equation 4.3) was then taken to stage 2.

2. A tighter range of parameters is searched around the best result of the previous stage, also training the classifiers with 400 images from the ***With Background***, but now evaluating 200 images per class, from the ***Without Background*** part. The best classifier is chosen as the set of parameters with the highest AUC_{total}.

4.2.3 Machine Learning Classifiers

Neural Network On neural networks, the choice of the number of hidden layers and their size is usually done empirically, not following any rule, being problem specific and depending to some extent on the training data amount and quality [61]. However, some guidelines should be followed; the number of hidden layers and neurons should be sufficiently low to ensure generalization, as a high number can encourage over-fitting of the network to the training data. Three depths were tested: 1, 2 and 3 hidden layers, with two activation functions, ReLu and Sigmoid. Two different optimization algorithms were tested, ADAM and SGD, with multiple values for their learning rate.

SVM Similarly to neural networks, the SVM parameters are also chosen empirically, taking into account their mathematical meaning. Intuitively[26], C can be thought as the complexity of the surface that separates the positive from the negative decisions. Low C makes the decision surface smooth, possibly leading to misclassifications while a high C might result in a perfect classification of the training data, but lead to lack of generalization due to over-fitting. γ determines the influence of each training example on the decision boundary. A high γ can lead to "islands" on the decision boundary while a too low γ results in a not complex enough surface. Table A.1 summarizes the parameters and the values initially used for each classifier.

All classifier were tested with an increasing number of classes. The objects used in each iteration can be seen on Table 4.1, their order chosen at random.

Table 4.1: Objects used at each iteration

# objects	***Objects***
4	anchor, chain, lead, box
5	anchor, chain, lead, box, weight
6	anchor, chain, lead, box, weight, float
7	anchor, chain, lead, box, weight, float, mark
8	anchor, chain, lead, box, weight, float, mark, ballast

4.3 Results

Air-trained model

Aerial Validation

Table 4.2a shows the best results obtained by the Neural Network classifier after the second training stage. As expected, the *TPR* of the classifier decreases inversely to the number of objects, while the *FPR* has a much smaller variation, meaning the classifier is identifying more objects as *not class* as the number of objects increases. The SVM (Table 4.2b), however, despite having a similar behaviour with respect to the number of objects, is only better than the NN with 7 objects, having a 2.1% lower average *TPR*.

Aquatic Validation

It can be seen from Table 4.3a that the best air-trained NN suffers a great performance hit when applied in the underwater environment. On average, *TPR* decreased 40.86%, 10 times the increase in *FPR*, 4.5%, meaning that the network (and its sub-classifier) is more biased towards classifying an image negatively. By changing the environment, the imaging conditions will also vary, altering the SIFT description of the keypoints which might be the cause for the great drop in *TPR*, in other words, when the air-trained classifier tries to classify an image of the class *anchor*, if the keypoint description changes (and consequently the BOW vocabulary), the classifier won't know that an *anchor* can be described that way.

*Table 4.2: Best results of the second training stage, validation on the **Aerial** dataset*

(a) NN

# objects	***TPR***	***FPR***	***AUC_{total}***
4	93.8	4.9	0.94
5	92.3	5.7	0.93
6	82.5	2.7	0.90
7	68.2	4.1	0.82
8	63.1	3.0	0.80

(b) SVM

# objects	***TPR***	***FPR***	***AUC_{total}***
4	90.6	3.5	0.94
5	87.2	3.6	0.92
6	76.8	6.5	0.85
7	76.4	4.4	0.86
8	58.5	2.6	0.78

*Table 4.3: Performance on the **aquatic Rough Background** dataset of the best air-trained model obtained after the second stage.*

(a) NN

# objects	***TPR***	***FPR***	***AUC_{total}***
4	66.4	6.6	0.80
5	47.5	11.1	0.68
6	35.6	9.5	0.63
7	27.9	6.0	0.61
8	14.3	5.2	0.55

(b) SVM

# objects	***TPR***	***FPR***	***AUC_{total}***
4	66.6	9.7	0.79
5	51.6	12.3	0.70
6	31.7	3.6	0.64
7	24.6	8.6	0.58
8	16.7	2.1	0.57

Air-trained model incremental training

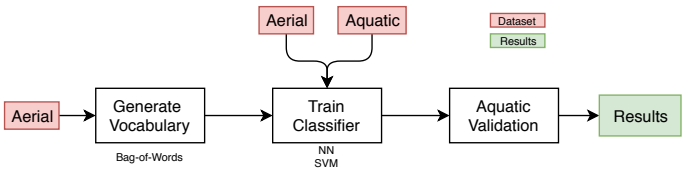


Figure 4.5: Incremental training approach 1: Reusing vocabulary

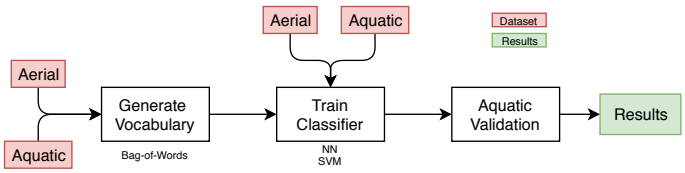


Figure 4.6: Incremental training approach 2: Retrain vocabulary

The best model obtained in the aerial evaluation step will be incrementally trained with aquatic images, in order to evaluate how the air-trained model improves with the addition of those images. Two approaches will be compared:

1. Reuse the vocabulary of the two previous tests, which only contains "visual words" extracted from aerial images, ***aerial vocabulary***. The model is then trained using both the aerial and aquatic datasets, using the hyperparameters of the best model obtained in the grid-search. Evaluation is then performed on the aquatic dataset (Fig. 4.5).

2. Create a new vocabulary, with "visual words" generated from the combination of the aerial and aquatic datasets, ***combined vocabulary***. The model is then trained and evaluated similarly to the first approach (Fig. 4.6).

Table 4.5: Incremental training performance, ***combined vocabulary***

(a) NN

# objects	***TPR***	***FPR***	***AUC_{total}***
4	44.7	4.0	0.70
5	58.0	8.9	0.75
6	27.2	6.5	0.70
7	45.5	1.4	0.72
8	40.3	7.9	0.66

(b) SVM

# objects	***TPR***	***FPR***	***AUC_{total}***
4	69.1	7.0	0.81
5	56.3	5.2	0.76
6	36.4	2.6	0.67
7	42.6	7.3	0.68
8	26.6	1.0	0.63

Table 4.4: Performance in the ***aquatic Rough Bottom****, training with both aerial and aquatic images,* ***aerial vocabulary***

(a) NN

# objects	***TPR***	***FPR***	***AUC_{total}***
4	74.1	4.5	0.85
5	52.6	5.1	0.74
6	39.3	5.7	0.68
7	39.8	2.8	0.69
8	33.3	3.1	0.65

(b) SVM

# objects	***TPR***	***FPR***	***AUC_{total}***
4	73.5	6.6	0.83
5	56.5	7.3	0.75
6	40.8	5.1	0.68
7	45.5	6.8	0.69
8	29.2	2.6	0.63

On the first approach (Table 4.4), both classifiers show improvement from being trained with the aquatic images, with *TPR* almost doubling when evaluating 8 objects, with the NN surpassing the SVM on that specific case. Difference in AUC_{total} between the classifiers is minimal. However, an anomaly occurs on the test with 6 objects, as it was predicted that *TPR* would decrease with the increase in the number of objects, but *TPR* is higher in the test with 7 than with 6 objects.

The second approach presents an even more erratic behaviour, especially with the 4 object test of the NN, which demonstrates an unusually low *TPR*, comparing with the other tests performed. Since model parameters were kept the same in both the aerial evaluation, ***aerial vocabulary*** and ***combined vocabulary*** tests, it can be speculated that the cause of this drop is the use of both the aerial and aquatic images in the vocabulary creation process, the unsupervised K-Means clustering created a vocabulary with visual words unable to uniquely characterize the different classes, thus the bad performance.

Again, similarly to the first approach, *TPR* from both classifiers has a sudden drop on the test with 6 objects. This non-monotonic behavior of the *TPR* regarding the number of objects in both the tests with the ***aerial*** and ***combined vocabularies*** might be indicative that *float*, the class added from the 5 object to the 6 object test, might not have the same classification potential as the other classes, i.e., *float* cannot be properly classified with the approach used in this part of the thesis: feature extraction combined with Bag-of-Words. To get more insight on this hypothesis, it is necessary to look at the individual performance of each binary (sub-)classifier.

From Fig. 4.8, which shows the ROC graph for each individual binary classifier used when evaluating a NN with 5 and 6 objects, it can be deduced that the hypothesis of the *float* class being

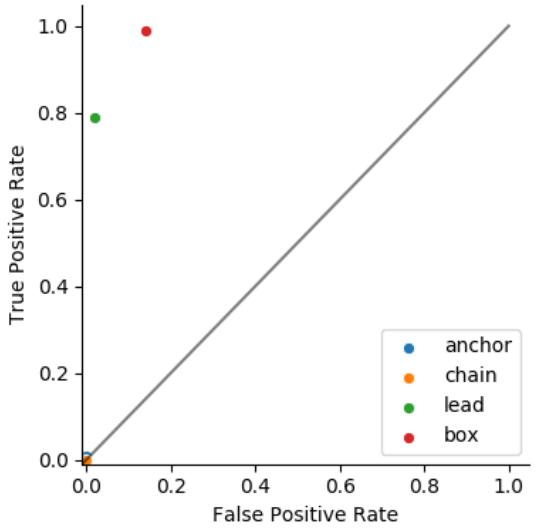


*Figure 4.7: NN **combined vocabulary**, individual object performance, 4 objects*

the cause of the drop in performance is false. In fact, it's only second to the *chain* class, with over 80% *TPR* and under 40% *FPR*.

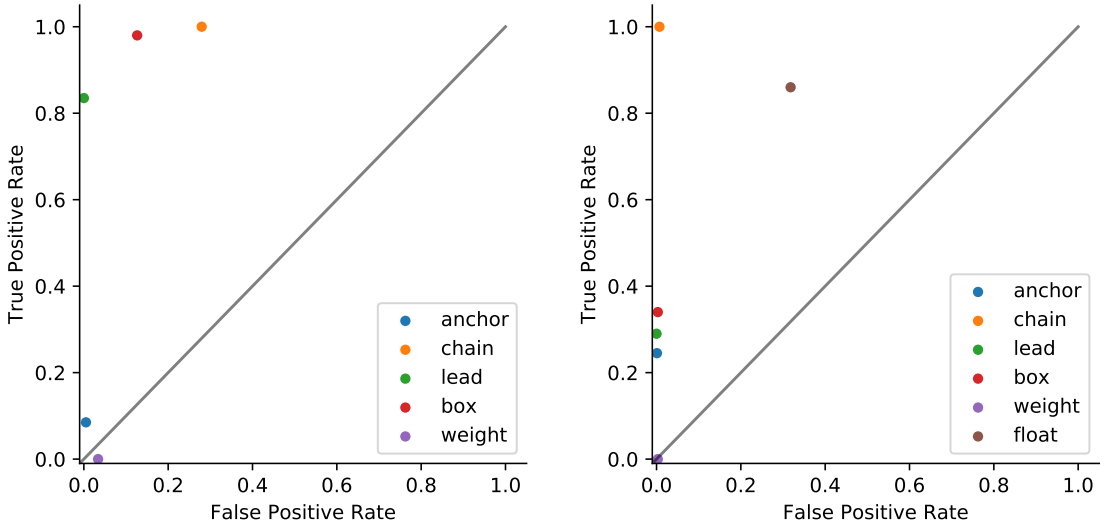
Figs. 4.8 and 4.9 also show a very neutral behaviour of the *weight* binary classifier, the majority of its predictions being negative, as belonging to the *not weight* class. Analyzing the ROC graphs of the combined training, aerial vocabulary tests, not inserted in this document, the same behaviour can be observed, which raises a similar hypothesis as posed regarding class *float*, the existence of a weaker class.

The answer to the new hypothesis can be found if a single example against it is encountered. If a model that outperforms the ones seen before can be found, then the problem lies in the model parameters, not in the class. Thus, a smaller variation of the parameter grid on Table A.1 is sweeped with both classifiers, considering only the 8 objects case.

As seen from Figs. 4.10 and 4.11, a binary classifier for the *weight* class that clearly outperforms the ones already known wasn't found. The best *TPR* found was 9.5% on a Neural Network classifier. On the other hand, an SVM was found with $AUC_{\text{total}} = 0.752$, $TPR = 75.0\%$ and $FPR = 24.5\%$. This result is significant as it confirms that the parameters of the best model in the aerial dataset do not correspond to those of the best model in the aquatic dataset, the model has to be adapted to the environment, not only by adding new images but by changing the parameters.

4.4 Conclusion

In the previous chapter two distinct classifiers, Neural Networks and Support Vector Machines, were compared regarding their ability to classify images from different environments. They were first trained exclusively with images taken in a controlled aerial environment and then validated with aerial images taken with different imaging conditions. When testing with 8 objects the NN



5 object *6 objects*

*Figure 4.8: Neural network **combined vocabulary**, 5 and 6 objects performance*

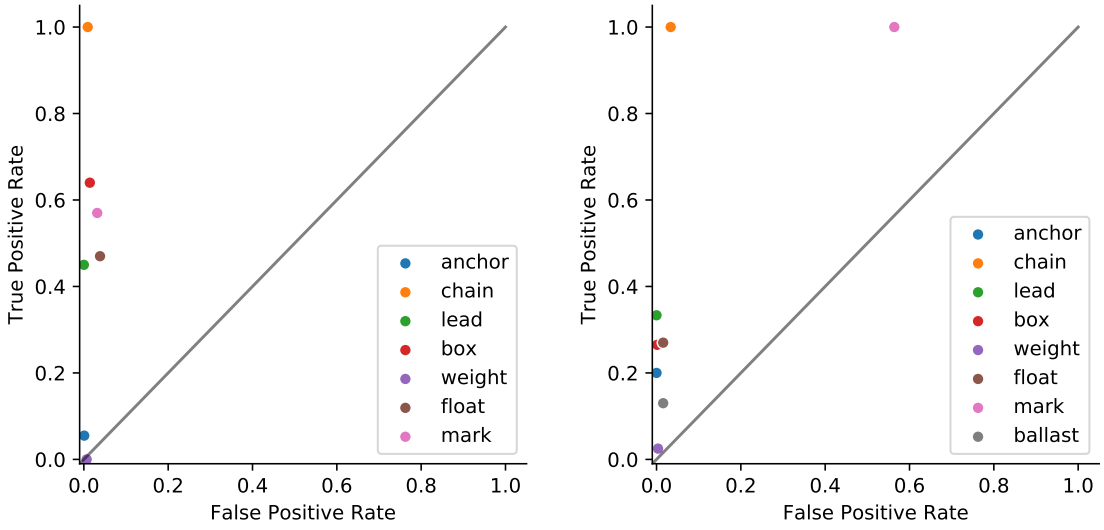
achieved a True Positive Rate of 63.1% and the SVM 58.5%.

Then, the best model of each classifier was used to classify images of the same objects, but taken in a completely different environment, underwater, to verify how the change in environment would affect the classification accuracy. Directly testing the models, without any change in their parameters, resulted in a *TPR* of 14.3% for the NN and 16.7% for the SVM, a drop of 40.86% and 41.86%, respectively. With this result it can be concluded that exclusively air-trained models used for image classification do not translate well to tasks in an underwater environment.

After incrementally training those best models with aquatic images, a maximum increase of 26.0% was achieved with the NN, for a *TPR* of 40.3%. Although the SVM did also improve, the increase in *TPR* wasn't as significant as seen in the NN, with 29.2% *TPR*.

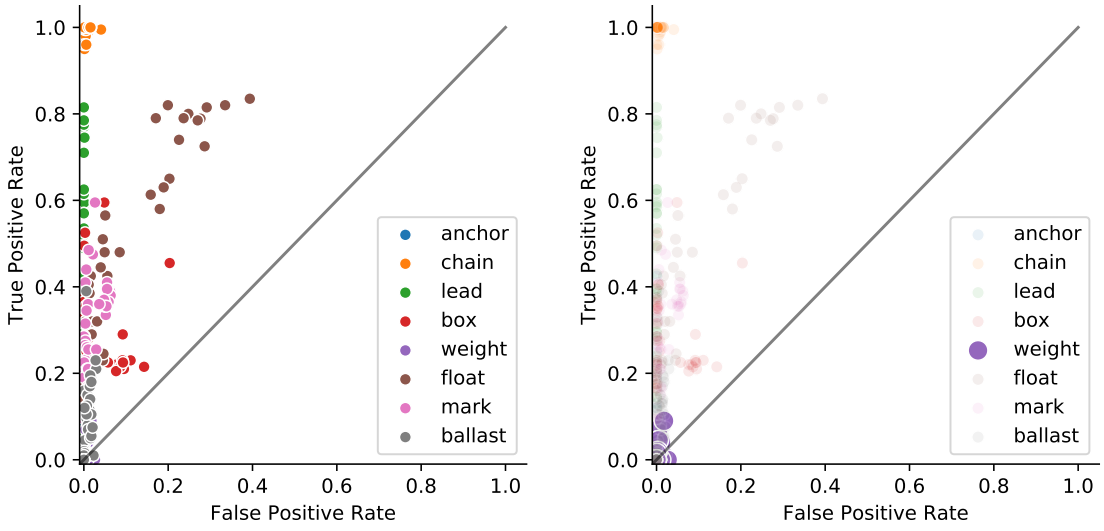
Figure 4.9 shows that certain objects suffered a great drop in *TPR* when being classified in the underwater environment. There is a necessary condition for feature-based methods to work: the feature detector (SIFT) has to be able to detect keypoints, points of interest. Despite SIFT having some robustness to different conditions, it is possible that for some objects the blurring and loss of detail characteristic of light propagation in water lead to the feature detector not being able to find robust enough keypoints to guarantee a good classification, thus the weak results.

Summing up, it can be concluded that the best aerial model obtained with the parameter search technique used does not translate directly to the best model in the aquatic environment. Using these classifiers, Neural Network and SVM, incrementally training the model with images from the aquatic dataset improves the metrics, but does not bring top performance. A new parameter search technique could be employed, but some knowledge of the environment is needed to validate the model.



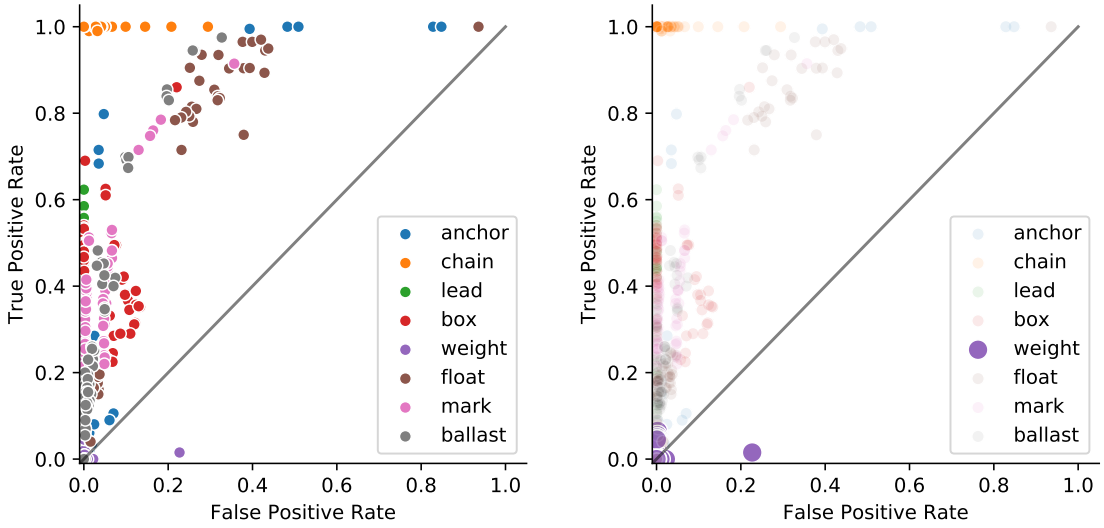
7 objects *8 objects*

*Figure 4.9: Neural network **combined vocabulary**, 7 and 8 objects performance*



Binary classifiers for multiple parameter iterations ***weight** highlighted*

*Figure 4.10: Neural network grid sweep, evaluation with aerial dataset and **combined vocabulary**, 8 objects*



Binary classifiers for multiple parameter iterations ***weight*** *highlighted*

Figure 4.11: SVM grid sweep, evaluation with aerial dataset and ***combined vocabulary****, 8 objects*

Chapter 5

Deep Learning

This section will present an evaluation of the performance of a Convolutional Neural Network from the state-of-the-art in image classification by following a similar approach as adopted on Chapter 4. First, the network will be exclusively trained using images from the aerial domain and evaluated on that same environment, establishing an accuracy baseline for the next experiments. The model will then be validated on the aquatic dataset, measuring the impact on performance of the change in environment. Finally, a second stage of training will be performed, where images taken on the aquatic environment will be used to retrain the network, and their influence on the accuracy will be evaluated. An overview on the tools used can be read on Appendix B.

5.1 Network training procedure

The network architecture chosen was Inception-V3 (Section 2.3.2) due to having the best balance between accuracy on the ImageNet dataset and model complexity of the image classification models analysed by Canziani (2016)[62].

Similarly to other works in the literature [63, 50], a transfer learning approach will be taken, which consists in using a pre-trained model and fine tuning it for the desired purpose, i.e., a model created for a specific classification task serves as a starting point for another task. It is especially useful in applications with limited dataset sizes [63], which is the case in an underwater or medical context. For example, transfer learning can be applied for plankton classification [50], as well as soft tissue sarcoma [63] or cardiovascular tissue classification [64] and both applications use the same initial starting point: the ImageNet dataset, presented in Section 3.1.1, which due to its great image variety and low dataset bias [65] is commonly used to pre-train classification networks, avoiding overfitting of features to a specific shape.

Following this transfer learning approach, the weights of the last 3 fully-connected layers of the fully-trained Inception-V3 network are replaced with uninitialized (random) parameters. The number of nodes in the last layer, the Softmax layer, will be changed to the desired number of classes, instead of the 1000 nodes (classes) for the ImageNet task. Unlike the neural networks used in the Conventional Machine Learning (Chapter 4), this will be a Multi-Class network.

Softmax layer

The softmax layer, usually the last layer of a CNN, consists of a number of nodes equal to the classes to identify. Its output is the probability of a certain image belonging to a certain class, and the sum of the output of all neurons of the softmax layer is 100%. This thesis will only consider Top-1 accuracy, that is, the class identified by the network is the class with the highest probability in the softmax layer.

Object	anchor	chain	lead	box	weight	float	mark	ballast
Probability	0.13	0.2	0.15	0.09	0.11	0.05	0.19	0.08
Output	0	1	0	0	0	0	0	0

Table 5.1: Example of a classification with Top-1 accuracy

Two transfer learning approaches were tested:

1. Only the weights of the 3 replaced layers will be updated, all other layers will remain untrained by the new data, their weights won't be updated in the training process. Thus, the network will only be using features generated with the ImageNet dataset, and the last 3 layers will recombine them to generate a prediction. A set of 2.1 million parameters are trained, roughly one tenth of the total number of parameters that consist the Inception-V3 architecture (23.9 million). This will be named the ***Coarse training*** approach.

2. While training the weights of the 3 replaced layers, the 2 last Inception modules will also be retrained. Training the later representation learning stages will force the network to learn and recombine new features which better reflect the nature of the data used in this work. A total of 13.2 million parameters will be trained. This will be named the ***Fine training*** approach.

For the first set of experiments, training and validation using the aerial dataset, two gradient optimization algorithms, **ADAM**[23] and **Stochastic Gradient Descent (SGD)**[18], will be compared in both approaches. The best optimization algorithm will then be validated against the aquatic dataset, measuring the performance loss caused by the environment. The network with the highest *TPR* on the aerial validation will then be trained with aquatic training data and the performance change in the aquatic environment will be evaluated.

The Keras framework [1] provides weights that were trained on the ImageNet dataset, with 299 per 299 resolution images. The network could be adapted to process images in the 1292 per 964 resolution, however, after some preliminary tests, the network complexity and processing time per image would be prohibitively large. This, in turn, required a pre-processing step where all images were resized to a resolution of 299 per 299. For this particular architecture no segmentation masks or bounding boxes are needed on the training data.

Each epoch consists of training the network with 400 images per class, with a batch size of 32 images and all training images are randomly shuffled between each epoch, in order not to bias the

[1] keras.io

weight update to a specific class. Validation consists of evaluating 200 images per class. *TPR* is calculated by averaging the *TPR* for each of the classes to be identified.

5.1.1 ADAM aerial validation

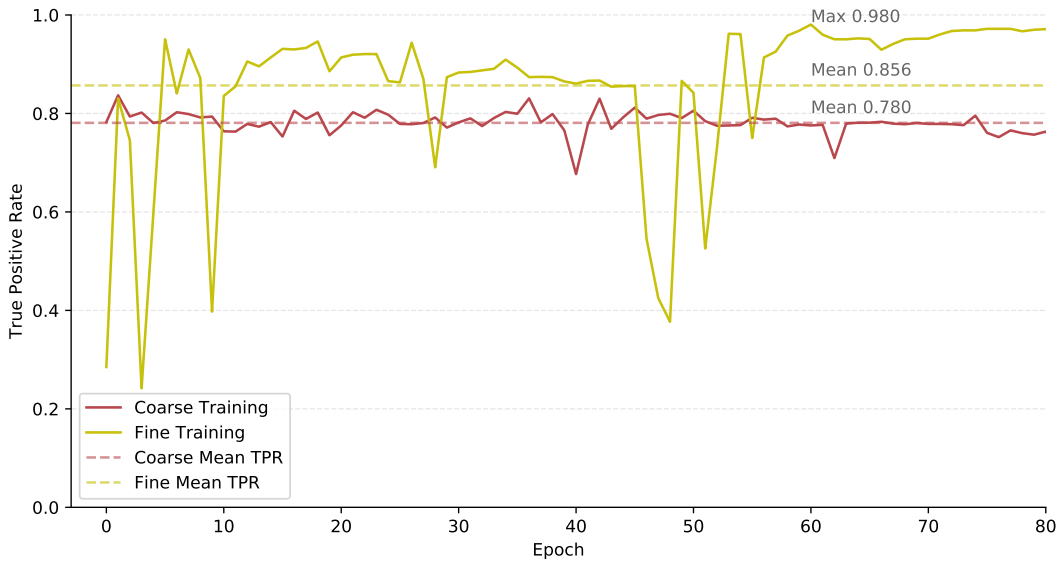


Figure 5.1: ***Coarse*** *and* ***Fine*** *approaches compared using the ADAM optimizer,* ***aerial*** *dataset 8 objects*

From Figure 5.1 it can be seen that there's a difference in the True Positive Rate between the ***Fine*** and ***Coarse*** training approaches.

The ***Coarse*** approach starts from Epoch 0 with a *TPR* of 78.2%, higher than the values obtained in the Conventional Machine Learning methods. However, that value doesn't improve, as it would be expected to with the increase in training epochs. The network performance peaks on Epoch 1, with 83.7% with an average of 78.0% for 80 epochs.

On the other hand, the ***Fine*** approach is more inconstant, having big performance jumps, but as a whole it obtains better results, with an average of 85.6% and a peak of 98.0%. It appears to have plateaued in epoch 60, as it stays over 90% *TPR* for more than 20 epochs. However, its instability raises some doubts regarding its usability, there isn't a concrete relationship between its performance and the number of epochs.

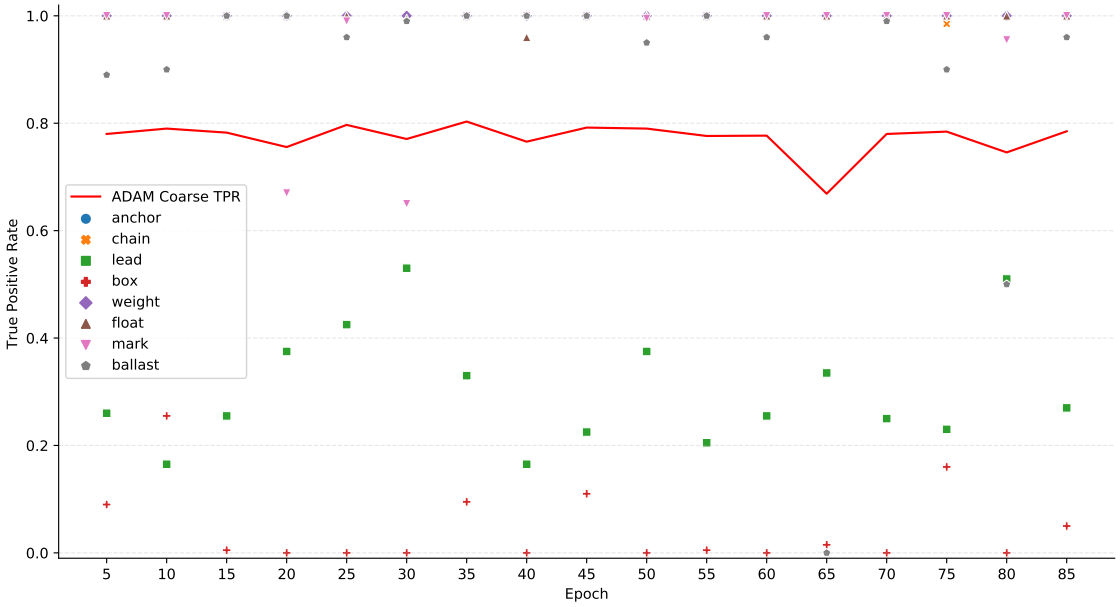


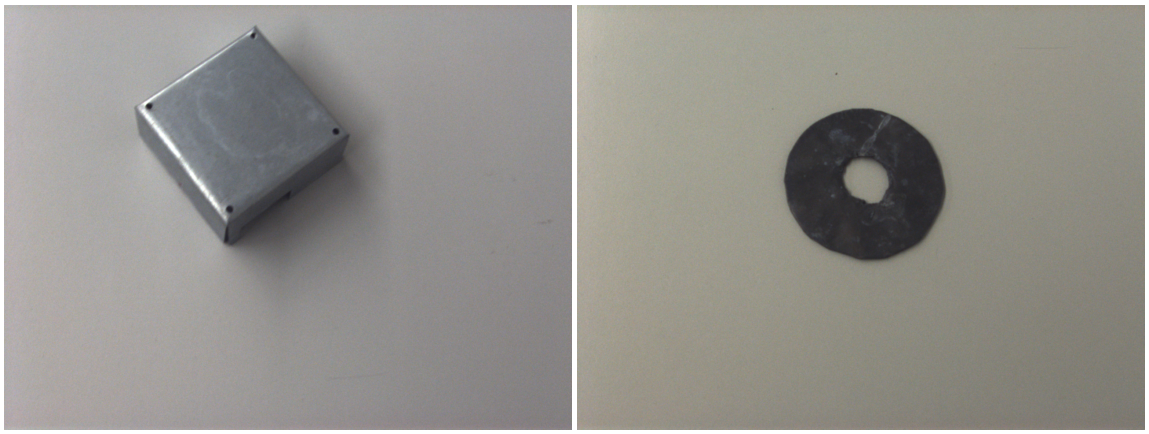
*Figure 5.2: TPR evolution per epoch obtained with the **Coarse** approach (using) ADAM in the **aerial** dataset with 8 objects*

Fig. 5.2 gives a deeper insight on how the network behaves with each of the objects when trained with ADAM and the ***Coarse*** approach. Analyzing the general trend of the graph, 2 objects stand out negatively due to their *TPR* being significantly below average in all epochs: *box* and *lead*. The *box* being the worst, there is only one iteration where its *TPR* is higher than 20%, staying most iterations close to 0%. Despite its higher *TPR*, the *lead* class has an oscillatory behaviour over the epochs, with a highest *TPR* of 53%.

Looking into the ImageNet classes that were used to pre-train the network[2] it can be seen that there is a much higher number of classes with complex, natural shapes (animals, for example) than there are of (perfectly) regular geometrically shaped, which in turn will bias those features towards more complex objects since the features created in the Representation Learning part of Convolutional Neural Networks are based on the shape of objects [66].

The shape of the *box* and *lead* might be indicative about the reason of the decreased performance of those two classes.

[2] https://gist.github.com/xkumiyu/dd200f3f51986888c9151df4f2a9ef30



(a) Box class *(b) Lead class*

Figure 5.3: The two classes with lowest performance: box and lead

It can be seen from Fig. 5.3 that both classes have a (rough) geometric shape, a square and a circle, which can be the reason for this lack of performance. This theory can be corroborated if the *TPR* of the *box* and *lead* classes increases after retraining the feature creation part of CNN, which is done in the ***Fine*** approach.

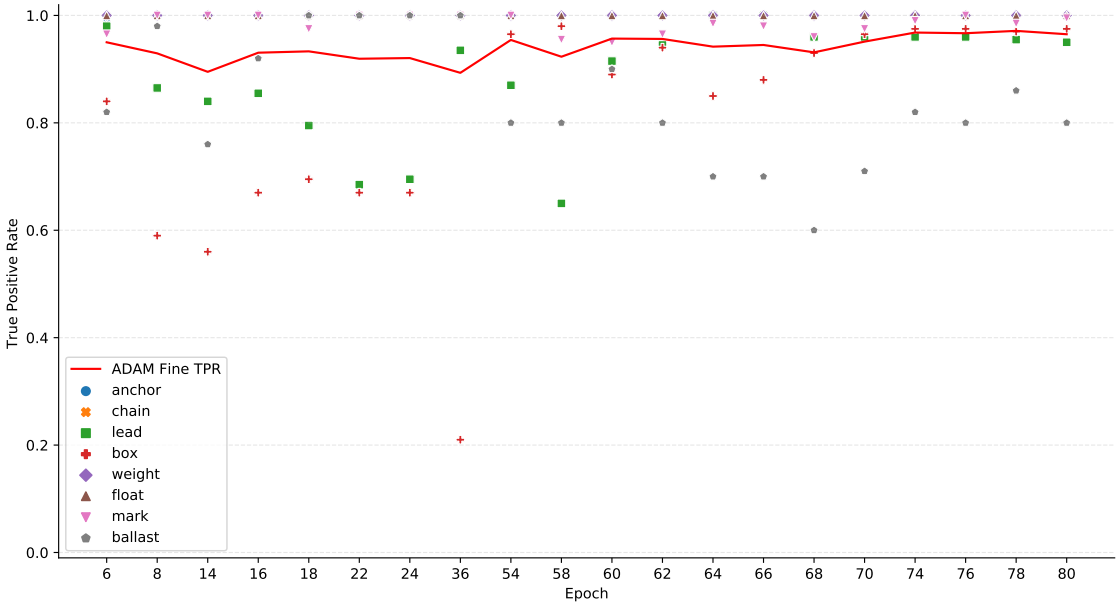


*Figure 5.4: Top 20 epochs using the **Fine** approach and ADAM with 8 objects*

Figure 5.4 shows the 20 iterations with the highest average *TPR*, using the ***Fine*** approach and ADAM. A generalized increase in performance is to be noted, where individual object *TPR* is over 50% in 19 of the top 20 epochs. It can be seen that the *TPR* of classes *box* and *lead* has improved considerably, their lowest *TPR* being 21% and 65%, respectively, reaching 98% and 100% in some iterations. This supports the hypothesis that ImageNet features aren't adequate for regular, geometric shape, as a better performance was obtained by retraining the network features.

5.1.2 SGD aerial validation

Following this evaluation of both ***Coarse*** and ***Fine*** training approaches with the ADAM optimizer, the same comparison can be performed using Stochastic Gradient Descent.

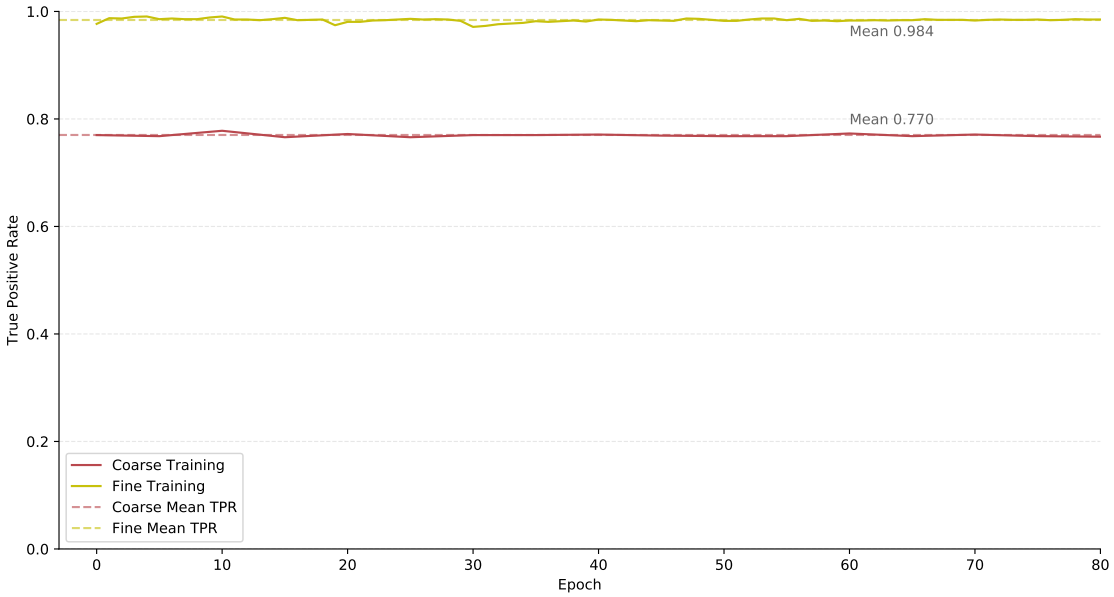
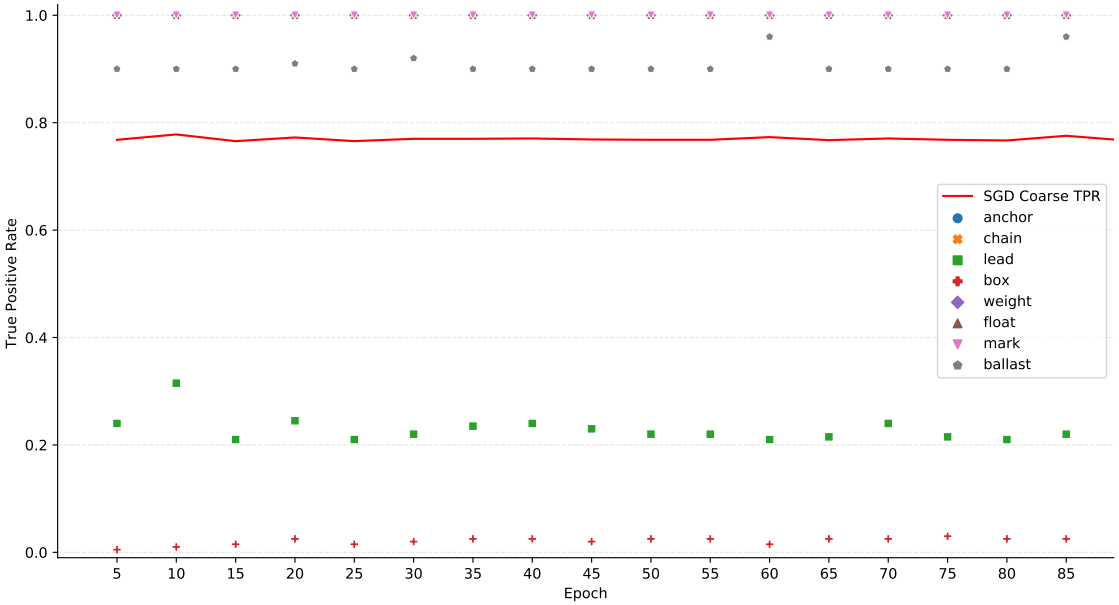


*Figure 5.5: Comparison between the **Coarse** and **Fine** approaches using the SGD optimizer in the **aerial** dataset and 8 objects.*

It can be seen from Fig. 5.5 that the SGD optimizer has a much more constant and predictable behaviour.

The ***Coarse*** approach, despite having a lower average *TPR* (77%) than with the ADAM optimizer, could be seen as more stable, as it doesn't have any significant variations in its value. Still, the value of *TPR* seems to be stagnated, there's no improvement with the increase in epochs, meaning that, as happened with the ADAM ***Coarse*** approach, there are some objects the network has difficulties classifying.

*Figure 5.6: Performance per object with the **Coarse** approach, SGD and **aerial** dataset: 8 classes*



As expected, global average *TPR* is being reduced by the same objects as in the ADAM evaluation: *box* and *lead*.

On the other hand, Fig. 5.7 shows that the ***Fine*** approach brings a very significant improvement, with an average *TPR* of 98.8%, the highest of the 4 experiments performed. Still, it can be seen from the same figure that the model has a lower *TPR* in the *lead* class, with a *TPR* of 88.5% which, despite being the class with the lowest *TPR*, achieves a higher performance than in all other approaches.

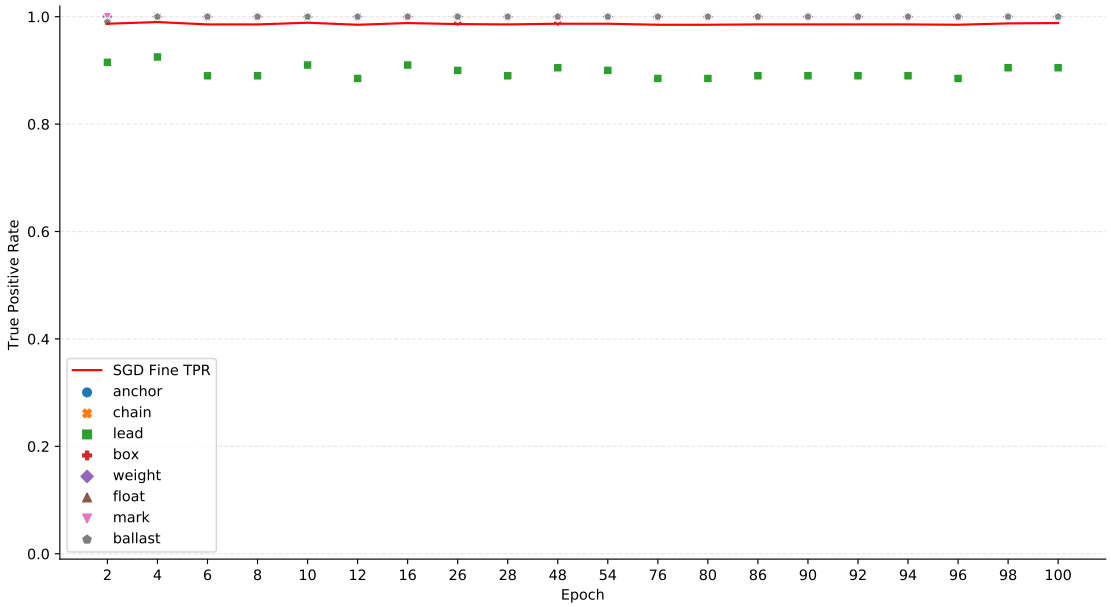


*Figure 5.7: Top 20 epochs using the **Fine** approach and SGD with 8 objects*

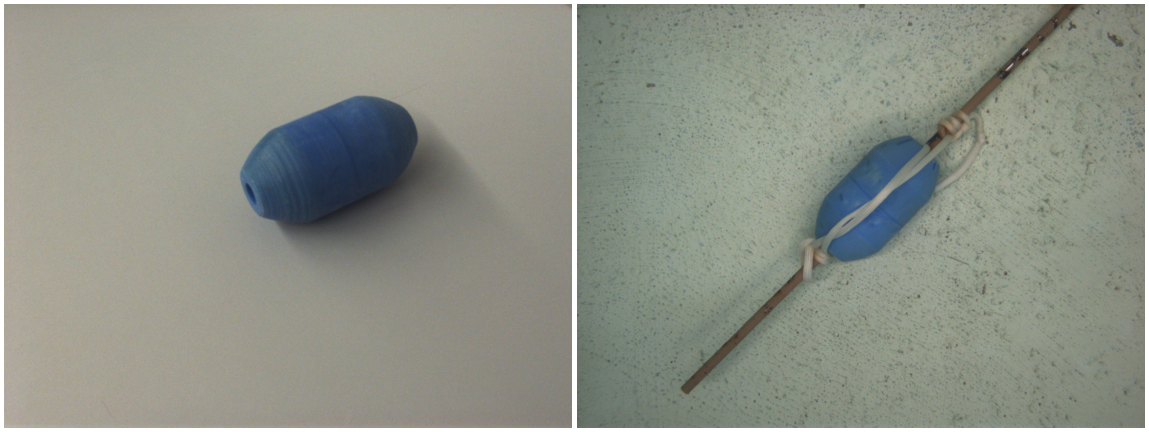
Object	Aerial	Aquatic	Difference
anchor	1.000	0.990	0.010
chain	1.000	1.000	0.000
lead	0.905	0.680	0.225
box	0.995	0.985	0.010
weight	1.000	1.000	0.000
float	1.000	0.000	1.000
mark	1.000	1.000	0.000
ballast	1.000	1.000	0.000
Total	98.7	83.18	15.54

Table 5.2: TPR drop per object from aerial to aquatic dataset on epoch 98

5.1.3 Validation on the Aquatic Dataset

The air-trained model with the ***Fine*** approach and SGD as optimizer was then validated on the aquatic dataset. Fig. 5.9 shows the underwater performance of the air-trained models with the increase in training epochs, reaching a peak *TPR* of 83.1% on epoch 98 and an average of 81.6%. As reference, that same epoch obtained 98.8% *TPR* on the ***aerial*** dataset, resulting in a performance drop of 15.7%.

From Fig. 5.9 and Table 5.2 it can be seen that 4 classes were minimally affected by the change in environment: *ballast*, *mark*, *weight* and *chain*. Class *box*, despite having a very high *TPR*, shows some oscillation, although smaller than 4% peak-to-peak. The *anchor* also shows a small difference on that specific epoch, but the improvement of that object's with the epochs is visible. On the other hand, the *lead* class decreases with epochs, leading to a drop of 22.5% in *TPR*. Contrary to all other objects, the *float* class isn't even detected in a single image, with a 0% *TPR*. The behaviour of the *float* class can be can be explained by comparing the images that constitute the *float* class in the aerial and the aquatic dataset.



*(a) Aerial **float*** *(b) Aquatic **float***

Figure 5.8: Class with the lowest detection rate in the underwater environment: float

As it can be seen from Figure 5.8 despite the same object being in both images, the underwater

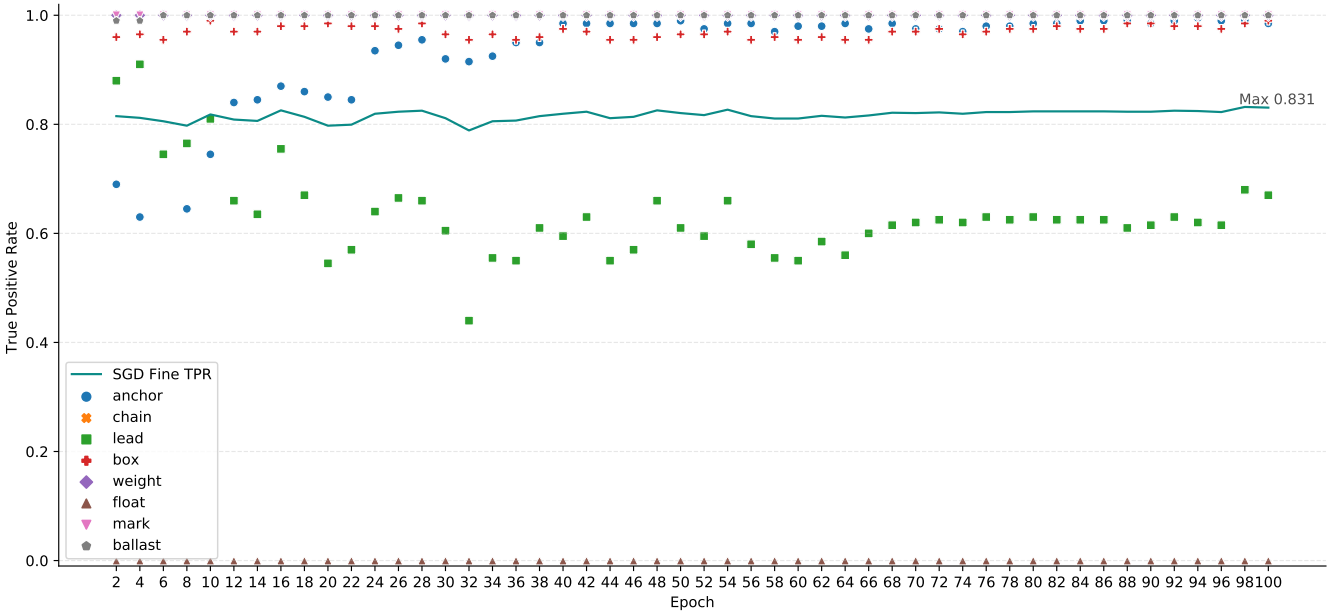


*Figure 5.9: Air-Trained **Fine** SGD network evaluated on the **aquatic** dataset*

picture has the addition of more elements which might be associated to the shape of the object and thus negatively influencing the classification. In the collection of images of the *float* object in the underwater environment, due to its floating nature, it was necessary to attach several heavy objects to weight it down and prevent it from floating, resulting in the object in Fig. 5.8b.

5.1.4 Training with the Aquatic Dataset

For the aquatic training and evaluation tests, the air-trained model with the highest *TPR* in the aquatic validation was chosen. The network was trained maintaining its parameters, namely the optimizer, SGD, and following the ***Fine*** approach, which allowed the network to train its features. Fig. 5.10 shows the results of training that network during 50 epochs. As it can be seen, the value of *TPR* of the model trained with SGD and aquatic images is quite similar to the *TPR* of the air-trained on the same aquatic dataset, with an average of 83.9% and 83.1% respectively and, although there is an improvement of *TPR* with epochs, it is very reduced, around 1% in 40 epochs.

However, when using different optimizers, the network behaves differently. With ADAM it can seen that, similarly to when the model was being trained with aerial images, it has an unpredictable oscillatory behaviour, seemingly stabilizing at a certain value and suddenly dropping. Nonetheless, it reached a higher *TPR* than SGD, peaking at 92.2%.

RMSProp, another optimization algorithm of which ADAM is based on, was also tested exclusively for the aquatic training part, achieving the highest *TPR* on this test, 98.6%. This value, the highest obtained in the aquatic dataset, is important as it shows that all classes in this dataset are identifiable.

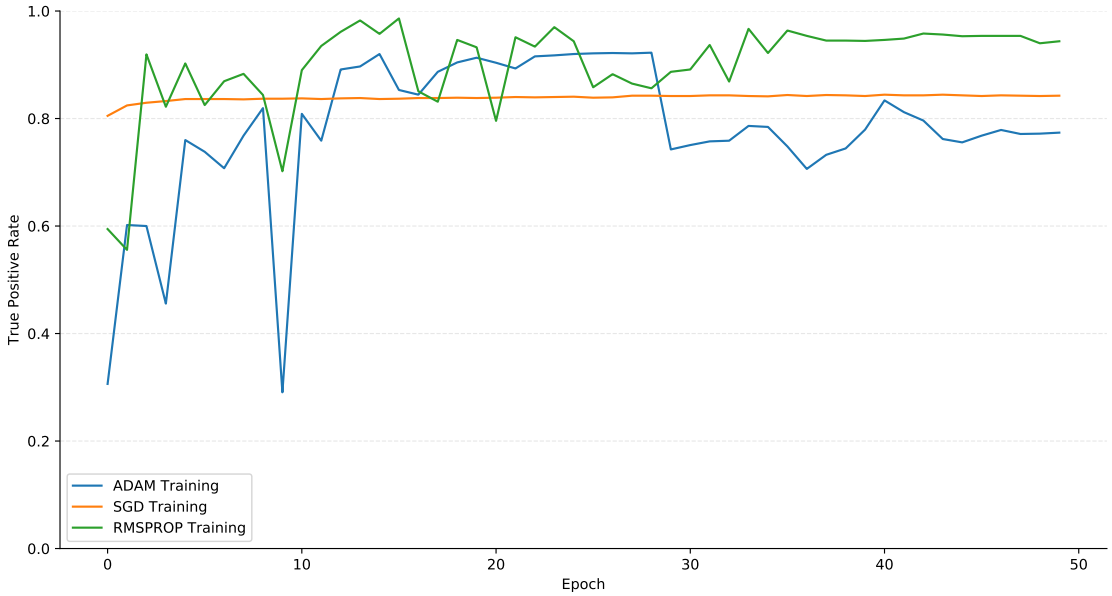


*Figure 5.10: Optimizer comparison: Incremental training with **aquatic** images*

5.2 Conclusion

On this chapter a ConvNet architecture, Inception-V3, from the state-of-the-art in image classification was tested on the custom dataset collected for this thesis.

With a *transfer learning* approach, a pre-trained Inception-V3 architecture was fine-tuned, first with aerial images from the dataset and its performance was evaluated on images also taken in the aerial environment. Two optimizers were compared: Stochastic Gradient Descent and ADAM combined with 2 training approaches: ***Fine*** and ***Coarse***. In ***Coarse*** approach, only the last 3 layers of the network were trained, accounting for 2.1 million parameters of the 23.9 million of the Inception-v3 architecture. In the ***Fine*** approach, 13.2 million parameters were trained, training the last 3 layers of the network and 2 of its 11 Inception modules. Despite training more parameters, the ***Fine*** approach had a better performance. In combination with the SGD optimizer, the best *TPR* obtained in the aerial dataset was 98.7% .

When the model with the best performance in the aerial domain was tested on images taken underwater, its *TPR* dropped to 83.18%. This drop in accuracy was due to some specific classes, *lead* and *float*, as the other classes were almost always correctly classified. As a means of improving performance in the aquatic domain, that model was then trained with images from the aquatic dataset. Keeping the model in the same conditions (SGD optimizer and ***Fine*** approach) there was no improvement in the network, with an average *TPR* of 83.9%. However, better results

Table 5.3: Best result obtained on the aquatic dataset, RMSProp optimizer

Objects	anchor	chain	lead	box	weight	float	mark	ballast
TPR	1.00	1.00	0.94	0.97	1.00	0.98	1.00	1.0

were obtained when changing the optimization algorithm. ADAM, despite having an oscillatory behaviour obtained a *TPR* of 92.3% and RMSProp, an algorithm not used in the aerial validation attained the top *TPR* of 98.6% on the aquatic dataset.

Conclusion points:

- When using a transfer learning approach, it is imperative to also train the representation (feature) learning layers of the network, it is not guaranteed that the existing features are representative of the new classes to identify, as seen on the objects with regular geometric shapes, which had lower classification rates.

- Stochastic Gradient Descent has a very stable classification rate with the increase in training epochs. While that was positive in the aerial dataset, as it reached almost maximum *TPR* in few iterations, that stability was detrimental in the aquatic training since the network's accuracy stagnated in a sub-optimal value. On the other hand, ADAM (and RMSProp) had a less predictable behavior with the increase in training epochs, not demonstrating a proportional relationship between accuracy and number of epochs, reaching high classification rates in some epochs but with a highly oscillatory behaviour.

Chapter 6

Conclusion

The main objective of this thesis was to compare the effects of changing from the aerial to the aquatic domain on the performance of both Conventional (neural networks and support vector machines) and Deep Machine Learning techniques applied to a visual classification task. No dataset was available for this purpose, thus the need to curate one.

This thesis presented ***HEIMDACA***, a novel dataset composed of 8 objects in two different domains: ***aerial*** (images taken above water) and ***aquatic*** (images taken underwater). After the collection of the dataset it was possible to proceed to the evaluation of the image classification techniques.

Starting with Conventional Machine Learning, two classifiers were tested: Neural Networks and Support Vector Machines. A feature-based approach using SIFT as the feature extractor was used along with a Bag-of-Visual-Words approach as a means of reducing dimensionality and providing robustness to the input data. The models were first trained and evaluated using ***aerial*** images. The model with the highest *TPR* was then evaluated on ***aquatic*** images, measuring the impact of the change in domain on the accuracy of the model. Models were then retrained with images from the ***aquatic*** environment, with hopes of increasing their performance on that same domain. When testing with 8 objects, the best air-trained models tested on the ***aerial*** domain achieved a *TPR* of 63.1% for the Neural Networks and 58.5% for the Support Vector Machines. Testing those same models on the ***aquatic*** domain resulted in a tremendous drop in performance, with a *TPR* of 14.3% for the Neural Network and 16.7% on the Support Vector Machine, a drop of 48.8% and 41.8%, respectively. Retraining that model with ***aquatic*** images increased the *TPR* in that domain to 40.3% and 29.2%, for the NN and SVM respectively. With this result it can be concluded that, with **Conventional machine learning methods and the approach taken, models can't be directly adapted from the aerial to the aquatic domain**. These Conventional Machine Learning classifiers leave much to be desired, as most of their performance is lost from changing the environment. It can be speculated that this result was negatively influenced by the imaging conditions introduced by the aquatic environment. Due to the scattering effect and other phenomena related to light propagation in water, images take underwater can become blurred and, since a feature-based approach was used, if the feature detector can't detect those keypoints or if they are

altered, the classifiers will fail.

A Deep Learning architecture, Inception-V3, from the state-of-the-art in image classification was also tested on this dataset. Two fine-tuning approaches were compared on a network with pre-trained weights: ***Coarse***, where the *representation learning* layers of the network weren't trained, only combining the already existing features of the network; and ***Fine*** where new features were trained. Two gradient optimization algorithms were also compared, Stochastic Gradient Descent and ADAM. A significant conclusion could be taken by comparing the two approaches. Using the SGD as optimizer, the ***Coarse*** approach managed an average of 77.0% *TPR* on the aerial dataset, while the ***Fine*** had an average of 98.4% *TPR*. The difference between the two approaches was found to happen in the classes with a regular, geometric shape: *box* and *lead*, a square and a disk, respectively. The same result was seen with the ADAM optimizer. From this results it can be speculated that the pre-trained ImageNet weights are not prepared to classify objects with this type of shape, as that dataset is more focused on organic shapes, such as those of animals. Between the two gradient optimizers, ADAM was found to be too unstable, seemingly converging to a specific value of *TPR*, but suddenly dropping in performance. SGD is a better choice, for the aerial dataset.

When an air-trained network (following the ***Fine*** approach and with SGD) was evaluated on the ***aquatic*** dataset, it managed a *TPR* of 83.1%, a drop of 15.54%. That same type of network was then incrementally trained with ***aquatic*** images, but no improvement on accuracy was recorded. It can be hypothesized that the optimizer stagnated in a local minimum of the loss function, not being able to evolve past that point due to the learning rate being too small. However, when changing the gradient optimizer to either ADAM or RMSProp, the maximum *TPR* recorded was 92.3% and 98.6%, respectively, on the ***aquatic*** environment. While the *TPR* remained constant when training with SGD, ADAM, although unpredictably oscillatory, managed to take the network out of a local minimum.

Concluding, the Deep Learning network completely outperformed the Conventional Machine Learning methods on this specific dataset. With a drop of only 15.54% when translating an air-trained network to an aquatic domain, the Inception-V3 seems the best choice for visual classification tasks.

6.1 Future work and remarks

Adding more objects to the dataset would result in a more complete study. For example, a "black-box" similar to the one used by Bonin-Font (2015)[35] can be an addition to the dataset. Additionally, it would be interesting to add more images in different conditions, for example, by incrementally varying the turbidity conditions, the image classification models could be tested for robustness.

If collecting images from new or the same objects isn't possible, the dataset can be virtually augmented by data augmentation techniques such as the ones presented by Huang et al. (2019)[67]. The authors augment a small publicly available dataset for marine wildlife recognition by simulating some effects of the aquatic environment on imaging conditions, such as turbulence,

lighting intensity and different camera shooting angles.

During this work, it was observed that objects with light hotspots tended to generate features (keypoints) in those same hotspots resulting in image classification models that had false or incorrect information, reducing their robustness. It is recommended that additional care should be taken when collecting images from objects with reflective surfaces, for example, by manipulating light sources so as to have a more uniform lighting. A misdirected light source can also contribute to shadows in the image which are also a source of irrelevant points of interest, that don't contribute to the robustness of the model.

Appendix A

Conventional Machine Learning classifiers parameter grid

Table A.1: Classifiers and hyperparameter values sweeped

Classifier			
NN		***SVM***	
Hyperparameter	***Value***	***Hyperparameter***	***Value***
Layer Sizes	100 500 100, 20 100, 50 100, 50, 10 500, 100, 10	*C*	0.1 0.316 1.0 3.16 10
Learning Rate	0.001 0.01 0.1 1	*Gamma*	0.1 0.316 1.0 3.16 10
Activation Function	ReLu Logistic		
Optimizer	Adam SGD		

Appendix B

Overview of the Deep Learning framework used

The increased interest in deep neural network architectures has given rise to a whole ecosystem dedicated to the optimization and development of these networks. Since deep neural networks have a number of parameters in the order of the millions, an efficient use of computational power is a must. Luckily, the basis of CNN, the convolutional layer is an embarrassingly parallel [1] task (e.g. all operations involved in convolving a filter with an image can be executed in parallel) which, in combination with GPU processing, makes the foundation of modern machine learning. Graphics Processing Units (GPU), also known as graphic cards, were originally created to accelerate the output to a display device and have a highly parallel structure. Modern GPUs can have thousands of processing units (cores)[2], making them the ideal candidate for parallel tasks. Some vendors, particularly NVIDIA, provide a great amount of support to developers interested in general computing on GPUs, having released and actively developing their own GPU parallel computing and programming model, **CUDA**[3][4], which powers many deep learning libraries. However, CUDA programming has a high learning curve, requiring some specialized knowledge in the inner workings of GPU for maximum code efficiency, which has lead to the creation of some open-source libraries that offer a more high-level API [5] for deep learning tasks.

Tensorflow [6] is an open-source library developed by Google for general numerical computation tasks, as well as machine learning. It was first used exclusively in-house and then made available to the general public in 2015, supporting both CPU and GPU execution, with an interface to CUDA. In spite of being one level of abstraction above CUDA, its focus is customizability, not convenience, being a very complete API, but not the first choice for someone starting in the machine/deep learning field.

[1] https://en.wikipedia.org/wiki/Embarrassingly_parallel
[2] https://www.nvidia.com/en-us/geforce/products/10series/compare/
[3] https://developer.nvidia.com/cuda-zone
[4] Compute Unified Device Architecture
[5] Application Programming Interface
[6] https://www.tensorflow.org/

Having convenience and fast experimentation in mind, Keras[7] was created, an even higher layer of abstraction on top of TensorFlow and other similar machine learning APIs such as Theano[8] or CNTK[9]. Table B.1 shows software installed and their version. Python 3 was used in the entirety of the work presented in this section. Networks were trained and evaluated in a computer with an Intel Core i5-8600k @ 3.6GHz CPU and an NVIDIA GeForce GT 1030, with 2GB of VRAM.

Table B.1: Software versions used

Tool	***Version***
NVIDIA driver	418.43
cuDNN	7.1.4
CUDA	9.0
TensorFlow	1.12.0
Keras	2.2.4
NumPy	1.16.3

[7] https://keras.io/
[8] http://www.deeplearning.net/software/theano/; Montreal Institute for Learning Algorithms
[9] Microsoft Cognitive Toolkit; https://github.com/microsoft/CNTK

References

[1] Silvia Corchs and Raimondo Schettini. Underwater image processing: State of the art of restoration and image enhancement methods. *Eurasip Journal on Advances in Signal Processing*, pages 1–14, 2010.

[2] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[4] David T Sandwell, R Dietmar Müller, Walter HF Smith, Emmanuel Garcia, and Richard Francis. New global marine gravity model from CryoSat-2 and Jason-1 reveals buried tectonic structure. *Science*, 346:65–67, 2014.

[5] Michael P Hayes and Peter T Gough. Synthetic aperture sonar: a review of current status. *IEEE Journal of Oceanic Engineering*, 34(3):207–224, 2009.

[6] Graham J Edgar and Rick D Stuart-Smith. Systematic global assessment of reef fish communities by the Reef Life Survey program. *Scientific Data*, 1:140007, 2014.

[7] Karel Zuiderveld. Contrast limited adaptive histogram equalization. *Graphics gems*, pages 474–485, 1994.

[8] Rafael Garcia and Nuno Gracias. Detection of interest points in turbid underwater images. In *OCEANS, 2011 IEEE-Spain*, pages 1–9. IEEE, 2011.

[9] Jules S. Jaffe. Computer Modeling and the Design of Optimal Underwater Imaging Systems. *IEEE Journal of Oceanic Engineering*, 15(2):101–111, 1990.

[10] Zhishen Liu, Yifan Yu, Kailin Zhang, and Hailong Huang. Underwater image transmission and blurred image restoration. *Optical Engineering*, 40(6):1125–1132, 2001.

[11] R. Garcia, T. Nicosevici, and X. Cufi. On the way to solve lighting problems in underwater imaging. *Oceans '02 MTS/IEEE*, 2(1):1018–1024, 2002.

[12] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[13] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[14] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, volume 11, page 2, 2011.

[15] Ondrej Miksik and Krystian Mikolajczyk. Evaluation of local detectors and descriptors for fast feature matching. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 2681–2684. IEEE, 2012.

[16] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[17] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[18] Simon S Haykin. *Neural networks and learning machines*, volume 3. Pearson education Upper Saddle River, 2009.

[19] Joost Van Doorn. Analysis of deep convolutional neural network architectures. In *Proceedings of the Twenty First Twente Student Conference on IT, Enschede, The Netherlands*, pages 1–7, 2014.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[21] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[23] Diederik P Kingma and Jimmy Ba. ADAM: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[25] Yuanqing Lin, Fengjun Lv, Shenghuo Zhu, Ming Yang, Timothee Cour, Kai Yu, Liangliang Cao, and Thomas Huang. Large-scale image classification: fast feature extraction and SVM training. In *CVPR 2011*, pages 1689–1696. IEEE, 2011.

[26] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[27] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[28] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[31] Michael Albert Thornton. *A survey and engineering design of atmospheric diving suits*. PhD thesis, Texas A&M University, 2000.

[32] John J Schultz, Carrie A Healy, Kenneth Parker, and Bim Lowers. Detecting submerged objects: the application of side scan sonar to forensic contexts. *Forensic science international*, 231(1-3):306–316, 2013.

[33] Avilash Sahoo, Santosha K Dwivedy, and PS Robi. Advancements in the field of autonomous underwater vehicle. *Ocean Engineering*, 181:145–160, 2019.

[34] Marco Jacobi. Autonomous inspection of underwater structures. *Robotics and Autonomous Systems*, 67:80–86, 2015.

[35] Francisco Bonin-Font, Gabriel Oliver, Stephan Wirth, Miquel Massot, Pep Lluis Negre, and Joan-Pau Beltran. Visual sensing for autonomous underwater exploration and intervention tasks. *Ocean Engineering*, 93:25–44, 2015.

[36] Michael J Swain and Dana H Ballard. Indexing via color histograms. In *Active Perception and Robot Vision*, pages 261–273. Springer, 1992.

[37] Mehdi Fatan, Mohammad Reza Daliri, and Alireza Mohammad Shahri. Underwater cable detection in the images using edge classification based on texture information. *Measurement*, 91:309–317, 2016.

[38] John Canny. A computational approach to edge detection. In *Readings in computer vision*, pages 184–203. Elsevier, 1987.

[39] Dana H Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.

[40] Donghwa Lee, Gonyop Kim, Donghoon Kim, Hyun Myung, and Hyun Taek Choi. Vision-based object detection and tracking for autonomous navigation of underwater robots. *Ocean Engineering*, 48:59–68, 2012.

[41] John P Lewis. Fast template matching. In *Vision interface*, volume 95, pages 15–19, 1995.

[42] G Santhan Kumar, Unnikrishnan V Painumgal, MNV Chaitanya Kumar, and KHV Rajesh. Autonomous Underwater Vehicle for Vision Based Tracking. *Procedia computer science*, 133:169–180, 2018.

[43] Concetto Spampinato, Simone Palazzo, Pierre-Hugues Joalland, Sébastien Paris, Hervé Glotin, Katy Blanc, Diane Lingrand, and Frédéric Precioso. Fine-grained object recognition in underwater visual data. *Multimedia Tools and Applications*, 75(3):1701–1720, 2016.

[44] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):743–761, 2011.

[45] Karsten Behrendt and Libor Novak. A Deep Learning Approach to Traffic Lights: Detection, Tracking, and Classification. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017.

[46] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[47] George A Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[48] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[49] Jin Li, Peng Wang, Yanzhao Li, Yang Zhou, Xiaolong Liu, and Kuan Luan. Transfer Learning of Pre-Trained Inception-V3 Model for Colorectal Cancer Lymph Node Metastasis Classification. In *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1650–1654. IEEE, 2018.

[50] Alessandra Lumini and Loris Nanni. Deep learning and transfer learning features for plankton classification. *Ecological Informatics*, 51:33–43, 2019.

[51] Michael Bewley, Ariell Friedman, Renata Ferrari, Nicole Hill, Renae Hovey, Neville Barrett, Ezequiel M Marzinelli, Oscar Pizarro, Will Figueira, Lisa Meyer, et al. Australian sea-floor survey data, with images and expert annotations. *Scientific data*, 2:150057, 2015.

[52] Muwei Jian, Qiang Qi, Hui Yu, Junyu Dong, Chaoran Cui, Xiushan Nie, Huaxiang Zhang, Yilong Yin, and Kin-Man Lam. The extended marine underwater environment database and baseline evaluations. *Applied Soft Computing*, 80:425–437, 2019.

[53] Andre B Figueiredo, Bruno M Ferreira, and Aníbal C Matos. Vision-based localization and positioning of an AUV. In *OCEANS 2016-Shanghai*, pages 1–7. IEEE, 2016.

[54] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Prentice Hall, 2008.

[55] Tomasz Łuczyński, Max Pfingsthorn, and Andreas Birk. The pinax-model for accurate and efficient refraction correction of underwater cameras in flat-pane housings. *Ocean Engineering*, 133:9–22, 2017.

[56] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.

[57] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, et al. Fiji: an open-source platform for biological-image analysis. *Nature methods*, 9(7):676, 2012.

[58] Agnieszka Mikołajczyk and Michał Grochowski. Data augmentation for improving deep learning in image classification problem. In *2018 international interdisciplinary PhD workshop (IIPhDW)*, pages 117–122. IEEE, 2018.

[59] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[60] Ryan Rifkin and Aldebaro Klautau. In defense of One-vs-All classification. *Journal of machine learning research*, 5(Jan):101–141, 2004.

[61] MY Rafiq, G Bugmann, and DJ Easterbrook. Neural network design for engineering applications. *Computers & Structures*, 79(17):1541–1552, 2001.

[62] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[63] Haithem Hermessi, Olfa Mourali, and Ezzeddine Zagrouba. Deep feature learning for soft tissue sarcoma classification in MR images via transfer learning. *Expert Systems with Applications*, 120:116–127, 2019.

[64] Claudia Mazo, Jose Bernal, Maria Trujillo, and Enrique Alegre. Transfer learning for classification of cardiovascular tissues in histological images. *Computer methods and programs in biomedicine*, 165:69–76, 2018.

[65] Antonio Torralba, Alexei A Efros, et al. Unbiased look at dataset bias. In *CVPR*, volume 1, page 7. Citeseer, 2011.

[66] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[67] Hai Huang, Hao Zhou, Xu Yang, Lu Zhang, Lu Qi, and Ai-Yun Zang. Faster R-CNN for marine organisms detection and recognition using data augmentation. *Neurocomputing*, 2019.