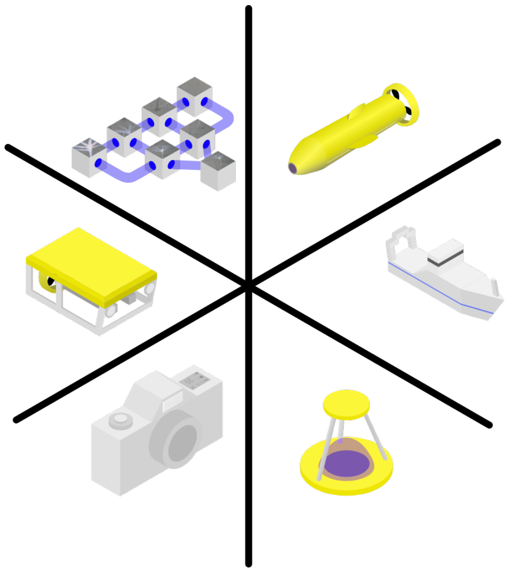
Data-driven models for taxonomic classification in marine science



Daniel Langenkämper
Bielefeld, Germany

Biodata Mining Group
Faculty of Technology
Bielefeld University

Der Technischen Fakultät der Universität Bielefeld vorgelegte Dissertation zur Erlangung des akademischen Grades: Doktor der Ingenieurswissenschaften (Dr.-Ing.)

Gedruckt auf alterungsbeständigem Papier nach ISO 9706.

Originality of Work: I hereby declare that all the work and ideas presented in this thesis are my own or that I have otherwise marked and/or cited all other ideas and the sources I rely on.

Bielefeld, October, 2019

Daniel Langenkämper

Dedicated to the memory of my father,
Konrad Langenkämper,
who always believed in me.

Most important contributions and findings

How much data is necessary?

We investigated, how much data is necessary to train conventional classifiers, e.g. SVM and deep learning classifiers to classify marine imaging datasets reliably. Conventional learners need few data with KNN ($\sim 10^1$) and the SVM ($\sim 10^2$). Deep learning architectures need significantly more data ($\sim 10^3$) to reliably classify. However, we could successfully employ CNNs with less data, at the cost that they are prone to large fluctuations in accuracy.

COATL – An assistance system for marine image annotation

We propose COATL, an assistance system for marine image annotation, based on a staged classifier architecture. COATL focuses on a transparent presentation of aids and real-time responses. We combine shallow learners especially the newly proposed H^{2}SOL with the deep learners and basic object detection in an intelligent fashion to yield a system that helps users to efficiently and effectively annotate marine image volumes. [Langenkämper and Nattkemper, 2016]

H^{2}SOL algorithm

We present the H^{2}SOL, an alternative to the widely used SVM. It needs a little more data, but is orders of magnitude faster than the SVM in training and classification. Moreover, we showed that it can be more accurate in the marine imaging context. [Langenkämper and Nattkemper, 2016].

Citizen Science object detection and deep learning

In a first-time study, we combined Citizen Science (CS) for detecting objects in marine imaging with deep learning classification. Using the CS data, we could simulate enough data to analyze the errors expected to occur in a real-world experiment. Furthermore, we provide recommendations for designing a Citizen Science study, resulting from the results of our error assessment. [Langenkämper, Simon-Lledó, et al., 2019]

Concept Drift

If we want to re-use already trained classifiers with a new but related dataset, concept drift is a problem to solve. We could show that by finetuning a classifier with 30% of data from the new dataset an accuracy of above 90% could be achieved. This is a maximum improvement of roughly 70 basis points over no adaption.

Tackling data scarcity and imbalance

In biological images, especially in marine imaging, some classes are scarce so a considerable imbalance in the distribution of classes is common.

We have proposed balancing rules to determine the number of samples per class in combination with sampling algorithms. We were able to improve the average classification performance of a ResNet deep learning classifier by 13 basis points. Per class the biggest improvement is 40 basis points. We could show that augmentation in combination with sampling strategies provides a significant improvement in classification accuracy. [Langenkämper, Kevelaer, and Nattkemper, 2019]

Published work of the author during PhD studies

Daniel Langenkämper, Alexander Goesmann, and Tim W Nattkemper (2014). "AKE-the Accelerated k-mer Exploration web-tool for rapid taxonomic classification and visualization". In: *BMC bioinformatics* 15.1, p. 384.

Daniel Langenkämper, Tobias Jakobi, et al. (2016). "Comparison of Acceleration Techniques for Selected Low-Level Bioinformatics Operations". In: *Frontiers in genetics* 7.

Daniel Langenkämper, Robin van Kevelaer, and Tim W Nattkemper (2019). "Strategies for Tackling the Class Imbalance Problem in Marine Image Classification". In: *Pattern Recognition and Information Forensics*. Ed. by Zhaoxiang Zhang et al. Cham: Springer International Publishing, pp. 26–36. ISBN: 978-3-030-05792-3.

Daniel Langenkämper and Tim W Nattkemper (2016). "COATL - a learning architecture for online real-time detection and classification assistance for environmental data". In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, pp. 597–602.

Daniel Langenkämper, Erik Simon-Lledó, et al. (June 2019). "On the impact of Citizen Science-derived data quality on deep learning based classification in marine images". In: *Plos One* 14.6, pp. 1–16. DOI: 10.1371/journal.pone.0218086. URL: https://doi.org/10.1371/journal.pone.0218086.

Daniel Langenkämper, Martin Zurowietz, et al. (2017). "BIIGLE 2.0 - Browsing and Annotating Large Marine Image Collections". In: *Frontiers in Marine Science* 4, p. 83.

Torben Möller, Daniel Langenkämper, and Tim W Nattkemper (2019). "Wind turbine segmentation performing kNN-clustering on superpixel segmentations". In:

Andreas Momber et al. (2019). "Monitoring und Zustandsbewertung von Oberflächenschutzsystemen an Offshore-Windenergieanlagen". In: *Korrosionsschutz in der maritimen Technik*, pp. 59–80.

Shan E Ahmed Raza et al. (2016). "Robust normalization protocols for multiplexed fluorescence bioimage analysis". In: *BioData Mining* 9, pp. 1–13.

Timm Schoening, Daniel Langenkämper, et al. (2015). "Rapid image processing and classification in underwater exploration using advanced high performance computing". In: *OCEANS'15 MTS/IEEE Washington*. IEEE, pp. 1–5.

Martin Zurowietz, Daniel Langenkämper, Brett Hosking, et al. (2018). "MAIA - A machine learning assisted image annotation method for environmental monitoring and exploration". In: *PloS one* 13.11, e0207498.

Martin Zurowietz, Daniel Langenkämper, and Tim W Nattkemper (2019). "BIIGLE2Go - A scalable image annotation system for easy deployment on cruises". In: *Proceedings of IEEE OCEANS 2019*.

Papers in preparation/submitted

George Hanke et al. (2019). “Quantifying Macrolitter on the Seafloor: Current Practices and Outlook”. in preparation.

Brett Hosking et al. (2019). “Classification of megafauna in seabed photography using deep convolutional neural networks”. in preparation.

Daniel Langenkämper, Robin van Kevelaer, Autun Purser, et al. (2019). “Gear-induced concept drift in marine images and its effect on deep learning classification”. submitted.

Timm Schoening, Autun Purser, et al. (2019). “Megafauna community assessment of polymetallic nodule fields with cameras: Platform and methodology comparison”. submitted.

Contents

3 Chapter 1 Introduction

4 Section 1.1 Marine images

15 Section 1.2 Automatic annotation

23 Section 1.3 Context of this work and research questions

25 Chapter 2 Data

27 Section 2.1 Hausgarten PS80/179-3

31 Section 2.2 PAP

35 Section 2.3 APEI6

39 Section 2.4 UVP5

43 Chapter 3 Craftsman's marine CV

45 Section 3.1 Pre-processing

53 Section 3.2 Feature extraction

61 Section 3.3 Object detection

63 | Section 3.4 Classification

81 | Section 3.5 Summary

85 | Chapter 4 Marine deep learning

89 | Section 4.1 Feature extraction & classifiers II

107 | Section 4.2 How much data?

111 | Section 4.3 Encoding annotation size in CNNs

115 | Section 4.4 Data scarcity and imbalance

125 | Section 4.5 Concept drift

131 | Section 4.6 Object detection revisited

147 | Section 4.7 Summary

151 | Chapter 5 Annotation Assistance

161 | Section 5.1 Real-time systems

165 | Section 5.2 Adaptive systems

169 | Section 5.3 Summary

171 | Chapter 6
Conclusion

181 | Chapter 7
Appendix

Foreword

Conventions Throughout the text the term “we” is used no matter if the work was only done by me or in collaboration with co-workers. If work was primarily done by a co-worker this is also denoted. In machine learning, datasets for training, optimizing and evaluating a classifier are traditionally named: training set, validation set and test set. Hereby, the training set is used to train the classifier, the validation set is used to optimize the classifier and the test set is a disjoint dataset, which is used to evaluate the classifier and has never been seen by it before. With the advent of deep learning several publications have used the term validation set for the data that has never been seen before by the classifier. Thus, in this work we use the term training set for training the network. The term optimization set is used for optimization and we adapt the deep learning vocabulary, so that the validation set is the set for evaluating a classifier, which has never been seen by it before. Tables are usually color-coded, if this is meaningful. The best result is displayed in **green and bold**. The second best is colored in orange and the third best marked in red.

Acknowledgement I want to thank my supervisor Tim Nattkemper, the current and former members of the Biodata Mining Group, especially Torben Möller, Martin Zurowietz, Jonas Osterloff and Robin van Kevelaer and my collaborators in various projects. Furthermore, I thank my wife for her support.

1

Introduction

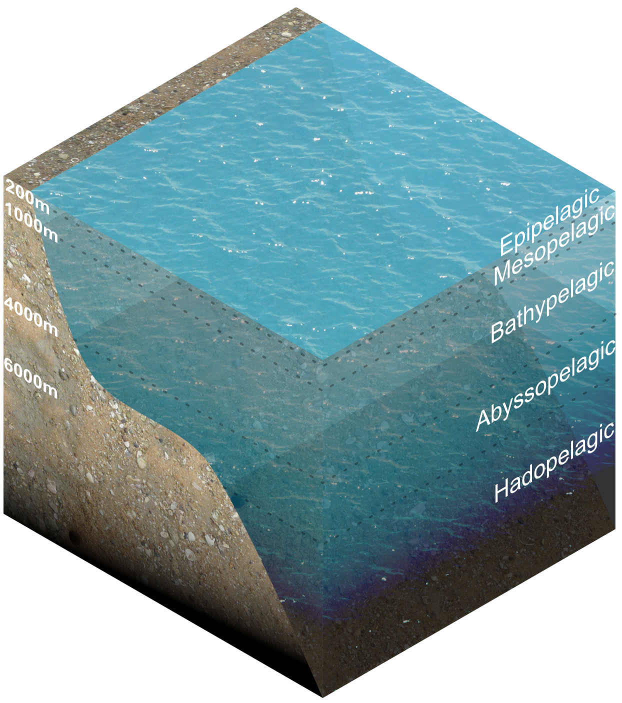
1.1 Ecological data - Marine images

The earth is covered with 70.8% ($361*10^6$km^2) water and 29.2% ($149*10^6$km^2) landmass. Most of the water can be found in the five oceans, i.e. Pacific, Atlantic, Indian, Arctic and Southern Ocean. The water column, also known as pelagic, can be divided into five layers - Epipelagic (0 - 200m depth), Mesopelagic (200-1000m), Bathypelagic (1000-4000m), Abyssopelagic (4000-6000m) and Hadopelagic (6000m+) (cmp. Figure 1). In addition, the seafloor, as well as a part of the subsurface, is called the benthic zone and the layer near the seafloor is called the demersal zone. The mean depth of the oceans is 3.8km, with a maximum depth at the Mariana trench of 11km [Pidwirny, 2006; *Layers of the Ocean* 2018; Steele, 2010].

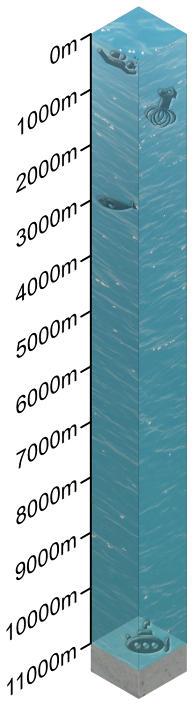
> "Whatever the cause, human beings know more about the surface of that dead rock we call the moon than the living depths of our own planet's seas. We still know very little about the deep ocean floor.' [Flannery, 2007]

Although most of the land mass has already been studied, only 5-20%[1] of the ocean is known today [*How much of the ocean have we explored?* 2018]. Particularly in the deep-sea, where very little information is known due to the difficult conditions and the complex and costly setups for investigating these areas. Research in coastal areas is much more advanced as the public and economic interest in these areas is higher and data collection is easier. In contrast, the moon is already 100% mapped to 7m detail [*China Released Clearest Full Moon Map* 2018]. The sea is essential for the ecology of our planet. It is responsible for 70% of the oxygen in the atmosphere, which is produced by marine plants performing photosynthesis [Nelson, 2019]. In contrast to this important function, it is estimated that 91% of the organisms in the oceans are undescribed [Mora et al., 2011].

[1]This is depending on what known and unexplored mean in that context. 5% is known in comparison to the level of detail of exploration above sea-level. "In broad strokes, the ocean's great mountain ranges, valleys are now defined in maps derived from satellite observations and sonar swaths from research vessels, but only about 5% of the ocean has been explored and mapped with the detail comparable to Earth's above-ocean terrain, or the surface of the Moon, Mars, or Jupiter." - Sylvia Earle 2011



(a)



(b)

Figure 1: Layers of the ocean *a)* Epipelagic (0m - 200m depth), Mesopelagic (200m - 1000m), Bathypelagic (1000m - 4000m), Abyssopelagic (4000m - 6000m) and Hadopelagic (6000m+) [Hedgpeth, 1957]. *b)* A scuba diver was able to reach a maximum depth of 332.35m, while the apnoe (without scuba gear) record is at 253.2m [Limited, 2016; Nitsch, 2017]. Cuvier's beaked whales can reach depths of up to 3000m [Schorr et al., 2014]. The deepest manned submarine dive was to the Mariana Trench at 10928m depth in 2019 [Fitzherbert, 2019]. The deepest part of the ocean is believed to be the Challenger Deep in the Mariana Trench with a depth of 10984m [Gardner et al., 2014].

In an attempt to help amend that disparity, this thesis aims to aid exploration of the oceans by addressing problems that present themselves in the context of manual and automatic annotation of marine imagery.

Marine imaging

Imaging is a non-invasive method for studying marine life. Unlike, for example, trawl fishing studies, no environmental impact is caused. In contrast to invasive microbiological techniques such as metagenomics, only macro- (0.5-50mm) and megafauna (>50mm [Gray and Elliott, 2009]) can be recorded. In the beginning of marine research, imaging techniques were limited to create drawings of the organisms caught in nets using pen and paper. With the advent of cameras, a more unbiased view could be captured. Unfortunately, a photographic film had to be used, which was quite limited in its usability due to its capacity. Furthermore, it had to be developed, which was costly and the results were visible only then. This lack of immediate feedback could mean that images were over- or underexposed, or out of focus and therefore useless. Furthermore, images had usually to be taken manually to select the correct focal point and aperture. With the advent of digital imaging techniques, the acquisition of images has become relatively cheap and easy. The image capture can be initiated via software and the results are directly visible to the user as no development is required. Additional functions such as remotely controlled focus or autofocus help in capturing high-quality images. Other modern imaging techniques such as photography of non-visible light or satellite-based remote sensing exist, but these are out of the scope of this thesis. Therefore, the term "image" in this thesis always refers to a photograph, recording the visible light spectrum.

Imaging underwater, especially at greater depths, requires the use of artificial luminaires such as flashlights or constant lights, as there is no or insufficient natural light available. The configuration of lighting is often crucial to success in capturing a uniformly illuminated, sharp image with natural colors. Digital alteration of images helps to improve flaws in the lighting setup, but often introduce additional artifacts. Therefore, the lighting configuration is of great importance. Naively increasing the lighting strength leads to two problems. Particles in the water column, also known as marine snow, are illuminated and reduce the visual quality of the image, sometimes even causing the autofocus to focus the marine snow instead of the seafloor. In addition, the power is usually limited and must be shared with other sensors on-board the imaging vehicle, as most vehicles serve multiple purposes and are therefore equipped with a variety of sensors and research devices.

Images $\mathcal{I}_i \in [0, 255]^{M \times N \times 3}$ with height M and width N and three color channels can be ordered in *volumes* $\mathcal{V} = (\mathcal{I}_0, \dots, \mathcal{I}_\mathsf{I} - 1)$, where I is the number of images in the volume. A volume can be either a location series or a time series. In a location series, the camera is mounted on a moving imaging platform. Location series are chosen to get an overview of a larger area in the ocean. In a time series, a fixed camera position is chosen and the images show a fixed view over time. Time series are useful to monitor the impact of an event on a site of interest. Images for time series are usually taken by a so-called Fixed Underwater Observatory (FUO) (cmp. Figure 2 a)), which is a camera and lighting on top of a fixed platform. Usually, a sensor array is attached as well, recording values such as salinity, pressure or temperature. Sometimes, the camera can be moved, which leads to several time series [Godø, Johnsen, and Torkelsen, 2014; Osterloff, Nilssen, and Nattkemper, 2016].

In this thesis we mainly examine images from location series taken by unmanned vehicles that are either autonomous or remote-controlled. Although there are manned vehicles for image capture, they are rarely used owing to their complexity and cost.

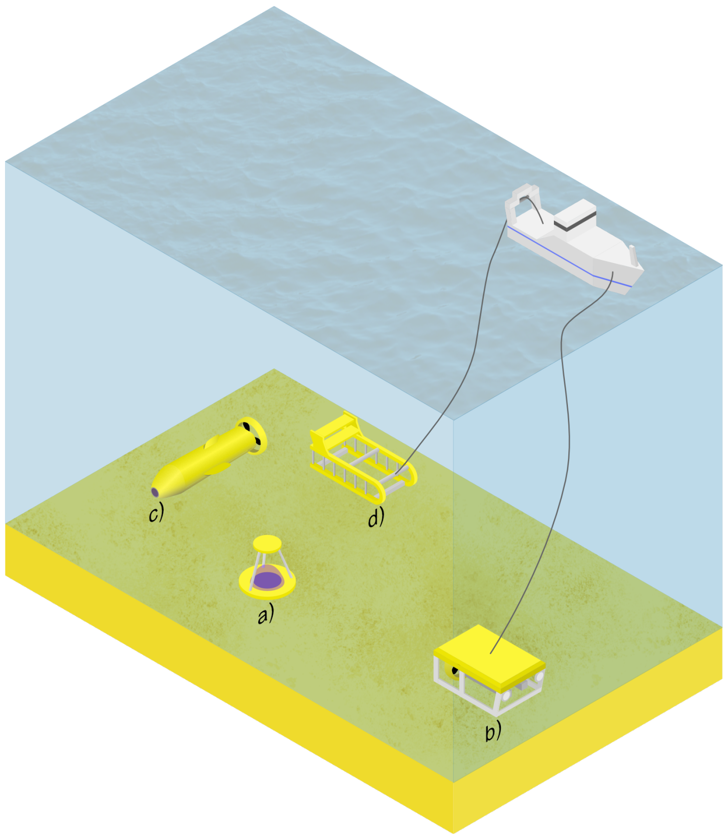


Figure 2: Imaging systems for marine research *a)* Fixed Underwater Observatories (FUOs) are used to monitor a certain position for a period of time, e.g. to monitor changes. *b)* Remotely operated vehicles (ROVs) are controlled from a research vessel. They have a propulsion system and can carry a variety of payloads, including sensor arrays, but also robotic arms. Therefore they are very flexible and can be operated close to the ground. *c)* Autonomous underwater vehicles (AUVs) are independent of a research vessel and usually follow a preprogrammed track. *d)* Towed platforms do not have a propulsion system, but are towed by the research vessel. They are simpler than the previous vehicles and also less flexible.

The simplest form is a towed imaging platform, i.e. a vehicle towed behind the research vessel (cmp. Figure 2 d)). The towed platform is used 5-7m above the seafloor at a speed of less than half a knot. The depth is controlled by the speed of the towing ship. A typical example for this class of underwater vehicles is TowCAM [Fornari, 2003]. Either videos or images can be recorded. Constant or flash lights can be used. Since the towed vehicle is maneuvered by the movement of the research vessel, a fine-grained control is not possible.

Remotely operated vehicles (ROVs) are tethered to the research vessel via a long tether (cmp. Figure 2 b)). This tether is used to power the vehicle, maneuver it and to have a live monitor for navigation and mission planning. This makes it possible to react to a given situation and also provides a fine-grained control. ROVs are able to be in operation for a long time (100h+) thanks to the tether. They are often used for maintenance and experiments that require interaction. ROVs can be equipped with a variety of sensors and devices. Therefore both, videos and images can be recorded. Additionally, they are equipped with other tools like cutters or a robotic arm. Size and depth restrictions vary greatly. The ROV Kaiko has even been deployed to the depth of the Mariana Trench at 10911m [Board, Council, et al., 2003].

Autonomous underwater vehicles (AUVs) are the most advanced vehicles for image acquisition (cmp. Figure 2 c)). They are independent of the research vessel, although they are usually deployed off one. AUVs are self-propelled. The operating time is usually between a few hours and several days. There are a few vehicles (e.g. the Autosub Long Range) with an operating time of up to six months. A course is pre-programmed to be followed using i) arrays of acoustic beacons on the seafloor (Long Base Line), or ii) a combination of Ultra Short Base Line acoustic communication, GPS positioning, and inertial navigation (if they are below the surface – based on dead reckoning using a combination of depth sensors, inertial sensors and Doppler velocity sensors) [Wynn et al., 2014]. This technology allows the operation in otherwise inaccessible areas, e.g. under ice sheets. In addition, the sound of the large propulsion engines of the research vessel is now eliminated. This allows for a somewhat quieter operation, which is necessary for the monitoring of certain wildlife. In contrast to towed vehicles, an almost stationary position can be maintained. Unfortunately the power output is limited due to the missing tether. Both, video and imaging systems are available. Flash-based lights are more common, owing to the limited power. For further information on AUVs see Wynn et al., 2014.

Manual annotation

Acquiring a lot of images is not a big problem anymore, but it does not immediately add value. For this purpose, images must be analyzed, i.e. objects must be located and named. We call this step, image annotation or for brevity annotation. Annotation is a combination of two cognitive tasks: 1. object detection and 2. classification, i.e. assignment of a label. Formally, we define an *annotation* $\mathcal{A} = (n, m, \mathcal{I}_i, \Omega)$ as a position (n, m) in an image $\mathcal{I}_i$ and a *class* Ω, i.e. a taxonomic/morphological label. Please be aware that in (marine) biology the term class refers to a taxonomic rank. In contrast to this, we generally refer to any label of an object as a class Ω in this thesis. Depending on the type of annotation such as point, circle, rectangle or polygon, extensions to this definition are necessary, which will be introduced as needed. The area that displays the object is also referred to as an *annotation patch* $\mathtt{I}$ in the following. Taxonomic categories are a set of standardized classes defined by experts, while morphological categories are classes based on a morphological, i.e. visual property of an object. These morphotypes can be useful if the object cannot be classified to the full extent. A typical example of a morphotype is the class "small round sponge". All classes can be either arranged in a flat structure, or in a hierarchical tree-like structure like e.g. the tree of life. The tree-like structure is called *label tree* in the following.

Annotation in marine science is a difficult task. Reasons for this are a) Many objects are unknown to humankind b) Even if already known there might be only a few known samples of a certain class known to a limited audience c) Many images show very few to no objects at all d) The image quality is often flawed due to poor lighting, focus problems or artifacts and e) The distance to the ground can be unfit for a reliable annotation of an object. Therefore, most of the annotations have to be created by experts. Even though experts are usually trained to perform the annotation task on the respective dataset, it is still a challenging task. The sole use of experts to annotate marine data leads to a bottleneck, not only because of the huge image datasets that can be captured today, but also because of the backlog of unannotated image datasets from the past. Automatic computational annotation seems to be the only solution. If a fully automatic annotation is not feasible, a semi-automatic computational annotation, i.e. the automation of parts of the annotation workflow, e.g. the classification or the detection, would be a huge benefit. Therefore, computational aid seems to be the only way to tackle this bottleneck problem and to gain knowledge from the vast amounts of image data of the past, present and future.

Annotation software In the past, image annotation could not be performed with modern web-based annotation software. In the beginning of marine image annotation, handwritten tables, listing the number of taxa or morphotypes in an image were used to describe the image content. Recording exact positions, such as today's pixel coordinates to locate an object, were laborious because a measurement with a ruler had to be performed. Storing this information in handwritten tables and using it was a challenging and error-prone task.

With the broad availability of computers and digital imaging, researchers used image manipulation software such as Adobe Photoshop to mark the position of objects in the image, e.g. by encircling them. Sometimes the image content was distorted by imprinting the circles in the image. Sometimes these circle markers were placed in additional layers (a virtual canvas placed on top of the image, which can be hidden). Handwritten tables were replaced by digital ones. However, this approach was hardly usable for a team of collaborating researchers.

The community faces the fact that the traditional approach of one observer, manually viewing and annotating all the image data, collected in one session with standard desktop software is no longer applicable in a growing number of projects. For huge annotation efforts, several experts with different areas of expertise are needed to accurately annotate the dataset. These efforts are only possible through web-based applications because often spatial gaps have to be bridged. In addition, quality control requires multiple experts. Otherwise, there is a risk of systematic errors, caused by a similar working environment, training or working methods. New methodological approaches are needed that focus on collaboration, interdisciplinary research and computational support [Langenkämper, Zurowietz, et al., 2017; Durden, Schoening, et al., 2016].

Modern web-based annotation software enables to permanently and consistently store the position and class of an object in a database. Experts from numerous venues can annotate images collaboratively and discuss certain objects in detail, which was not possible until now. Global positioning data such as GPS can be used to link the annotation data with a geoinformation system and thus provide a spatial context to multiple analyses. This web-based approach revolutionized the ability to efficiently and effectively generate knowledge from multiple sources, multiple experts and metadata. We have developed the Biigle 2.0 [Langenkämper, Zurowietz, et al., 2017] annotation software to improve the (I) scientific collaboration, (II) data fusion and integration and (III) standardization and annotation quality.

In many projects, the interpretation of an image collection may require a collaborative approach (I), so that the input of different experts with specialized domain knowledge, sometimes from different scientific disciplines, is required. To enable a collaborative analysis of marine image data, the annotation software must run in a web browser, independent of the operating system. It must provide ways to organize selected images into projects and share these images with other users, by inviting them to the projects. The possibility of making images accessible via the Internet and offering different options for assigning semantics to the images or image areas is already available by a small number of new tools such as CATAMI[2] (Collaborative and Automated Tools for Analysis of Marine Imagery [Althaus et al., 2015]), SQUIDLE[3], CoralNet[4], or BIIGLE (BioImage Indexing, Graphical Labeling and Exploration [Ontrup, Ehnert, et al., 2009]), the predecessor of Biigle 2.0. In addition to image data, users should also share and edit the same nomenclature or naming convention for the annotation task, i.e. use the same taxonomic categories. These can be downloaded from external sources (such as the World Register of Marine Species (WoRMS, [Horton et al., 2016] or other catalogs) or manually designed by users using a simple editor provided by Biigle 2.0.

To address the integrative analysis of the resulting annotation data (II), the tool must support fusion with data from other sources and sensors such as time, geo-coordinates, current, or chemical measurements. The tool re-

[2]`http://catami.org`

[3]`http://squidle.acfr.usyd.edu.au/`

[4]`http://coralnet.ucsd.edu/` O. Beijbom et al., 2015

quires customizable export functions so that users can export the annotations in different degrees of granularity. A low level of detail export would be one accumulative table of abundances for an entire volume. A high level of detail export would be a list with detailed descriptions of all annotations per image for all images in a volume. In addition, information from the images should be included in the export, such as metadata or pixel scale information from the images, to allow size measurements for each annotation $\mathcal{A}$ (for e.g. biomass assessment), or the whole image footprint to determine relative abundances per m^2. While metadata can either be extracted from the image file header or read from additional protocol files, the scale information sometimes has to be computed, based on laser point marker positions. These are identified manually by a user or an image processing algorithm in each image [Schoening, Kuhn, et al., 2015].

Quality control and quality measures (III) are an essential part of the annotation process. Unfortunately, this is often neglected due to time constraints or carelessness. Globally standardized annotation protocols such as those used in pathology do hardly exist for image annotation, since marine image analysis on this scale is a quite recent effort. An annotation protocol is a guide on how to annotate, including information such as zoom level and the label tree to use, duration of an annotation session, or a guide on how to screen the images. Although there is no standard, there are recommendations for the standardization of annotation efforts in Schoening, Osterloff, and Nattkemper, 2016. Image annotation software should provide means to enable standardization, such as linking to a written protocol, the ability to set a specific zoom level, or sharing of label trees.

In disciplines with a longer history of annotation, e.g. pathology, it is common for multiple experts to look at the same sample together. For marine biology this is often not the case as the number of samples to be viewed, is usually much higher and the cost of a misjudgement is lower, i.e. nobody is going to die. In the case of several observers, measures can be defined to evaluate their agreement. The inter-observer agreement is a measure of the extent to which the opinions of different experts agree. It is also a good indicator of whether an object can consistently be annotated. It might also indicate that systematic problems exist. This can be the case, for example, if only a few examples of the object are known, i.e. it is so rare that a thorough training of experts is not possible. This lack of training possibilities means that the mental model does not correspond to the "reality of life", i.e. the photographed object. Another quality measure would be the intra-observer agreement. It compares repeated annotation studies of one expert, i.e. one expert annotates the same set of images after a reasonable period of time again, e.g. after two weeks. This measure indicates whether annotations done by one expert are reproducible or not, i.e. an object annotated in one session but not found in the other sessions would lead to a lower intra-observer agreement. Errors such as these can be caused by a variety of influences, such as different zoom levels, mood, priming before annotation [Ware, 2012], e.g. viewing examples of different white anemones before annotation can increase the sensitivity to finding white anemones. Even the order of images might have effects. If an object a has not been found for a long time, it is more likely that a similar-looking object b is incorrectly marked as a, thus making the order of images in a volume an important factor.

In order to improve the quality of annotation, the annotation software should provide means to evaluate the above measures (III). In addition, tools are needed to improve the reproducibility and thus the accuracy of annotation. As already mentioned above, annotation is a combination of two cognitive tasks: 1. object detection and 2. classification. In earlier studies, it was observed that users seem to have individual strengths and weaknesses in these two sub-tasks [Culverhouse et al., 2003; Durden, Bett, et al., 2016]. For Biigle 2.0 we have therefore developed the Lawnmower Mode (LMM) as well as Largo and Volare. The LMM assists in object detection. Before entering the LMM, the user first selects the desired zoom level. The area of the image that is now visible is called a virtual tile of that image. The image is now divided into regular virtual tiles [e.g., 1)–12), Figure 3]. After activating the LMM, the system starts meandering (like

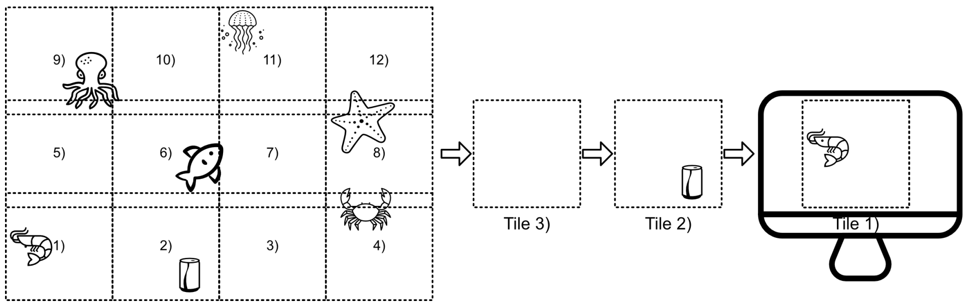


Figure 3: Lawnmower Mode The Lawnmower Mode (LMM) can be used to annotate an entire image while focusing on small tiles at a time. The users select the desired zoom level on which they want to annotate, and afterwards activate the LMM. The image is virtually cut into small tiles. The view will move to the lower left corner of the image, i.e. tile 1). By pressing the right arrow key the view will advance to tile 2), 3), until tile 12). After the final tile of an image has been displayed the view will proceed to tile 1) of the next image in the volume. Analogously, this works with the left arrow key to move backwards.

a lawnmower) through the image tile by tile [e.g., starting at tile 1) and moving to tile 2), 3), until 12), see Figure 3], controlled by the user who clicks on an arrow icon or uses the keyboard to switch to the next tile. The system starts in the lower left-hand corner of the image and ends in the upper right-hand corner. The tiles are computed to have minimal overlap to prevent screening the same image region multiple times.

For reviewing classification results, we developed the tools Largo and Volare.

A second review of the annotations without spatial context information, i.e. only the annotation patches are displayed, is useful to compensate for a bias caused by the spatial context. That's why we introduced Largo. It displays all annotations, annotated as belonging to a single class to spot odd ones out (see Figure 4). This process of screening the annotations is fast and reliable. This is under

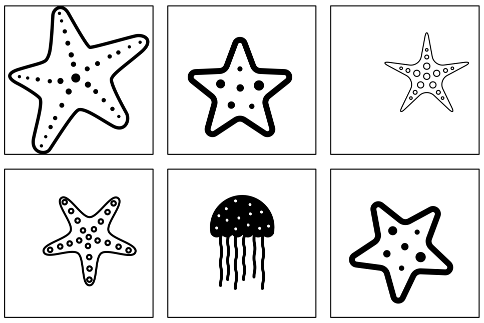


Figure 4: Largo Largo shows all objects, which were annotated as belonging to one class. Therefore it allows to spot the odd one out quickly.

the assumption that the classes are homogeneous in their visual appearance. In contrast, it can be argued that contextual information is an essential part of recognition and that biases caused by false annotations are negligible or can be corrected. Therefore we created Volare. It initially zooms in on the first annotation. With the arrow keys (or the forward/backward button in the web interface) the user can pan the view to display the next/previous annotation. Annotators can still zoom in/out, pan the view or manipulate the image to view context information.

With modern annotation tools such as Biigle 2.0, the ability to effectively and efficiently divide manual annotation into detection and classification leads to different possible annotation study designs. A number of the most common ones are listed below: **A) The expert** One annotator, who performs the entire annotation task alone, i.e. detection and classification is performed by one person. **B) Group Annotation** Detection and classification is done by a group of people collaboratively, while no specific task is assigned to specific persons. **C) Area Expert Annotators** A group or possibly an individual is detecting the objects and an expert or group of experts of the geographic area classify the previously found objects. **D) Species Expert Annotators** Like C), but the classification task is too difficult to be executed by experts with a more general knowledge of the classes present in that area, so that instead experts of a particular taxon group are needed. Therefore the detections might be classified roughly, i.e. a class of a higher taxonomic group might be attached to each object and the fine-grained taxonomic classification is left to the experts. E) **Citizen Science** A group of untrained citizens is assigned with the task of detecting objects. The task can be to either focus on a number of objects or generally label everything that might be of interest. The classification will then be left to an expert/experts. F) **Citizen Science Classification** Citizen scientists could be used to classify objects that have already been detected. This is usually done by screening the detections for one class and only deciding whether that class is depicted in the detection.

1.2 Automatic annotation

As already pointed out annotating datasets is of crucial importance for marine imaging. Owing to the limitation of manual labor automatic annotation would be very handy to have. Automatic annotation like manual annotation can be split into two subtasks, i.e. 1. object detection and 2. classification. These two tasks can be addressed individually and a variety of possible approaches to them exists.

Automatic classification Automatic classification of images or annotation patches has a rich history. Although a thorough description of all methods is out of focus of this thesis and left to a textbook in computer vision or machine learning a few methods will be mentioned in the following. The so-called (naive) Bayes classifier uses probabilistic methods to classify objects [Ponce et al., 2011]. It is named after the Bayes' theorem by Reverend Thomas Bayes that is known since the 18th century. More information can be found in Hand and Yu, 2001. K-Nearest Neighbor uses the assumption that data within a close distance tend to be of the same class. It was presented in 1967. More information can be found in the original paper [Cover and Hart, 1967]. In the following, Neural Networks such as the Multi-Layer-Perceptron (MLP) got popular. Neural Networks try to reenact mechanisms of the brain, to classify objects. The Perceptron, one part of the MLP was presented in 1958 [Rosenblatt, 1958]. A technique called backpropagation learning made the MLP famous in the 1980s [Rumelhart, G. E. Hinton, and R. J. Williams, 1985]. The Support Vector Machine (SVM) presented in 1995 [Cortes and Vapnik, 1995] rose quickly in popularity quickly, owing to the good performance. It uses a so-called decision function to decide to which of two classes a data point should belong. Although being a binary classifier it can be extended to multiple classes by combining multiple individual SVMs [Kressel, 2002]. Further groups of methods include decision trees [Breiman et al., 1984], ensemble learning [Sagi and Rokach, 2018], active learning [Settles, 2012] or extreme learning machines [G.-b. Huang, Zhu, and Siew, 2006]. Most of these methods are quite limited in a few aspects. Some are designed for binary classification but can be extended to multi-class, e.g. SVM. However, this limits the number of classes, as the combination of individual classifiers is computationally expensive and the performance is likely to decrease. Classifiers designed in the early days of computers and classification were designed for a limited amount of data, due to limitations of the hardware. Huge datasets were not existent at that time and therefore the runtime increased dramatically with increasing dataset size.

The most prominent development has been the advent of deep learning and especially convolutional neural networks (CNNs). Many deep learning algorithms have been proposed in recent years mostly targeting everyday images. Everyday images are images depicting everyday objects such as cars, houses, streets, or trees. Please note that in machine learning, especially in deep learning, everyday images are often referred to as natural images, but due to the possibility of misunderstandings in interdisciplinary research we will use the term everyday image. Preliminary works of deep neural networks, e.g. LeNet5 [LeCun, Bottou, et al., 1998] reach back to 1998. Unfortunately, due to lack of data and computing power bigger networks were not considered. Image classification using modern deep neural networks had its advent with the ImageNet classification challenge in 2012 and the overwhelming results of the winner AlexNet [Krizhevsky, Sutskever, and G. E. Hinton, 2012]. Since then every year new

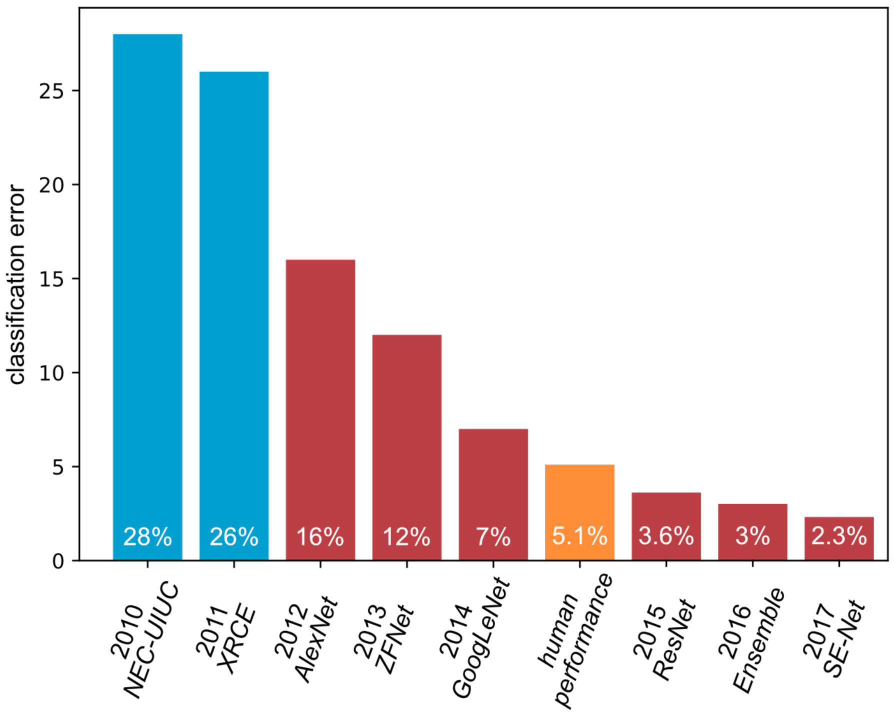


Figure 5: Winners of the ImageNet LSVRC classification contest [E. Park, 2017; K. He, X. Zhang, et al., 2015; Labs, 2019; Tsang, 2019]. The bars show the top 5 classification errors. All networks since 2012 (red) are convolutional neural networks. The orange bar depicts the human performance on the same task.

architectures advancing the results even beyond human performance [K. He, X. Zhang, et al., 2015] have been proposed (see Figure 5). ZFNet [Zeiler and Fergus, 2014] improved AlexNet by optimizing kernel sizes and stride size. VG-GNet [Simonyan and Zisserman, 2014] uses only 3×3 convolutions and is deeper than previous networks. GoogLeNet [Szegedy, Liu, et al., 2015] introduced the Inception module – a concatenation of various convolutions and poolings of the same input. ResNet [K. He, X. Zhang, et al., 2016] introduced residual connections between layers. Inception-V4 [Szegedy, Ioffe, et al., 2017] improves on previous Inception networks by restructuring the architecture and introducing even more Inception modules. Squeeze-and-Excitation Networks [J. Hu, Shen, and Sun, 2017] introduce the squeeze and excitation module, which weights the importance of a feature map. Further networks worth mentioning are Xception [Chollet, 2017], which modifies the Inception-V3 architecture using depthwise separable convolutions, InceptionResNet [Szegedy, Ioffe, et al., 2017] that introduces residual connections to the Inception-V4 architecture, MobileNet [Howard et al., 2017], which is a computationally lightweight network obtained by cleverly factorizing convolutions and DenseNet [G. Huang et al., 2017], which is a very deep network using 1×1 convolutions in an intelligent fashion. Increasing performance of the networks is owed to increasing depth in networks or deep learning techniques, e.g. batch normalization [Ioffe and Szegedy, 2015], dropout [N. Srivastava et al., 2014], inception blocks [Szegedy, Liu, et al., 2015], residual blocks [K. He, X. Zhang, et al., 2016], highway blocks [R. K. Srivastava, Greff, and Schmidhuber, 2015]. Some of these, e.g. the residual or highway blocks, are a means to be able to train even deeper networks, which would usually have been impossible owing to gradient optimization problems. A more thorough introduction of most of the above networks and deep learning techniques is presented in Section 4.1. For an in-depth analysis of deep neural networks for practical applications have a look at [Canziani, Paszke, and Culurciello, 2016]. For a review about deep learning in general please have a look at Schmidhuber, 2015.

Digression: ImageNet

ImageNet is a huge annotated image database accompanied with an accumulation of machine learning challenges. The most prominent one is the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) - Image Classification task featuring $1.2*10^6$ training annotation patches annotated with one of 1000 classes. For examples have a look at Figure 6. Additionally, $5*10^4$ optimization and 10^5 validation images are provided. For each class around 1000 samples are provided. The images are everyday images extracted mostly from Flickr and other image search engines [Russakovsky et al., 2015]. ImageNet has been a major driving force for innovation in deep learning and computer vision research in recent years. A predecessor to ILSVRC image classification challenge is the Pascal VOC Challenge, starting in 2005. Twenty classes divided into four categories (Vehicles, Household, Animals and Other) are given with $3*10^4$ annotation patches for training/optimization. The validation dataset is not publicly available but reachable through an evaluation server [Everingham et al., 2015].

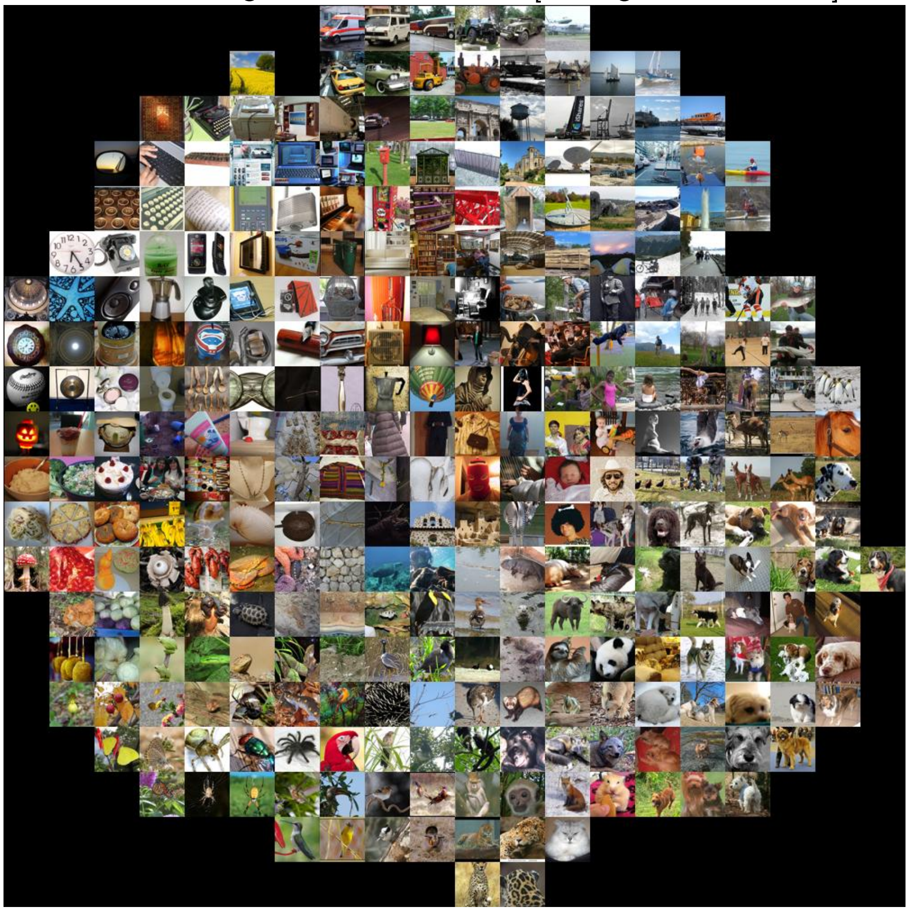


Figure 6: Sample annotation patches of the ImageNet dataset. Images courtesy of Andrej Karpathy [Karpathy, 2018].

Marine image informatics: Classification Marine image informatics is an emerging field settled at the intersection of marine biology and ecology, image processing, and machine learning. Owing to the availability of digital camera systems and underwater imaging platforms like Remotely Operated Vehicles (ROVs), Autonomous Underwater Vehicles (AUVs) and towed sleds, large image collections (approx. $10^3 - 10^4$ images) can be acquired during a single dive [Monk et al., 2018]. While the worldwide volume of available marine image data is huge and continuously growing, the application of automatic classification for marine image analysis is by far not straightforward because of the following reasons:

a There is usually not enough training data, because image annotation in marine sciences is non-trivial and laborious. Morphological/taxonomic classification is an n-class problem and requires a considerable amount of education, training and experience. Unlike in everyday images, where almost all objects can be recognized by almost everyone growing up in a modern civilization, marine objects of interest, such as fish, sea cucumbers, starfish, can be hard to detect and classify even for human experts. Therefore, image annotation performed by experts is time-consuming, error-prone and expensive, particularly as there are a limited number of experts available [Schoening, Osterloff, and Nattkemper, 2016].

b The automatic annotation task is further complicated by the high diversity and low abundance of marine life in the deep-sea [Jones et al., 2017]. A large number of classes are represented by a low number of examples. Moreover, it is a common observation that 80%-90% of the data belong to a small subset of L' classes among the total number of L observed classes, with $L' << L$. This observation is called the data imbalance problem in machine learning classification tasks (see Section 4.4).

c The images are often of mixed quality, with some images featuring high signal-to-noise-ratio, extreme light exposure, cast shadows and an insufficient pixel resolution for objects of interest, i.e. the objects are too small. This leads to problems such as concept drift (see Section 4.5). ROVs, AUVs, and towed sleds are rarely used just for the purpose of image acquisition. These platforms often carry other payloads and a variety of sensor arrays serving different purposes. Even if only images are acquired, quite often there are multiple objectives, such as biodiversity studies, mapping the seafloor, resource assessment and habitat mapping calling for different, often contradicting modes of operation. Therefore, the speed of the platform and the distance to the seafloor might be unfit for the image classification. In contrast to everyday images, where objects usually take up 30%-70% of the image pixels, the pixel resolution of the object and image quality may not be ideal for classification. In many scenarios, the images are recorded below 200m depth where there is no natural light.

Thus a flash has to be used, which might lead to lighting artifacts, such as vignetting or illuminating marine snow.

Depending on the algorithms used for classification, some of the problems mentioned above are more severe than others. Especially deep learning algorithms need a large number of training samples, whereas conventional or shallow learners such as the SVM or k-NN perform well even with a smaller number of training data. Besides, pre-processing methods can be applied to compensate for some of the problems, e.g. image filtering to compensate for bad image quality, or creating artificial data to compensate for data scarcity and the class imbalance problem.

Although problems exist, a number of machine learning approaches to automatic marine image classification have been published. Marcos, Soriano, and Saloma, 2005 propose a feedforward backpropagation neural network to distinguish between three classes in coral reef imagery. Marcos, David, et al., 2008 use Linear Discriminant Analysis to classify data of a shallow-water reef. In Stokes and Deane, 2009, the authors propose a method computing the distance of an unknown annotation patch to a library of already labeled annotation patches to classify images depicting coral reefs. Denuelle and Dunbabin, 2010 propose a similar approach using annotation patch libraries and the Mahalanobis metric to compute the distance of an unknown annotation patch to the already known library patches, to classify kelp. In Oscar Beijbom, Edmunds, Kline, et al., 2012 and Oscar Beijbom, Edmunds, Roelfsema, et al., 2015, the authors present SVM-based-classifier systems for benthic coral images applied to datasets of Pacific coral reefs. Shihavuddin et al., 2013 investigate the use of different features with different classifier systems – K-Nearest Neighbor, neural networks, SVMs and probability density weighted mean distance to classify coral reef imagery. Bewley et al. extend their SVM-based system [Bewley, Douillard, et al., 2012] to a hierarchical multi-class system to classify or rather detect kelp in Bewley, Nourani-Vatani, et al., 2015. They use a hierarchy of binary logistic regression classifiers to do a hierarchical classification with 19 classes on AUV dive images recorded in the Tasman Sea. Oscar Beijbom, Treibitz, et al., 2016 use support vector machines on fluorescense images to improve classification performance. Mahmood et al., 2016 use convolutional neural network features in combination with hand-crafted features to classify the Morea Labeled Coral dataset. Rimavicius and Gelzinis, 2017 compare different deep learning methods, i.e. convolutional neural networks, deep belief networks and deep neural networks on marine datasets classifying five different classes. Siddiqui et al., 2017 propose a deep neural network feature extractor in combination with an SVM for a 17 class fine-grained fish classification with 94% accuracy. Gómez-Ríos et al., 2019 use the convolutional neural network ResNet to classify coral texture images. In Allken et al., 2018 the authors propose the generation of synthetic images to augment the training data to achieve a 94% accuracy on a three class fish classification task using the Inception V3 deep learning network.

Most of these papers focus on benthic shallow-water coral reef imagery or on non-benthic plankton or fish classification. This thesis focuses on deep-water benthic mega- and macrofauna classification. To the best of our knowledge the only related works are published by Schoening, Bergmann, et al., 2012, Marburg and Bigham, 2016 and Lu et al., 2018. In Schoening, Bergmann, et al., 2012 the authors propose a cascade of binary SVM classifiers to detect and classify megafauna in deep-sea Arctic imagery. In Marburg and Bigham, 2016 the authors use Nvidia DIGITS and tensorflow to train convolutional neural networks (AlexNet and LeNet) to either classify annotation patches to either background or non-background, or ten different fauna classes, or the combination of both, i.e. either background or one of the ten fauna classes. Images are acquired using an ROV at the Pacific continental shelf. Lu et al., 2018 propose the FDCNet CNN to classify images from the Kyutech10K image database into one of seven categories with 92

Automatic object detection Usually, the goal is not only to do automatic classification but also to perform object detection. Although this thesis focuses on automatic classification, object detection was also investigated and will be briefly discussed. We limit the focus to current developments for brevity. For a more detailed overview have a look at [Cyganek, 2013; Zhao et al., 2019].

Deep learning-based object detection methods had their advent with region-based convolutional neural networks (R-CNN) [Girshick et al., 2016; Uijlings et al., 2013] and its extensions Fast R-CNN [Girshick, 2015] and Faster R-CNN [Ren et al., 2015]. A further extension is Mask R-CNN [K. He, Gkioxari, et al., 2017], which additionally provides instance segmentation, i.e. the object is not only detected by providing a bounding box but rather a pixel-accurate mask is computed. The You Only Look Once (YOLO) [Redmon, Divvala, et al., 2016] and Fast YOLO networks are high-speed real-time enabled alternatives to the R-CNN algorithms. YOLO v2 and YOLO-9000 are presented in a follow-up paper improving the detection performance [Redmon and Farhadi, 2017].

An overview of object detection in the marine imaging context is provided by Moniruzzaman et al., 2017. The authors divide the approaches into one of three domains – fish detection [Li et al., 2015; Villon et al., 2016], plankton detection [Lee, M. Park, and Kim, 2016; Dai et al., 2016] and coral detection [Marcos, Soriano, and Saloma, 2005; Elawady, 2015; Mahmood et al., 2016].

The only solutions to marine benthic image object detection known to the authors are Cline et al., 2009; Schoening, Bergmann, et al., 2012, although they are quite limited in terms of number of classes or detection performance. In 2018 we proposed a new deep learning-based approach using Mask R-CNN and autoencoder networks to tackle object detection in marine image informatics [Zurowietz, Langenkämper, Simon-Lledó, et al., 2018]. This approach is described in detail in a later chapter.

Digression: COCO

The equivalent of the Imagenet ILSVRC image classification challenge for object detection is the Common Objects in Context (COCO) dataset[a]. It comprises $8 * 10^5$ objects on 10^5 images divided into 80 classes. The ground truth for object detection is given as a mask on the original image, e.g. Figure 7. The images are extracted from Flickr. There are also COCO challenges on object and semantic segmentation and captioning [T.-Y. Lin et al., 2014].

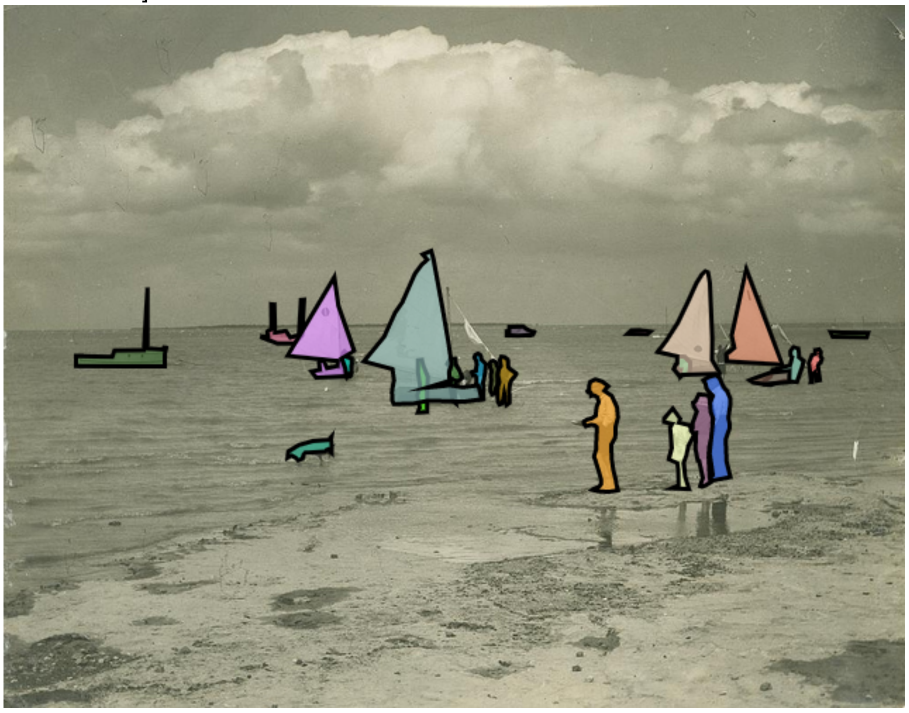


Figure 7: Sample image mask of the COCO dataset Sample image of the COCO dataset with the detection masks overlaid [COCO, 2019].

[a]Although, Imagenet also provides a detection challenge, nowadays.

1.3 Context of this work and research questions

This thesis deals with the topic of marine benthic classification of macro- and megafauna. Although huge leaps were made towards automatic classification in computer vision, these were proposed for everyday objects. It is unclear whether this knowledge can be transferred to domains, such as marine images. The annotation process is a complex task that is limited to experts, as opposed to everyday images where almost everyone can perform the task. The complexity makes this task error-prone. Besides, data acquisition in the marine sciences is tedious and costly, compared to everyday images. This limits the ability of generating additional data quickly if required. All of this hampers the ability of deep learning classifiers to work reliably. Therefore in addition to modern deep learners, shallow learners have to be considered, as well. This thesis analyzes whether automatic classification of marine macro- and megafauna is possible up to a reasonable performance, nowadays. Such an automation is urgently needed to relieve the small number of experts and thus create value from the vast quantities of images of the past, present and future. Further detailed questions are, how and with which algorithmic means this automation can be achieved best. It must also be checked whether this depends on the dataset. If so, guidelines should be developed for the selection of appropriate methods. In addition, we have a look at typical problems of marine imagery like concept drift, inaccurate annotations possibly from citizen scientists, or incorporating size information. If a classification of the marine macro- and megafauna with sufficient accuracy is possible, the question remains, which conditions must be fulfilled, so that an accurate classification is possible.

If an entirely automatic classification is not possible, suitable solutions should be developed, in order to make e.g. semi-automatic or assistive classification possible. Here a special focus is on real-time assistive solutions. Besides, not only classifications should be considered, but the annotation process as a whole. For this purpose, suitable interfaces and possibilities for interaction through the use of synergy effects with detection solutions must be kept in mind. Furthermore, own solutions for detection are to be offered. Since this is an applied work, it is about real, applicable solutions, i.e. solutions, which are not feasible because of their runtime, will not be presented in detail.

2

Data

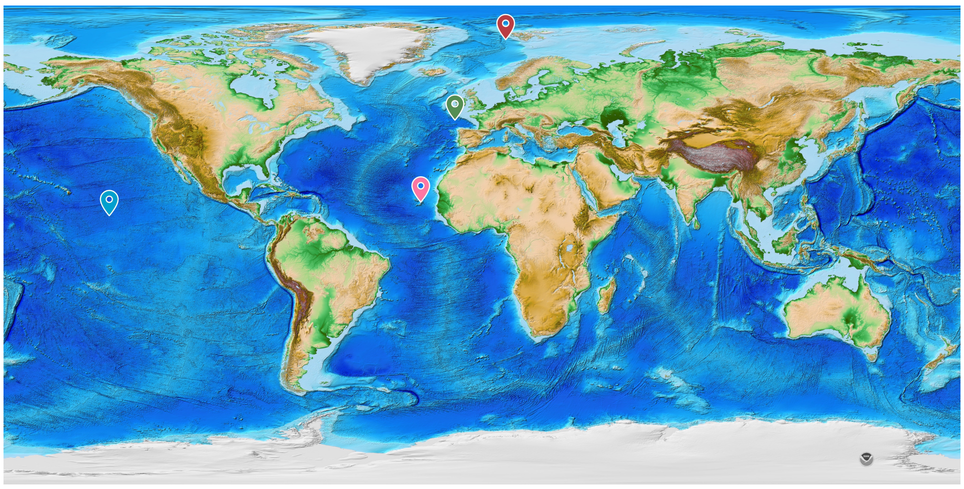


Figure 8: Location of the datasets used in this thesis Red: Hausgarten, Green: PAP, Pink: UVP5, Blue: APEI6

Several marine datasets were used in this thesis to evaluate methods and algorithms. The datasets are quite diverse and were captured at different locations (see Figure 8), using different equipment, i.e. camera type, lighting setup, distance to the seafloor and thus object size in the image (see Figure 9). The type of imaging platform, e.g. ROV, AUV or towed platform may be different. Also, different researchers, with different experience in marine taxonomic annotation, annotated these. All except one dataset show photos of the seafloor. The characteristics and imaging setup of the individual datasets are presented below. The key data characteristics for each dataset are also listed in Table 1.

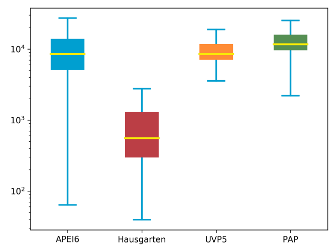


Figure 9: Annotation area in pixels per dataset The size of an annotation varies with factors such as image resolution, distance to ground, or object size. The size of an object is one important factor for machine learning algorithms to yield good results. Please note that outliers have been omitted and the y-axis has a logarithmic scaling to better display the data.

name	#Images annotated/total	#Annotations	#classes	Image size	Object size min/max/mean[5]
Hausgarten PS80/179-3	488/1042	1815	9	5616×3744	6/140/29px
PAP	3708/12116	31227	19	ca. 2400×18000	36/1726/121px
APEI6	9199/10052	54938	10	2448×2048	4/1224/98px
UVP5	42318/42318	42318	33	$1,280 \times 1,024$	37/2556/112px

Table 1: Key data for the datasets used in this thesis.

2.1 Hausgarten PS80/179-3

The Arctic long-term ecological research (LTER) deep-sea observatory Hausgarten station IV (at about 2500m depth) [Soltwedel et al., 2016] located between Svalbard and Greenland in the North Greenland Sea (see Figure 8) is used for investigations on interannual changes in megafaunal communities. The images, kindly provided by the Alfred Wegener Institute, were captured with a towed platform called OFOS (Ocean Floor Observation System[6]). It

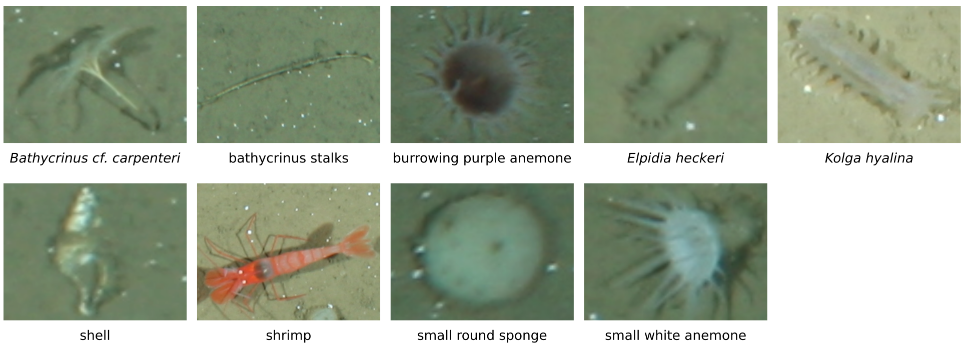


Figure 10: Classes found in the Hausgarten dataset Example annotation patches for the classes found in the Hausgarten dataset.

is equipped with a Canon EOS-1Ds Mark III digital single lens reflex camera modified for marine use, one flash, four LED lights and three laser pointers for measurement of the footprint of an image. We analyzed photos taken along profile PS80/179-3 [Bergmann, Schewe, and Soltwedel, 2017] in 2012, which were labeled by experts, resulting in 1815 labels divided into nine classes (see Figure 10). The most common classes are small white anemone and small round sponge, which make up more than half of the dataset. With more than 100 annotations, *Kolga hyalina*, *Elpidia heckeri*, and *Bathycrinus cf. carpenteri* are still quite abundant. Shells, burrowing purple anemones, bathycrinus stalks and shrimps account for about 15% of the dataset (cmp. Figure 11). For a first exploration of the dataset, dimensionality reduction can be used to visualize the data distribution of classes. Therefore, MPEG-7 features of the annotation patches were computed (for further information on MPEG-7 features see

[6]https://www.awi.de/forschung/biowissenschaften/tiefsee-oekologie-und-technologie/technologie/fotovideosysteme.html

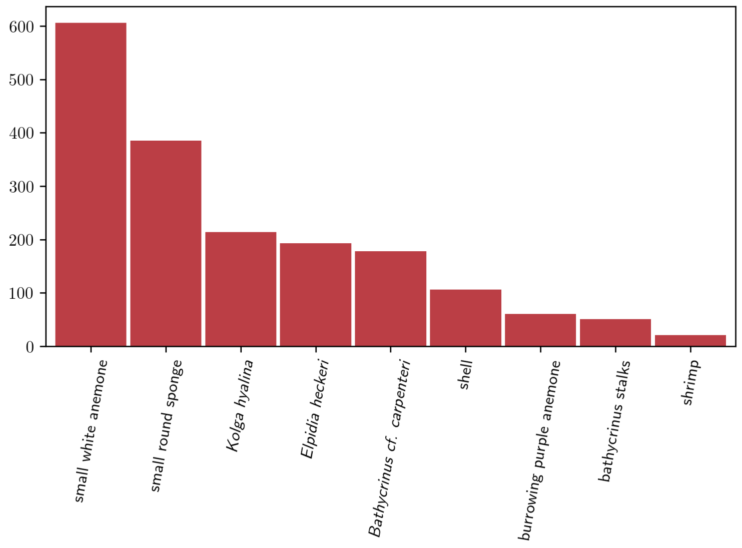


Figure 11: Distribution of classes found in the Hausgarten dataset Each bar shows the abundance of a class in the Hausgarten dataset.

Section 3.2). These features are high-dimensional but can be projected to 2D using dimensionality reduction. For this purpose we used t-SNE [Maaten and G. Hinton, 2008], which is known to provide a good embedding. The scatter plot of the 2D projected feature vectors is shown in Figure 12. Some clusters can be detected, i.e. areas with high data density. Large parts of *Bathycrinus cf. carpenteri* form a cluster, as do burrowing purple anemones and shrimps. A part of the *Kolga hyalina* annotations form a cluster (Figure 12 lower left), mixing with *Elpidia heckeri*. This is no wonder, as both are morphologically similar and are kindred at the taxonomic level of family. Nevertheless, there is also part of the plot (Figure 12 lower middle), where *Elpidia heckeri*, *Kolga hyalina*, small round sponge and small white anemone overlap. Small round sponge and small white anemone form clusters, but overlap in most parts of the plot, although some spots with an occurrence of only one of them exist. This was also to be expected, as both are similar in morphology. Although some classes seem to be quite easy to be automatically classified, many classes have overlaps and can be confused quite easily. For the sake of brevity, we refer to this dataset as Hausgarten in the following text.

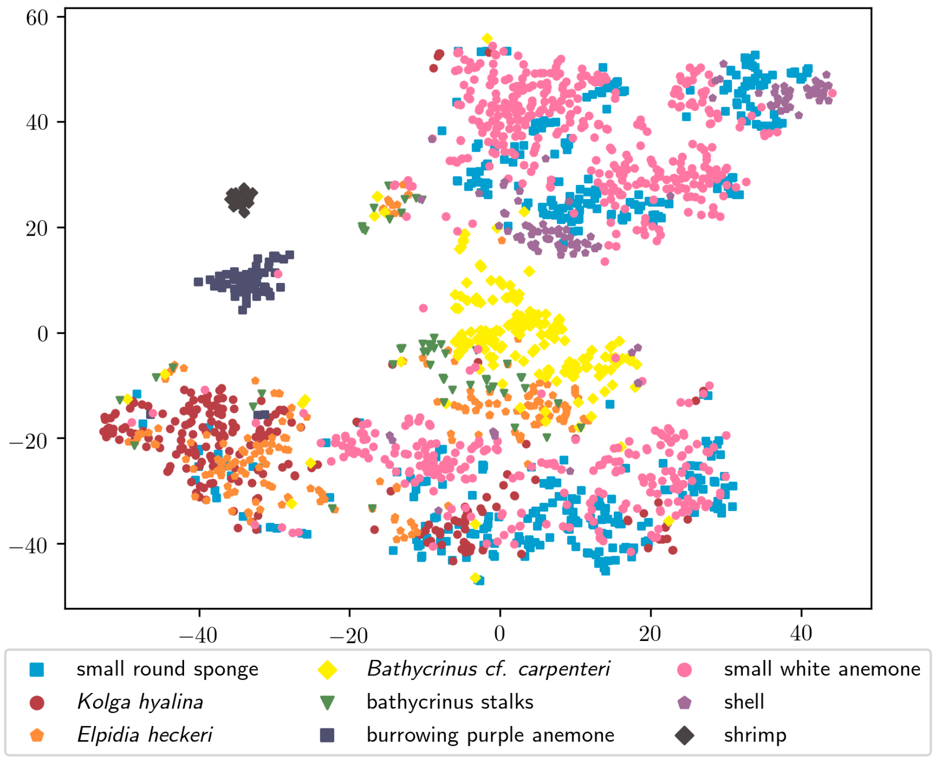


Figure 12: Hausgarten data distribution t-SNE is applied to reduce the dimensionality of the Hausgarten dataset to 2D. Each point is one annotation.

2.2 PAP

The Porcupine Abyssal Plane is located in the northwest Atlantic, southwest of the United Kingdom in international waters (see Figure 8, [Morris et al., 2014]). With the AUV Autosub6000 [McPhail, 2009] about 60000 images with a resolution of 2448x2048 pixels were taken. These were pre-processed by a third party, as described in Morris et al., 2014 to obtain 12116 mosaic images, i.e. large images created by stitching together individually captured images using information from overlapping areas of these images. These mosaics were kindly provided by the NOC. The images were taken at a depth of 4600-4900m with a target distance to the seabed of 3.2m. No laser pointers were equipped, but from visual inspection, the distance to the seafloor seems to be quite constant. A flash was used for illumination. The original dataset consists of 33354 annotations, divided into 72 classes. Unfortunately, many of these classes contain only a few (< 10) annotations. For this reason, we have compiled a new dataset of the nineteen most abundant classes (see Figure 13) consisting of 31227 annotations (93.62% of the original dataset, see Figure 14). The four largest classes, *Ophiuroidea* (9200 annotations), cnidaria2 (7575), *Amperima* (5702) and *Foraminifera* (2632) account for 80.4% of all annotations in the dataset. Another four classes have more than 500 annotations. The rest of the classes only make up about 10 percent of the dataset. These classes have more than 150 samples but less than 500. Similar to the dimensionality reduction above (see Section 2.1 Hausgarten), a scatter plot was created for this dataset (cmp. Figure 15). Unlike above, local binary patterns (see Section 3.2) were used as features and the maximum number of annotations per class was limited to 500 by random sampling each class. In addition, only the ten most common classes were used. This was done to facilitate the visualization of the data. In Figure 15 we can see that cnidaria 16 forms a distinctive cluster. All further classes are mixed. Nevertheless, there are areas with predominantly one class. This indicates that automatic classification is not a trivial task for this dataset.

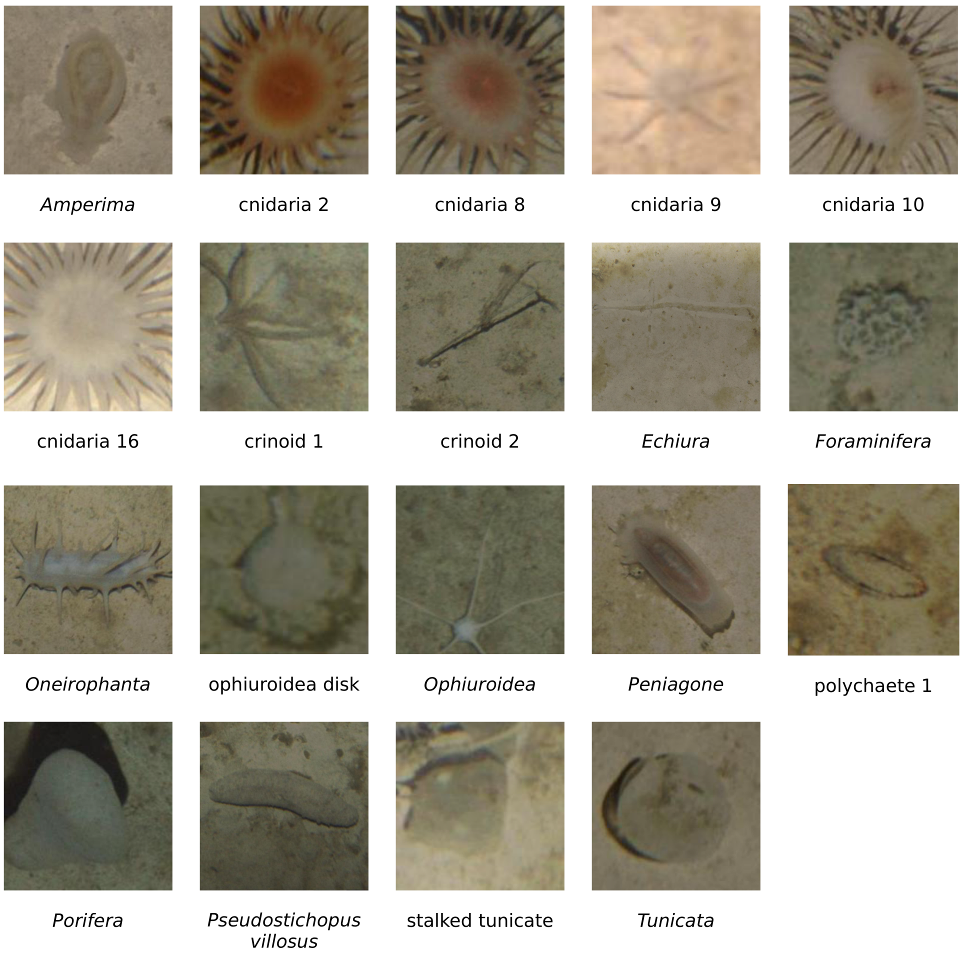


Figure 13: Classes found in the PAP dataset Example annotation patches for the classes found in the PAP dataset.

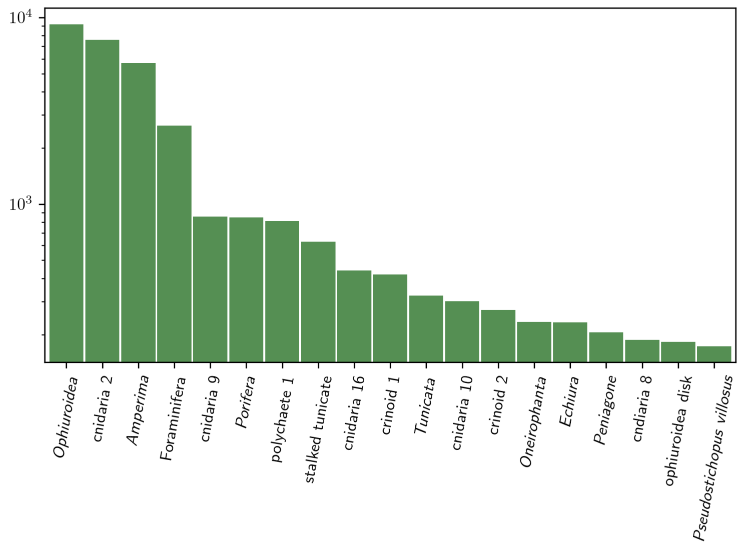


Figure 14: Distribution of classes found in the PAP dataset Each bar shows the abundance of a class in the PAP dataset.

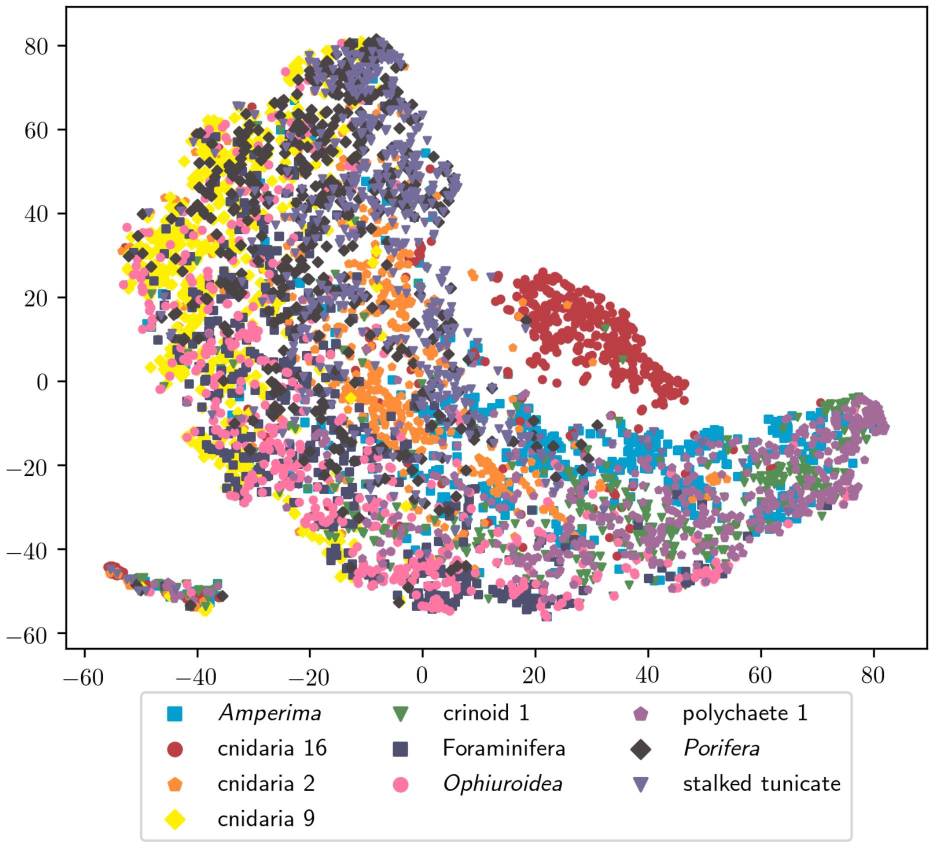


Figure 15: PAP data distribution t-SNE is applied to reduce the dimensionality of the PAP dataset to 2D. Each point is one annotation.

2.3 APEI6

The Area of Particular Environmental Interest 6 (APEI6) is a Pacific region centered on 122°55' W, 17°16' N (see Figure 8). It is a marine protected area designed to safeguard and monitor biodiversity and ecosystem function in an abyssal Pacific region (the Clarion Clipperton Zone) targeted for deep-sea nodule mining [Wedding et al., 2013]. The dataset contains 10052 images with a size of 2448×2048 pixels. The images were taken with a digital camera mounted on the Autosub 6000 AUV [Morris et al., 2014] at a depth between 4013m and 4235m and with a distance to the seabed between 2m and 4m. Ten different classes are annotated by experts, with a total of 54938 annotations (see Figure 16 for examples). *Protista* is the majority class with 85.8% of

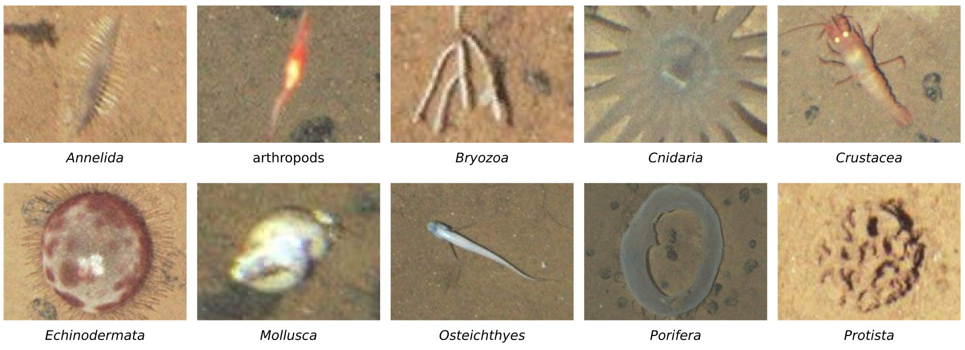


Figure 16: Classes found in the APEI6 dataset Example annotation patches for the classes found in the APEI6 dataset. *Mollusca*, *Echinodermata*, and also *Cnidaria* have a wide variety of shapes, but only one example image per label is shown for brevity.

the annotations belonging to it (cmp. Figure 17). *Cnidaria* and *Porifera* each feature more than 1000 annotations. All other classes only make up about 6% of the dataset. All classes are taxonomic classes except arthropods, which is a morphotype. It features a variety of heterogeneous objects annotated on a coarse level, as they cannot be labeled on a finer level due to the image quality. Just as above (see Section 2.2 PAP) a dimensionality reduction was performed using local binary patterns. The number of annotations per class was limited to 500 by random subsampling (see Figure 18) to achieve a better visualization. There are no visible clusters featuring only one class. Most classes are located

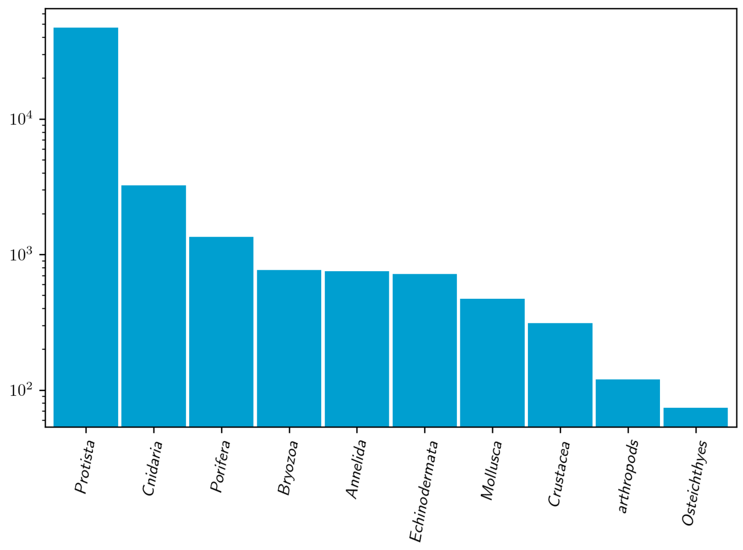


Figure 17: Distribution of classes found in the APEI6 dataset Each bar shows the abundance of a class in the APEI6 dataset.

in all parts of the plot. Nevertheless, classes such as *Porifera* are only found in a certain part of the plot. This indicates that the automatic classification of objects in this dataset may not be easy.

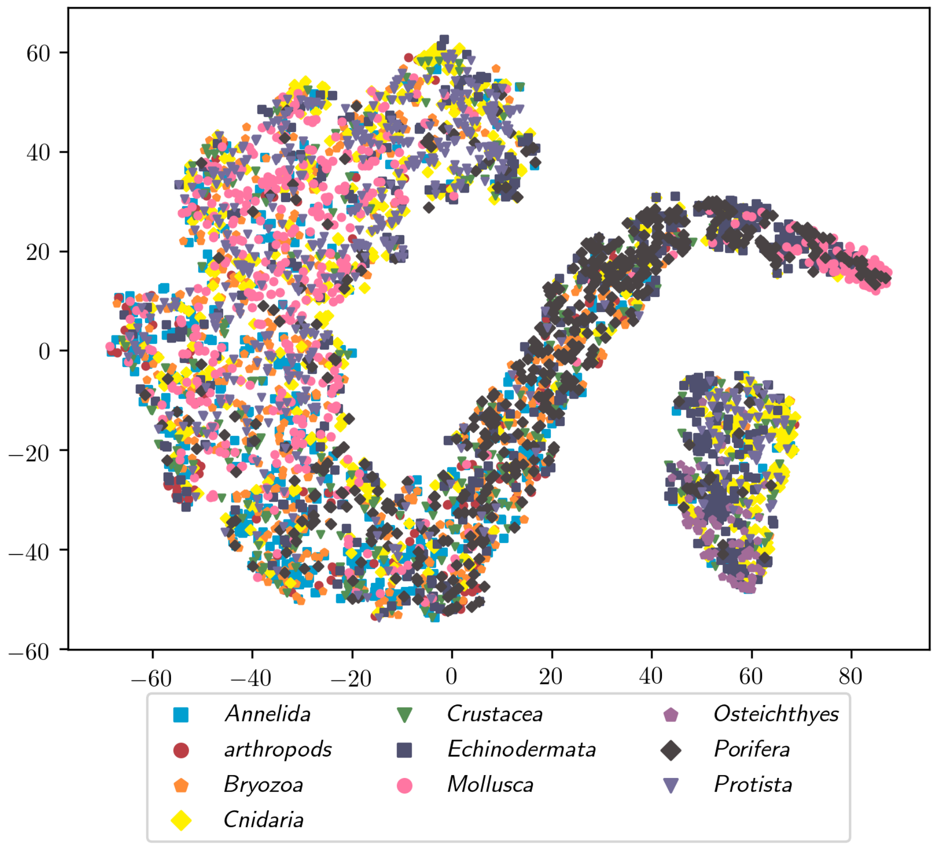


Figure 18: APEI6 data distribution t-SNE is applied to reduce the dimensionality of the APEI6 dataset to 2D. Each point is one annotation.

2.4 UVP5

The original dataset, provided by Rainer Kiko of GEOMAR, consists of 42318 plankton images taken with the Underwater Vision Profiler 5 (UVP5, [Picheral et al., 2010]). The images are recorded with a Sony XCI SX1 CCD B&W camera with a resolution of 1280×1024 pixels. The illumination is provided by collimated red LEDs at 625nm. The images are in grayscale on a white background. In contrast to the other datasets, no screening of the image for objects of interest is necessary, but the image recorded by the camera is post-processed onboard the UVP5 to display only the isolated object that is additionally rescaled [S.M. Schröder, personal communication June 18, 2019]. Therefore, the number of images and annotations is the same. In addition, the images show plankton from

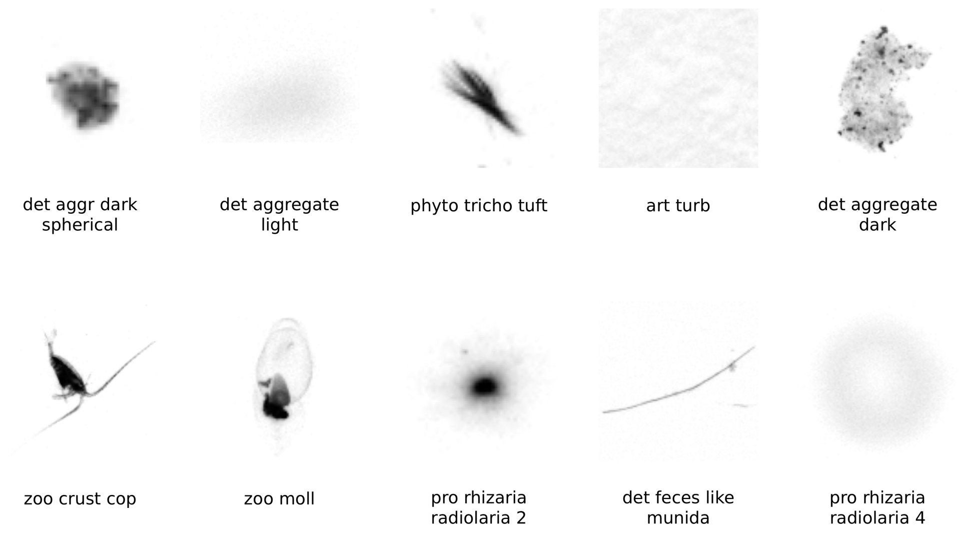


Figure 19: Classes found in the UVP5 dataset Example annotation patches for the classes found in the UVP5 dataset.

the water column, unlike all other datasets that show benthic images. Plankton captured with the UVP5 can be much smaller than mega and macrofauna targets in the other datasets, with sizes down to a fraction of a µm. Although this thesis focuses on benthic images of mega- and macrofauna, this dataset is used to analyze whether methods presented in this thesis are valid for a broader range of applications. The annotations of the original dataset can be divided into 32 classes (see Figures 19 and 20).

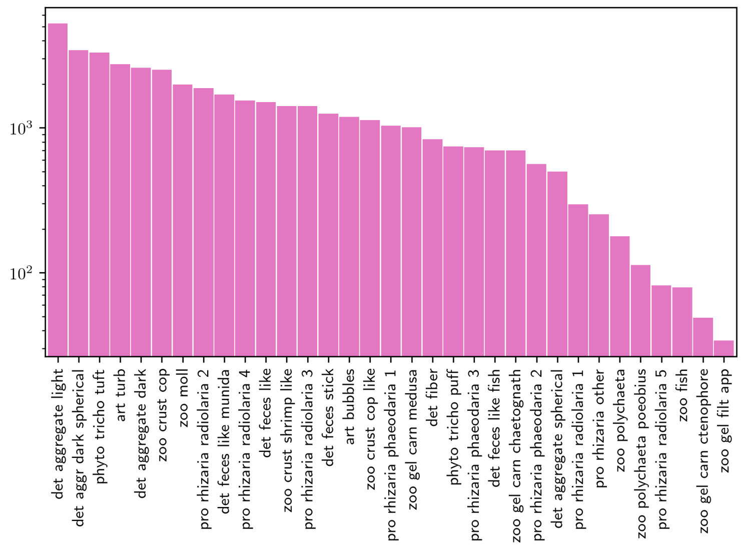


Figure 20: Distribution of classes found in the UVP5 dataset Each bar shows the abundance of a class in the UVP5 dataset.

Of the 33 classes, 18 of them have a high abundance of over 1000 annotations per class and a further six classes of over 500 but under 1000. Only the nine remaining classes have fewer items, with four classes having even fewer than 100 annotations. Unlike most of the other datasets, there are not a few classes that dominate the others in terms of abundance. The dataset was curated by Rainer Kiko for training of the deep learning component of PlanktonID (`https://planktonid.geomar.de`), therefore the distribution of classes is skewed by the selection process and not like in the original data acquisition. It features much less aggregates and objects that are hard to classify manually. Furthermore, the dataset was balanced by subsampling [R. Kiko, personal communication June 12, 2019]. The dimensionality reduction was carried out as in the previous datasets (see Section 2.2 PAP) limiting the classes to the ten most abundant ones and the number of annotations to a maximum of 500 by random subsampling (see Figure 21). The class art turb forms two clusters that are almost void of other annotations. Pro rhizaria radiolaria 2 and det feces like mundia form areas, where they are predominantly present. All other classes are mixed, although some of them can only be found in a specific part of the plot. All in all automatic classification is a non-trivial task for this dataset.

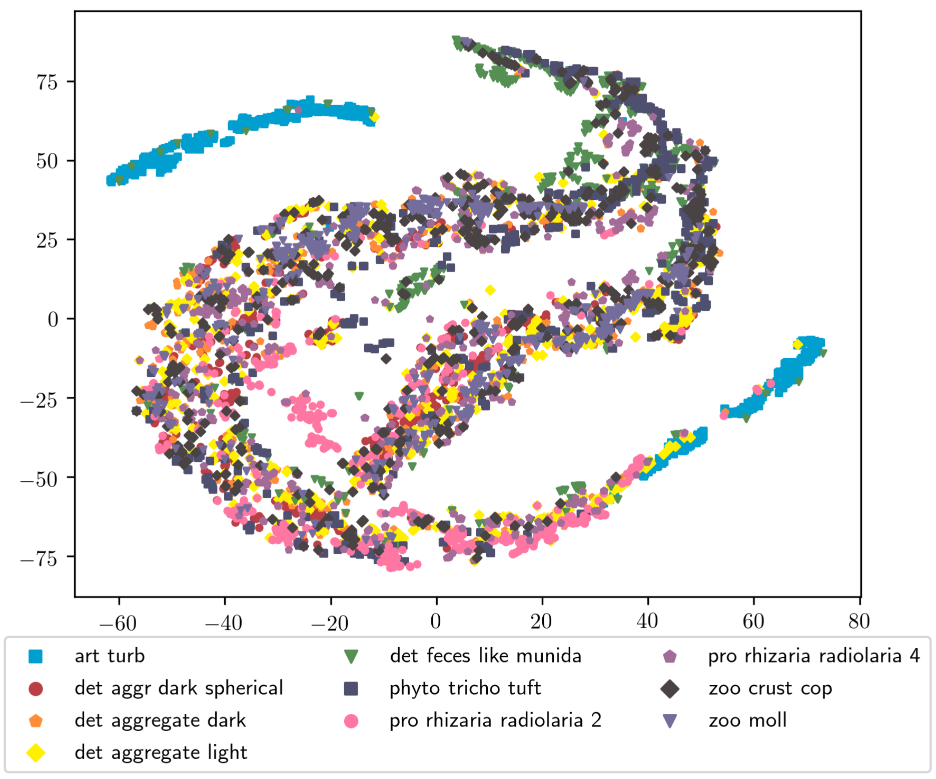


Figure 21: UVP5 data distribution t-SNE is applied to reduce the dimensionality of the UVP5 dataset to 2D. Each point is one annotation.

3

A craftsman's approach to marine computer vision

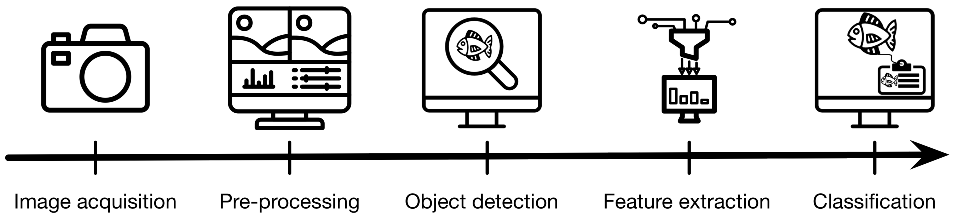


Figure 22: A typical Computer Vision (CV) pipeline Our CV pipeline consists of image acquisition, pre-processing, object detection, feature extraction, and classification.

Computer Vision (CV) is an interdisciplinary research area, which develops methods to make the information of visual images understandable for the computer.

> "Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that can interface with other thought processes and elicit appropriate action." [Shapiro, 1992]

In our context this means that marine benthic images are transformed to statistically usable information, i.e. counts of classes, or masses of classes, to generate knowledge. To achieve this we employ a computer vision pipeline. The computer vision pipeline as used in this thesis consists of image acquisition, pre-processing, object detection, feature extraction, and classification (see Figure 22. Image acquisition deals with the recording of marine images and was already discussed in Section 1.1 and is therefore omitted here. During pre-processing the images are analyzed and corrected for things such as illumination, color aberrations or blurring. Also, images unfit for downstream processing can be filtered out. Object detection deals with finding the interesting objects in the image. In the following, features are computed to turn visual images into mathematical representations. Last, the found objects are classified using the features extracted in the prior step. This information can be compiled into a list or a figure and further investigated by e.g. ecologists using statistical methods to gain knowledge.

3.1 Pre-processing

As already mentioned, marine images are usually affected by uneven illumination, color aberrations or blurring. Image pre-processing can be used to compensate for this. It should be noted that pre-processing can be either used to optimize for visually appealing photos, i.e. images that look "more beautiful", or to optimize for manual annotation, e.g. images are manipulated by changing brightness or contrast to make objects more recognizable, or to optimize for automatic recognition/classification.

In reality, it is quite difficult to define what "more beautiful" means. Visually appealing is often a very colorful, vivid image with a balanced color histogram. This often contradicts with the rather dull environment found in the deep-sea. Besides, this is of little scientific interest in the light of marine image informatics.

For manual annotation, it is best to give the user the ability to edit the image data dynamically and interactively while screening the image. Different classes have different features, which distinguish them from close relatives. Similarly, the images within a volume may vary due to different distances to the seabed or simply due to a change in habitat. Therefore different settings might fit for each image and class combination and image manipulating should then be part of the data exploration process (also cmp. [Ware, 2012] chapter 10).

This tends to be true for automatic classification as well. We do not know in advance where objects are located, which has an impact on their appearance. A corresponding manipulation of the image properties for the best possible further processing is therefore not possible. In addition, it is difficult to say which image manipulations help the classifier architecture to distinguish between classes. Changes that increase classification performance can even appear to be counterintuitive at first. An optimization for automatic classification might amplify disturbances, making the image even more difficult to annotate manually. An example from real life would be that a classifier was able to classify the shadows of water lily stems with high accuracy, but due to their slim appearance was not able to classify the water lily stems directly. Removing this shadow would drastically reduce the classification rate. Unfortunately, such phenomena are not known in advance and depend on the combination of image acquisition gear, location, depth, object properties and others. Therefore, there is not one perfect strategy for all datasets. A desirable goal is to have comparable conditions across all images, i.e. the luminance should be somewhat constant, as should the color composition of the object.

Typical disturbances (Figure 23) in marine imaging are vignetting (Figures 23a and 23b), color aberrations (Figures 23c and 23d), marine snow (Figures 23e and 23f) or blur (Figures 23g and 23h) caused by motion blur or autofocus failure. Vignetting is caused by an underpowered light source or in rare cases by the poor quality of the lens or a poor combination of camera and lens, i.e. the size of the camera chip is larger than the imaging area of the lens. Color aberrations can be caused by an underpowered light-source, too large distance of the imaging device to the seafloor or poor lens quality. Due to the wavelength-specific absorption of light in water, there are always at least slight color deviations in the images. "Marine snow is loosely defined as inanimate particles with a diameter greater than 0.5mm" [Lampitt, 2001]. In images, it can be perceived as mostly white matter that lies either on the seafloor or can be found in the water column. Marine snow in the water in combination with an overpowered lighting system, e.g. a strong flashlight, may lead to serious problems. The marine snow is too bright because it is closer to the lens than the seafloor, while the seafloor is quite dark due to the limited dynamic range of the camera chip. In this case, the autofocus if used may be deceived by the bright particles and focus them. This leads to a strong blurring of the seafloor, which is now out of focus. For more information about marine snow have a look at Silver, 2015. Blur can be caused by several factors, such as the above mentioned autofocus-failure, poor lens quality, insufficient light sources, too fast movement of the capture vehicle, scattering of light by turbid waters, or compression losses of the image format used.

Image pre-processing for the compensation of vignetting is possible by analyzing the luminance distribution in the image and modifying it. The methods can be divided into local and global methods. The global methods take a look at the global illumination or color distribution, while the local methods divide the image into smaller parts and analyze the local color or illumination distribution. A typical local method is CLAHE [Zuiderveld, 1994]. Global methods are histogram equalization, filtering with a Gaussian blur filter and subtraction of the results from the original image as described in Schoening, Bergmann, et al., 2012, or subtraction of the mean image of an image volume. In this thesis, the application of CLAHE and the subtraction of the guassian filtered image on the Hausgarten PS80/179-3 dataset (cmp. Section 2.1) were investigated.

Color aberrations are hard to compensate when there is no color reference plate, as the amount of light absorption is wavelength dependent and therefore too many parameters would need to be known to mathematically model the absorption and scattering of light to transform the image to reflect the real-world appearance. If a color plate is present in the image, it can be used as a reference and by restoring the known colors of the plate in the image to obtain an undisturbed image. Most methods without a reference plate use the distance to the ground to model the absorption using different light distribution/absorption assumptions [Galdran et al., 2015; Berman, Treibitz, and Avidan, 2017; Ancuti et al., 2018]. A quite simplistic approach is to use histogram equalization, which assumes that the composition of color is balanced in the images, which is most often not given in marine imaging, due to the wavelength dependent light absorption of water.

As marine snow is quite diverse, it is difficult to find a general method for removal. If it is small enough, it can be treated as salt and pepper noise. Removal using mean, median or bilateral filtering is possible, although introducing blur to a limited degree. For larger marine snow or to minimize blur modified median filters were proposed [Banerjee et al., 2014; Farhadifard, Radolko, and Lukas, 2017]. In Cyganek and Gongola, 2018 the authors propose a method employing a three-dimensional median filtering using the spatiotemperoal properties of the video frames to their advantage.

Blurring is almost impossible to remove, but it can be compensated using sharpening. This enhances the edges in the image. Unfortunately, artifacts, especially noise, are amplified as well. Richardson-Lucy deconvolution can be used [Richardson, 1972] as well as unsharp masking [M. Petrou and C. Petrou, 2010]. Here we test unsharp masking as it is often preferred to Richardson-Lucy deconvolution as it is more stable. For unsharp masking, a blurred image $blur(\mathcal{I})$ is computed and the enhanced image is then computed as

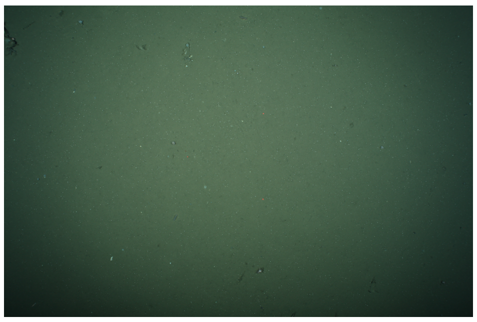
$$\mathcal{I}^{\text{filtered}} = \mathcal{I} + \lambda^{\text{usm}} * (\mathcal{I} - blur(\mathcal{I})),$$

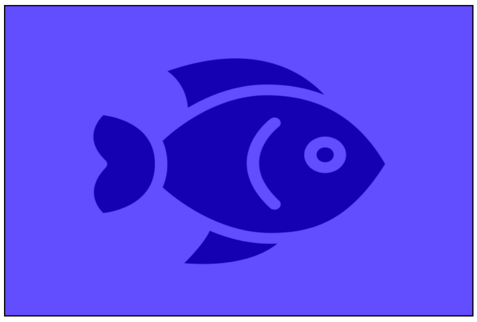
where λ^{usm} is a parameter increasing the influence of the so-called masked image $\mathcal{I} - blur(\mathcal{I})$.

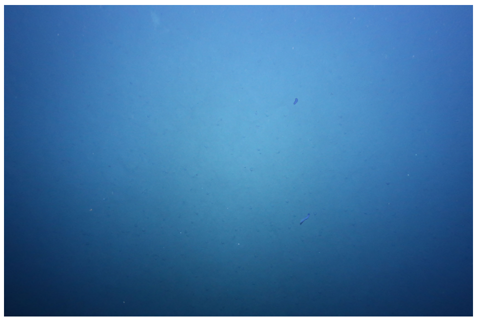
The visual results of applying unsharp masking, CLAHE, median filtering, histogram equalization, or subtraction of a Gaussian blur filtered image are displayed in Figure 25. Unsharp masking (Figure 25b) significantly sharpens the image, but also highlights the marine snow (small white dots) and increases the shadows of the objects. CLAHE (Figure 25c) slightly removes the vignetting (barely visible in print) and makes the structure in the sediment, but also on the shell more visible. The median filter (Figure 25d) removes the marine snow completely, but also blurs the image significantly, thus eliminating all fine-grained structure. Histogram equalization (Figure 25e) alters the image significantly. The sediment is greener, the marine snow is emphasized and the shell seems to be over-illuminated. The structure in the sediment is more pronounced. Subtraction of the Gaussian blurred image (Figure 25f) seems to produce a uniformly illuminated image. This can be seen even better when having a look at the full image (Figure 25f is an image excerpt) in Figure 25g and the full pre-processed image in Figure 25h. Unfortunately, the image seems to be quite grayish, lacking color.

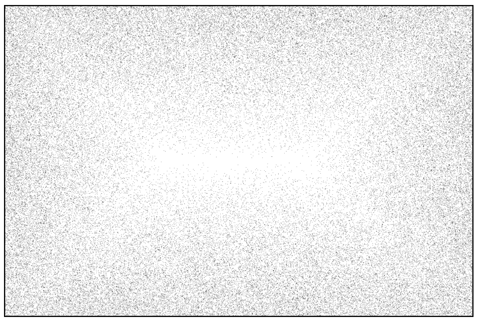
To evaluate the pre-processing methods qualitatively for automatic classification, a cross-validation experiment was performed, using MPEG-7 features and an SVM classifier. MPEG-7 features (for details have a look at Section 3.2) were selected because they consist of a mixture of color and texture features. Therefore impacts of pre-processing on both color and texture

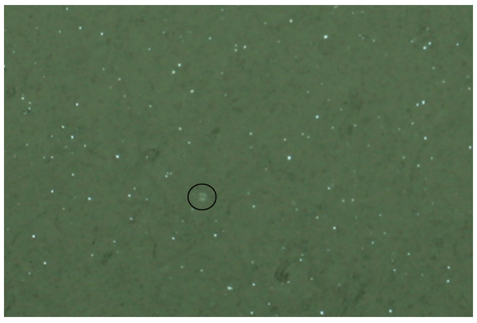
(a)


(b)

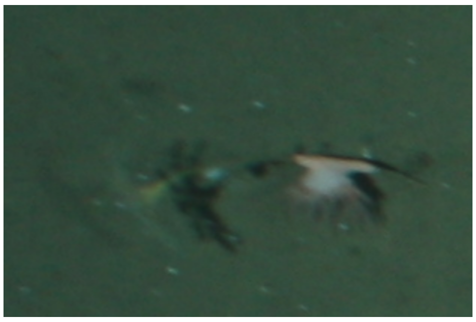

(c)


(d)


(e)


(f)

(g)



(h)

Figure 23: Typical marine image disturbances (*a*)/*b*) vignetting, *c)/d)* color aberration, *e)/f)* marine snow, *g/h)* (motion) blur, whereas the left image is an illustration of the problem and the right one a real-world example from Hausgarten PS80/179-3 ((b),(f),(h), cmp. Section 2.1), or SO242 ((d) cmp. appendix Section 7), The circle in (f) encircles a small round sponge, which is hard to distinguish from the marine snow.

can be analyzed. The SVM is a robust classifier, which is often favored as a baseline for classifier systems. As dataset, the Hausgarten PS80/179-3 was selected. Let $\mathcal{V}^{\text{Hausgarten}} = \{I_{i=0,\ldots,\mathsf{I}-1}\}$ with $\mathsf{I} = 1024$ images be the Hausgarten image volume. For these images I_i a set $\Lambda = \{\mathcal{A}_{j=0,\ldots,J-1}\}$ of point annotations were provided by experts, where J is the number of point annotations. For Hausgarten we have 1815 point annotations. Λ is split into ten equally sized folds $\Lambda^0, \ldots, \Lambda^9$. We now define a training set Λ^t and validation set Λ^v with $\Lambda = \Lambda^t \cup \Lambda^v$ and $\Lambda^t \cap \Lambda^v = \emptyset$. We repeat all experiments ten times to do cross-validation. In cross-validation run 0, $\Lambda^v = \Lambda^0$ and $\Lambda^t = \bigcup_{i=1}^{9} \Lambda^i$. For all following runs one fold from Λ^v is exchanged with one fold from Λ^t. For each experiment (see Figure 24), a pre-processing function $p : [0, 255]^{m\times n\times 3} \to [0, 255]^{m\times n\times 3}$ is applied to all I_i in $\mathcal{V}^{\text{Hausgarten}}$ with the result $p(I_i)$. On these pre-processed images $p(I_i)$ annotation patches of size 32×32 px centered around the point annotations $\mathcal{A}_j$ are extracted. We denote these with $\mathtt{I}_j$. For each annotation patch $\mathtt{I}_j$ we compute MPEG-7 features $x_j = f^{\text{mpeg7}}(\mathtt{I}_j)$. The set of all MPEG-7 features is denoted with $\Gamma = \{x_0, \ldots, x_{1814}\}$. Analog to the training/validation split of Λ we define $\Gamma = \Gamma^t \cup \Gamma^v$ and $\Gamma^t \cap \Gamma^v = \emptyset$. For each experiment, an SVM is trained with Γ^t and then the classification accuracy is estimated with Γ^v. The results are averaged over all ten cross-validation experiments. An SVM with radial basis function kernels was used and the penalty parameter, also-called slack value, was set to 1. For unsharp masking, a Gaussian blur filter with a 5×5 kernel was used and λ^{usm} was set to 1.0. For CLAHE the kernel was set to $\frac{1}{8} * M \times \frac{1}{8} * N$ with the clip limit set to 0.01 and 256 histogram bins. Subtraction of a Gaussian blur filter image with a kernel size of 701×701 was employed. Histogram equalization was computed on the RGB image and is parameterless. Median filtering with a disk structuring element with radius 5 was used, i.e. all pixels with Euclidean distance lower or equal 5 were set to 1.

The results of the cross-validation experiment are listed in Table 2. Unsharp masking improves the classification accuracy by eight basis points, CLAHE by six, while Gaussian blur filtering and histogram equalization still yield an increase of about four basis points. Only median filtering reduces the classification accuracy by twelve basis points. Although removing the marine snow, the median filtering also increases the blurring in the image, which hurts the performance significantly. Unsharp masking and CLAHE increase the visibility of fine-grained structures thus increasing the potential to extract distinctive texture features. Furthermore, both significantly increase the shadow of the objects, probably helping in the classification. Interestingly, the subtraction of the Gaussian blur filter, which eliminates almost all colors, does not hurt performance. Objects near the edge of an image can probably be better classified, increasing overall accuracy.

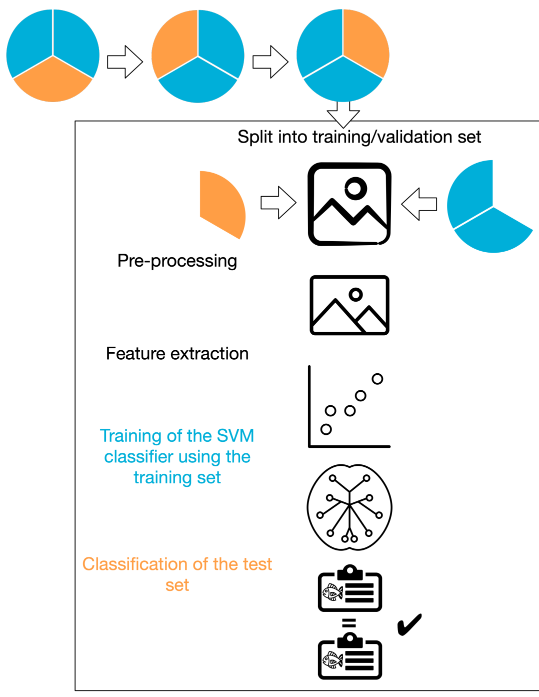
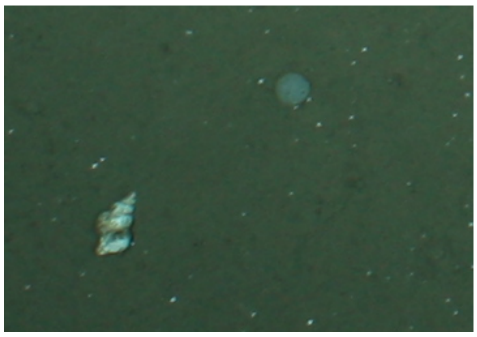
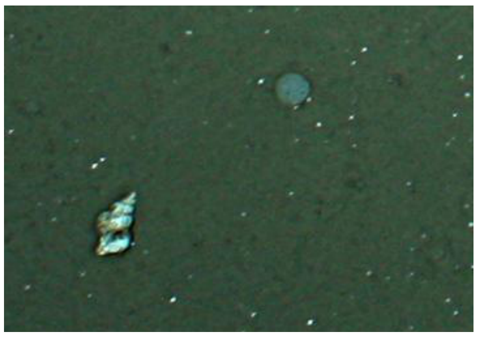

Split into training/validation set
Pre-processing
Feature extraction
Training of the SVM classifier using the training set
Classification of the test set
=
✔

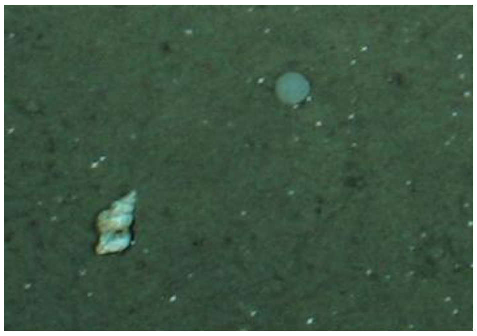
Figure 24: Cross-validation experiment for evaluation of different pre-processing strategies. For better visualization only a three fold cross-validation is depicted, although in our experiment ten folds were used. One experimental run, i.e. the execution of each step shown in the figure, is depicted, although we did five experimental runs. For each run a different pre-processing was applied – Unsharp masking, CLAHE, Subtraction of Gaussian blur, Histogram equalization, Median filtering.

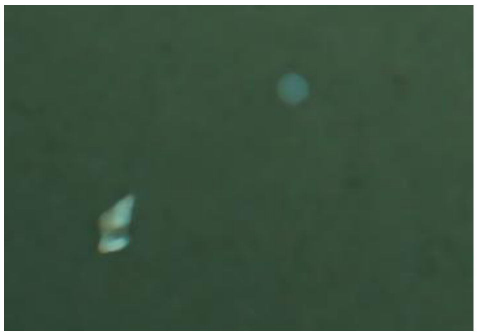
features	accuracy
no filter	78.35%
Unsharp masking	**86.61%**
CLAHE	84.57%
Subtraction of Gaussian blur	82.69%
Histogram equalization	82.31%
Median filtering	66.23%

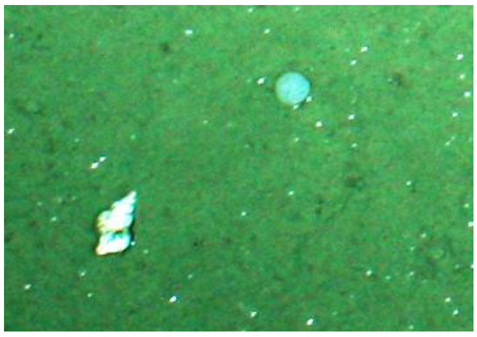
Table 2: SVM cross-validation classification accuracy on Hausgarten PS80/179-3 using different pre-processings. Best value is in green, second in orange and third in red.

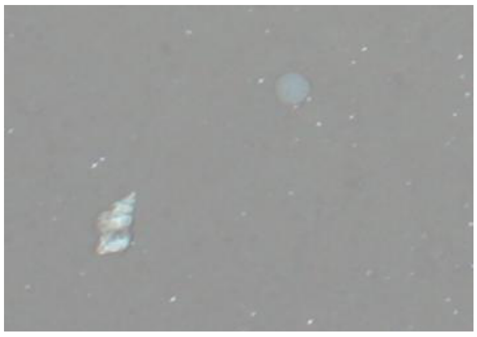

(a)

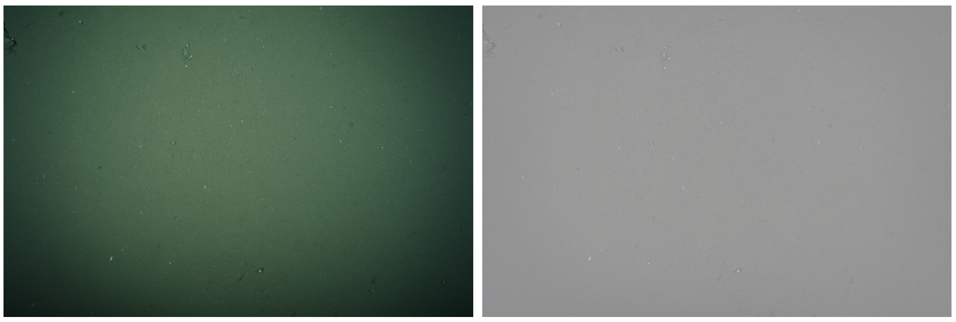

(b)


(c)


(d)


(e)


(f)



(g) **(h)**

Figure 25: Results of different pre-processing algorithms on a Hausgarten PS80/179-3 image (cmp. Section 2.1) *a)* original image excerpt, *b)* unsharp masking *c)* CLAHE, *d)* median filtering, *e)* histogram equalization, *f)* subtracting Gaussian blur filter, *g)* original complete image *h)* subtracting Gaussian blur filter. In order to be able to present the result better a) to f) are image excerpts. Nevertheless, the pre-processings were applied to the whole image. For the subtraction of the Gaussian blur filter the results are presented for both the image excerpt f) and the complete image h) to see the results on both levels.

3.2 Feature extraction

Feature extraction is the process of reducing the visual information of the images, to a computer-processable format needed to perform efficient and effective object detection or classification. Typical features extract peculiarities of images or annotation patches. These focus on color or texture. Formally, an annotation patch I_j is transformed to a feature x_j with a feature extraction function $x_j = f(\mathrm{I}_j)$. Haralick features, local binary patterns, threshold adjacency statistics, Zernike Moments and MPEG-7 features were evaluated on the Hausgarten and PAP datasets (cmp. Sections 2.1 and 2.2). If the features are not parameterless, the parameters were determined empirically. Parameters that are fixed for the experiments are listed in the respective sections. For some features a set of parameters has been tested. These are not listed in the respective sections, but only in the table of results.

Histogram features

The histogram features are computed on the intensity values, i.e. gray values, with B equally sized bins. The idea behind histogram features is that similar-looking annotation patches also have similar intensity distributions. Because of their simplicity, they are easy and fast to compute but also limited in their ability for classification purposes. An advantage of the histogram features is that they can be computed and compared on different sized annotation patches. Besides, they are rotation and translation invariant, i.e. even if the object is rotated (rotation invariance) or moved (translation invariance), the histogram stays the same. Formally, we define the feature as $x_j^{\text{hist}} = f^{\text{hist}}(\mathtt{I}_j) = \text{hist}(\mathtt{I}_j)$.

Square histogram feature

For many organisms, the body can be divided into a core part and extensions such as tentacles, spikes or arms, which are often quite different in morphology or color. The histogram feature treats all parts the same, regardless of the distribution of pixel intensity values. An example why this might lead to problems is depicted in Figure 26a and it's inverted version Figure 26b. The histograms are almost the same although the images are quite different. To counter this downside of histogram features, we propose the square histogram feature, which consist of two histograms. One is computed on the core, i.e. a square region centered on the center of the annotation patch, and the other one on the remainder. The size of the core can be provided as a parameter. It is given as a percentage of the side length of the annotation patch. Like the histogram feature, the square histogram feature can be computed and compared on arbitrarily sized objects, but rotation and translation invariance of the histogram features are lost in most cases (see Figure 26). Formally, we define the feature as $x_j^{\text{shf}} = f^{\text{shf}}(\mathtt{I}_j) = \Big(\text{hist}\big(\text{core}(\mathtt{I}_j)\big), \text{hist}\big(\text{remainder}(\mathtt{I}_j)\big)\Big)$.

Square median features/Super square features

A great deal of marine objects are roundish. Therefore defining rings $\mathtt{r}_0, \mathtt{r}_1, \ldots$ around the center (cmp. Figure 27) and the measurement of properties of the underlying pixels seems to be reasonable. The ring $\mathtt{r}_e$ is the e-th ring counted from the outside and is defined as $\{(n,m) | \min(n, m, N-n, M-m) = e\}$. We propose the square median features where for each ring the median is computed. This results in $x_j^{\text{smf}} = f^{\text{smf}}(\mathtt{I}_j) = (\text{median}(r_0), \text{median}(r_1), \ldots)$. Furthermore, we propose the super square feature, which also takes other statistical features into account. In addition to the median, mean, variance, standard deviation and peak to peak value, i.e. the difference between min and max value, are computed resulting in $x_j^{\text{ssf}} = f^{\text{ssf}}(\mathtt{I}_j) = \Big(\text{median}(\mathtt{r}_0), \text{mean}(\mathtt{r}_0), \text{var}(\mathtt{r}_0), \text{std}(\mathtt{r}_0), \text{ptp}(\mathtt{r}_0),$ $\text{median}(\mathtt{r}_1), \text{mean}(\mathtt{r}_1), \text{var}(\mathtt{r}_1), \text{std}(\mathtt{r}_1), \text{ptp}(\mathtt{r}_1), \ldots\Big)$.

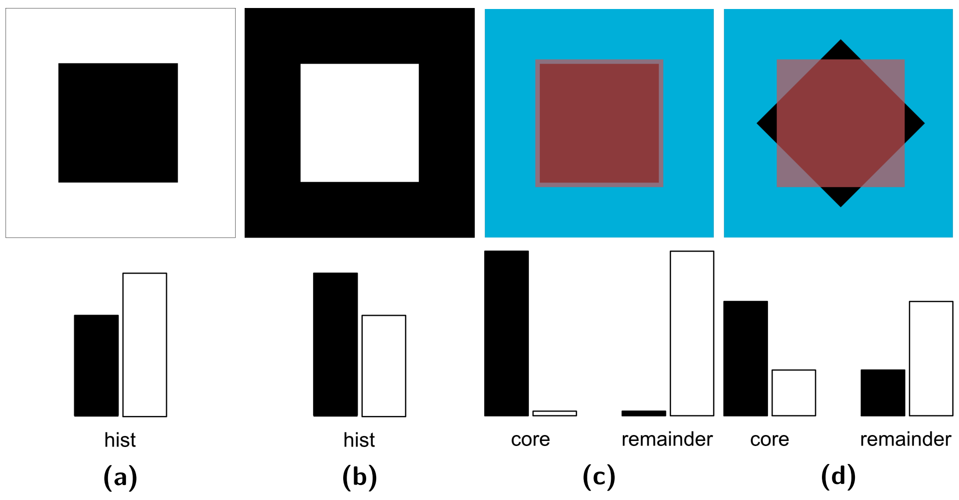


Figure 26: Square histogram feature *a)* An example image with the "common" histogram feature. The "common" histogram just shows the abundance of each intensity value, but not the local distribution. *b)* The same image as in a) but inverted. We see that the histogram of a) and b) are quite similar although the images are almost the opposite. *c)* The same image as a) with the square histogram feature (with the core (red) and remainder region (blue) colored). *d)* The same image as b) rotated by 45° object. Unfortunately, this feature is not rotation invariant.

Haralick features

Haralick, Dinstein, and Shanmugam, 1973 describe the Haralick texture features named after Haralick. There are 14 features based on the gray-level co-occurrence matrix, i.e. a matrix of size $O_g \times O_g$ with each entry counting the number of times a gray-level $g_{\hat{a}}$ is the neighbor to a gray-level $g_{\hat{b}}$, where $a, b = 0, \dots, O_g$ are the distinct gray-levels. Please note that there are different definitions for neighborhood (horizontal neighbors, vertical neighbors and neighbors in both diagonal directions). For the haralick features, we compute statistics for all these neighborhoods and concatenate the results. The features on this gray-level co-occurrence matrix are: angular second moment, contrast, correlation, sum of squares variance, inverse different moment, sum average, sum variance, sum entropy, entropy, difference variance, difference entropy, two information measures of correlation and the maximal correlation coefficient. Please note that in this study, only the first 13 features are computed, as the 14th is considered unstable [Mahotas, 2018]. For rotational invariance, the features for all neighborhood definitions are computed and therefore the features are $4 * 13 = 52$ dimensional. For more information on the features, have a look at the paper [Haralick, Dinstein, and Shanmugam, 1973]. Formally we define Haralick features as $x_j^{\text{har}} = f^{\text{har}}(\mathrm{I}_j) = \text{har}(\mathrm{I}_j)$.

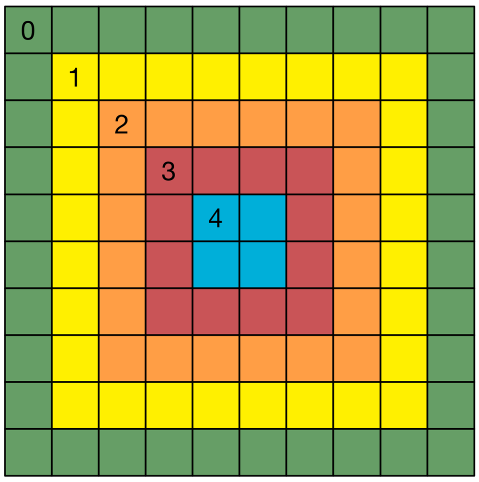


Figure 27: Ring structure of the square median feature and the super square feature For each ring statistical properties are computed – median for square median feature and median, mean, variance, standard deviation and peak to peak value for super square feature.

Local Binary Patterns

Local Binary Patterns are texture descriptors. For each pixel they compare it's value to the value of its neighboring pixels, where the neighboring pixels lie on a circle of radius r^{lbp} (here $r^{\text{lbp}} = 5$) sampled at p points (see Figure 28, here p $= 16$, please note that these points do not necessarily correspond to one pixel and therefore interpolation may be necessary). If the value of the neighboring pixel is smaller or equal, a 0 is generated, otherwise a 1, resulting in a p-digit binary number. The Local Binary Patterns are then generated by computing a histogram of these digits for all pixels in the image. Rotation invariance can be achieved by a bit shift on the p-digit number described in detail in Ojala, Pietikainen, and Maenpaa, 2002. Formally we define Local Binary Patterns as $x_j^{\text{lbp}} = f^{\text{lbp}}(\mathrm{I}_j) = \text{lpb}(\mathrm{I}_j)$.

Threshold adjacency statistics

For the threshold adjacency statistics, the average intensity μ is computed, without taking intensities below 30 into account. The image is thresholded so that

$$\mathcal{I}^{\text{thresholded}}{}_{(m,n)} = \begin{cases} 255 & \text{for } (\mu - 30) < \mathcal{I}_{(m,n)} < (\mu + 30) \\ 0 & \text{else} \end{cases} \tag{1}$$

where $\mathcal{I}_{(m,n)}$ is the value of the pixel at the position (m, n) in the image $\mathcal{I}$. For each white pixel in the threshold image, i.e. $\mathcal{I}^{\text{thresholded}}{}_{(m,n)} = 255$, the number of adjacent white pixels in a 3×3 neighborhood is computed. The thresh-

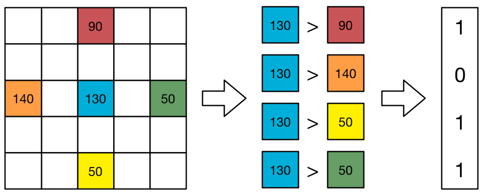


Figure 28: Depiction of the generation of the binary number for the Local Binary Pattern features A center pixel is compared to the neighboring pixels on a circle of radius r^{lbp} at p discrete points (for better visualization we choose $r^{\mathsf{lbp}} = 2$ and $p = 4$, although in the experiments we use $r^{\mathsf{lbp}} = 5$ and $p = 16$). For each of these neighboring pixels in anti-clockwise direction, we test if it's intensity is smaller than that of the center pixel and output the boolean value. All boolean values are concatenated to create a binary number.

old adjacency statistics are then the number of pixels with n white neighbors, normalized by the total number of white pixels in the 3×3 neighborhood, resulting in 9 statistics. Instead of thresholding with $(\mu - 30) < \mathcal{I}_{(m,n)} < (\mu + 30)$, thresholding using $(\mu - 30) < \mathcal{I}_{(m,n)} \leq 255$ and $\mu < \mathcal{I}_{(m,n)} \leq 255$ is performed. The threshold adjacency statistics computed on all threshold images are concatenated, resulting in a total number of 27 threshold statistics [Hamilton et al., 2007]. An extension that automatically determines the threshold value and sets the margin (here statically set to 30) is proposed in Coelho et al., 2010. Formally we define Threshold adjacency statistics as $x_j^{\mathsf{tas}} = f^{\mathsf{tas}}(\mathrm{I}_j) = \mathsf{tas}(\mathrm{I}_j)$.

Zernike moments

Moments were introduced to image analysis in 1962 by M.-K. Hu, 1962. They are global shape layout descriptors that usually have invariant properties and the ability to compensate for noise. However, it is necessary for them to assume that a view of the entire object without occlusion is given. In contrast to centralized moments, which are only translation invariant, Zernike moments [Teague, 1980] are additionally rotation variant. For the computation, so-called Zernike polynomials are used. Additional scale variance can be achieved through normalization [Nixon and Aguado, 2012]. Zernike moments of a maximum degree of 16 were used in this thesis on a radius of 8. Unfortunately, the assumption that there is always a clear view of the entire object does not necessarily apply to all marine imagery. Some organisms are known to be attached to other organisms or to hide behind flora. For more detailed information about the definition and computation of these Zernike moments, have a look at Teague, 1980. Formally we define Zernike moments as $x_j^{\mathsf{zer}} = f^{\mathsf{zer}}(\mathrm{I}_j) = \mathsf{zer}(\mathrm{I}_j)$.

MPEG-7

The MPEG-7 standard describes several features used primarily for image retrieval [Sikora, 2001]. Of these the ColorStructure descriptor, the ColorLayout descriptor, the DominantColor descriptor, the EdgeHistogram descriptor and the ScalableColor descriptor are used. These can be used individually or in concatenation. Due to different value ranges of each descriptor it is reasonable to normalize the descriptors blockwise, i.e. every descriptor is normalized to unit length.

The Color Structure descriptor is used with a descriptor size of 32. It describes the local color structure of an image using a structuring element. With a descriptor size of 32, there are 32 quantized colors: $c_0, \ldots, c_{31}$. The quantized colors are computed in the HMMD color space (for a description see [Manjunath et al., 2001]). A histogram is then computed by sliding the structuring element over the image increasing a histogram bin when the respective color is found at least once in the structuring element. For a more detailed description see [Manjunath et al., 2001].

The Color Layout descriptor is used with a Y and Cr/Cb descriptor size of 32. It describes the spatial distribution of the color of an image. The descriptor size is the number of coefficients computed for each color component in the YCrCb color space (for a description please have a look at [Manjunath et al., 2001]). The descriptor is computed by laying an 8×8 grid over the image and extracting the average color of each block in the grid, followed by a discrete cosine transformation and encoding. For the exact encoding scheme and further information see [Manjunath et al., 2001].

The Dominant Color descriptor consists of five dominant colors. These are determined by clustering the colors of the image. These colors are stored with their percentage of occurrence in the image normalized to 5 bits. The optional variances of the dominant colors are computed as well as the optional spatial coherency, which provides information about the compactness of the dominant colors in the image (large blobs or spread all over the image). For a more detailed description of the clustering and the descriptor have a look a [Manjunath et al., 2001].

The Scalable Color descriptor is used with 128 coefficients and without discarded bitplanes. Therefore the HSV space is quantized into evenly sized 256 bins (16 levels for H and 4 levels for each of S and V). A histogram of the image is computed and the values are truncated to 11 bits and a nonlinear quantization is applied to further map to 4 bit. After that, a Haar transform is applied to the histogram to provide scalability for the descriptor, i.e. descriptors with different number of coefficients can still be compared. The authors call this fine-to-coarse comparison. For more detail on the transformation and descriptor see [Manjunath et al., 2001].

The Edge Histogram descriptor is parameterless. The image is cut into 4×4 equally sized blocks. For each block, a histogram of the edges types, i.e. vertical edge, horizontal edge, 45-degree edge, 135-degree edge and a bin for all non-directional edges, i.e. all other edges neither resembling one of the aforementioned, is computed. This results in a $4*4*5 = 80$ dimensional feature vector [Won, D. K. Park, and S.-J. Park, 2002].

Formally we define MPEG-7 features as $x_j^{\mathsf{mpeg7}} = f^{\mathsf{mpeg7}}(\mathrm{I}_j) = \left(\frac{\mathsf{csd}(\mathrm{I}_j)}{||\mathsf{csd}(\mathrm{I}_j)||}, \frac{\mathsf{cld}(\mathrm{I}_j)}{||\mathsf{cld}(\mathrm{I}_j)||}, \frac{\mathsf{dc}(\mathrm{I}_j)}{||\mathsf{dc}(\mathrm{I}_j)||}, \frac{\mathsf{sc}(\mathrm{I}_j)}{||\mathsf{sc}(\mathrm{I}_j)||}, \frac{\mathsf{ehd}(\mathrm{I}_j)}{||\mathsf{ehd}(\mathrm{I}_j)||}\right)$.

SVM cross-validation for feature comparison

An SVM cross-validation experiment was carried out analogous to the one in the previous section, but with 5 instead of 10 folds, (cmp. Section 3.1). Instead of exchanging and evaluating the different pre-processings, we evaluate the different features presented in this section. Two datasets, Hausgarten and PAP, were evaluated. The results can be found in Table 3. The best descriptor in both cases was the Color Layout descriptor of the MPEG-7 feature descriptors. On the Hausgarten dataset, the runner-ups are the concatenation of all the presented MPEG-7 features and finally, the Local Binary Patterns. For the PAP dataset, MPEG-7 features are outperformed by the super square feature computed on the Lab color space, followed by the square histogram features with a bin size of 18. This experiment shows that the square histogram features are quite competitive regarding the PAP dataset, although for the Hausgarten dataset tend to be less performant. Due to their fast computation, they are an asset to keep in mind. Although the MPEG-7 features are performing not as well as the CSD alone, one should keep in mind that the CSD might be quite fit for these two datasets. Nevertheless, the MPEG-7 features carry a lot more potential in cases where the color layout is not as discriminative as texture features. Therefore in future experiments, MPEG-7 is favored in most cases.

features	Hausgarten	PAP
Color Layout descriptor	**88.37%**	**75.68%**
MPEG-7 normalized	78.34%	71.41%
Local Binary Patterns	73.71%	72.72%
Dominant Color descriptor	70.79%	52.15%
Threshold adjacency statistics (RGB)	69.64%	67.02%
Threshold adjacency statistics (gray)	64.62%	60.16%
Square histogram feature (16)	63.91%	72.30%
Square histogram feature (18)	63.74%	72.97%
Square histogram feature (14)	63.03%	70.65%
Super square feature (Lab)	61.21%	73.01%
Edge Histogram descriptor	60.82%	68.65%
Square histogram feature (12)	60.77%	70.26%
Square histogram feature (8)	58.73%	56.63%
Square histogram feature (10)	58.23%	62.68%
Square histogram feature (6)	48.53%	62.75%
Scalable Color descriptor	46.50%	50.74%
Square histogram feature (4)	43.19%	44.32%
MPEG-7	39.77%	68.05%
Zernike moments	39.50%	40.57%
Color Structure descriptor	37.90%	38.20%
Haralick features (gray)	34.53%	48.54%
Square median feature (gray)	34.04%	44.03%
Haralick features (RGB)	33.75%	41.30%
Square histogram feature (15	33.38%	35.14%
Square median feature (RGB)	33.38%	37.01%
Super square feature (RGB)	33.38%	37.01%
Histogram (5)	33.38%	35.14%
Histogram (10)	33.38%	35.14%
Histogram (20)	33.38%	35.14%

Table 3: SVM cross-validation classification results. For threshold adjacency statistics, Haralick features and square median feature variants computed on the RGB and gray color space exist. These are denoted with (RGB) and (gray). For the Super square feature, a variant on the Lab color space was computed and denoted with (Lab). For the histogram features the number of bins is listed after the feature in brackets, e.g. (12).

3.3 Object detection

Object detection deals with finding interesting objects, also-called regions of interest (ROI), in an image, usually as a prior step to classification. Although not the main focus of this thesis, object detection is briefly discussed here.

Human experts

One of the most effective ways to get accurate ROI detections is the use of human experts, although there are problems with the use of human labor concerning object detection, such as attention problems or biases [Schoening, Osterloff, and Nattkemper, 2016]. Unfortunately, this method is also the most expensive one, since the number of experts and their time is limited.

Citizen Science

In the context of image analysis, citizen scientists (CS) are usually non-domain experts who perform a detection or classification task with a web-application. Citizen scientists ROI detection are more easily available, but the number of participating CS and the quality of their annotations are highly dependent on the problem at hand.

If the task of the CS is of broad interest or related to goodwill, e.g. to protect the environment (dolphins/whales/oceans/pollution), to help victims of natural disasters or to help the poor, it is expected that a fair amount of people are eager to join this effort, if promoted correctly. In addition, there must be some kind of reward, which can either be monetary, based on the belief to support a good cause, or based on a reward system applied through gamification of the effort [Deterding et al., 2011; Bowser et al., 2013]. In gamification, the CS receive virtual rewards, which can be, e.g. a title or a badge for fulfilling certain tasks. More sophisticated approaches such as rankings or special events associated with festivities, e.g. tied to World Oceans Day, can further increase the popularity of the effort.

CS annotation studies can result in a huge amount of possible ROIs. Nevertheless, special care has to be taken about post-processing of the received data. If the task was not well defined or too challenging, e.g. the CS were not told in detail what to look for or the objects to look for are hard to recognize, many false ROI detections will be the result. Usually, by creating a gold standard, i.e. creating a new annotation set that contains only overlapping ROIs of multiple CS will reduce this problem. If a global bias is introduced, such as a poorly described task definition, even the use of a gold standard will be of little use. The preparation of the CS for the task to be performed is therefore of crucial importance.

Entropy-driven ROI detection

An annotation-free method for detecting points of interest is based on local Shannon entropy [Shannon, 1948] and inspired by Johnson-Roberson, Pizarro, and S. Williams, 2010. The Shannon entropy is used to create saliency maps, i.e. a visual representation of the Shannon entropy at each pixel. The points of interest are peaks in these saliency maps. The idea of this approach is that usually underwater images, especially in the deep-sea, have a rather homogeneous background and therefore a low entropy. In contrast, objects of interest stand out. Therefore, they introduce a local entropy peak in the homogeneous background area.

The image $\mathcal{I}$ of size $m \times n \times 3$ is converted to grayscale. For each sub-image of size $M' \times N', M' \leq M, N' \leq N$ with center point (c, d) a histogram ξ with B bins is computed. If $\xi(b)$ is the number of occurrences at bin b the local entropy at position (c, d) is defined as

$$H'_{(c,d)} = - \sum_{b \in B} \frac{\xi(b)}{M' * N'} * \log_2 \left(\frac{\xi(b)}{M' * N'} \right) \tag{2}$$

This results in an entropy-based saliency map $H_{(m,n)} = H'_{(m,n)} \forall (m, n)$. H is binarized with threshold $t = \mathcal{T} * \max(H)$ to retain a fraction of $\mathcal{T}$ of the most entropic pixels in H. The lower the value of $\mathcal{T}$, the more ROIs are retained, but they may be less relevant. Morphological operations are applied to H to avoid less relevant small regions and to close holes in detected regions. Contours are extracted and circles are computed that enclose these contours with a minimal radius. A factor ϵ is multiplied with the computed circle radius to compensate for the underestimation of the object radius, which usually happens with this method.

The advantage of an annotation-free method is that it can be used to generate ROIs without any prior knowledge. Therefore, it is most helpful when annotating a new dataset. In addition, it is fair concerning that no assumptions are made other than the presumption of a homogeneous background. However, this presumption is not valid for all marine datasets, especially in shallow waters. Coral reefs are very diverse, so local Shannon entropy would not be a good indicator for a ROI. Nevertheless, in the deep-sea setting, this detector is a valuable asset. Another advantage is that the computation of the local Shannon entropy is quite fast.

3.4 Classification

K-Nearest Neighbor

Most classification algorithms reduce the training data to obtain a model of it that is usually smaller in size. In contrast to all other presented classifiers, there is no training of a model, since the decision for a label is made solely on the basis of the training data, more specifically on the K-nearest-neighbors. For classification with only a few annotations, the computation of a model would be ineffective. So often, the K-Nearest Neighbor (KNN [Cover and Hart, 1967] algorithm is preferred. In addition, the KNN algorithm is quite fast for smaller datasets. Let Ω_j be the label of the feature vector x_j. With $y(\Omega_j)$ we will denote the so-called One-Hot-encoding of the label of the feature vector x_j, i.e. $y(\Omega_j) \in \{0,1\}^{|\Omega|}$ with the element corresponding to Ω_j is set to one and all others being zero. If not specified otherwise, $y(\Omega_j)$ consists of only a one and all others zero. However, if an annotator is unsure about his decision for one class, the likelihood for more than one class can be encoded as well, by assigning percentage values to the respective classes. A normalized output $y'(x_j)$, i.e. $||y'(x_j)||^2 = 1$ for feature vector x_j, also known as probability estimates can be directly computed for the KNN as follows:

$$y'(x_j) = \frac{1}{K} \sum_{k=0}^{K-1} y(\Omega_{\pi_k}) \tag{3}$$

where π is the permutation that permutates $a = (0, \ldots, J-1)$ (reminder: J is the number of annotations) such that $||x - x_{\pi_a}|| \leq ||x - x_{\pi_{a+1}}||$. The number of nearest-neighbors to consider is a parameter K. Here we set $K = \min(J, 5)$ depending on the number of annotations J. This formulation allows the classification of data, even for a minimal number of annotations.

Support Vector Machine

The Support Vector Machine (SVM [Cortes and Vapnik, 1995] is a more sophisticated classifier. However, more labeled data is needed to train it because a model is learned from the data. The SVM learns a class-separating hyperplane. It can separate classes that cannot be separated linearly with the so-called kernel trick. A kernel function is used to transform the data to a high dimensional space, where a separating hyperplane can be found. Usually, the classifier is made for binary classification. Nevertheless, for multi-class classification a one-vs-one scheme or a one-vs-rest scheme can be applied. In each setting, multiple SVMs are trained. In the one-vs-one scheme, SVMs for all combinations of classes are trained and a majority vote is used to decide for a class. We prefer the one-vs-one scheme, due to the increased speed-up when employing multi-processing. Although a larger number of SVMs must be trained, each SVM uses only a small amount of data, which can be processed much faster. A normalized output, i.e. probability estimates, is computed using the method described in [Wu, C.-J. Lin, and Weng, 2004]. A radial basis function kernel is used. The penalty parameter C, also known as slack, is automatically determined out of the values $[10^0, 10^1, 10^2, 10^3]$, while γ is set to $\frac{1}{J}$.

Hierarchical Hyperbolic Self Organizing Maps

The Self Organizing Map (SOM) is a neural network proposed by Teuvo Kohonen [Kohonen, 1982]. It was proposed to perform dimensionality reduction and clustering, i.e. generating a descriptive model of the data in a low dimensional space. The basic setup consists of a set of H neurons $\{(u_h, z_h)_{h=1,\dots,H}\}$ that are arranged in a grid with z_h being the grid coordinate and u_h being the attached neural unit also-called the prototype. The architecture of the grid differs throughout the various types of SOMs proposed since then.

In the Hyperbolic SOM (HSOM) [Ritter, 1999] the algorithm is defined for the non-Euclidean hyperbolic space. Thus the grid coordinates z_h are also defined in the hyperbolic space. The Hyperbolic Hierarchical SOM (H^{2}SOM [Ontrup and Helge Ritter, 2005] additionally introduces a hierarchical grid structure to the hyperbolic version. The grid consists of a number of rings $\mathfrak{r}$, with neural units on the ring, centered around a central neural unit. Each ring is built in such a way that each neural unit (except for the ones at the edge of the grid) has s neighbors. A neighbor is defined as another neural unit sharing a connection. Using the Möbius transformation, the grid is built by placing a central grid node and spawning its s children around it. This is done recursively for every neural unit until all have s neighbors and the maximum number of rings $\mathfrak{r}$ is reached. An example of an H^{2}SOM grid with two rings and seven neighbors $(s = 7, \mathfrak{r} = 2)$ is shown in Figure 29. For further information refer to [Ontrup and Helge Ritter, 2005].

The learning of a non-Euclidean SOM is done equivalently to an Euclidean SOM. The learning steps of SOM algorithms are described below.

- Creation of the grid structure
- For each learning step
    - Choose a random stimulus x_j out of the training set Γ^t

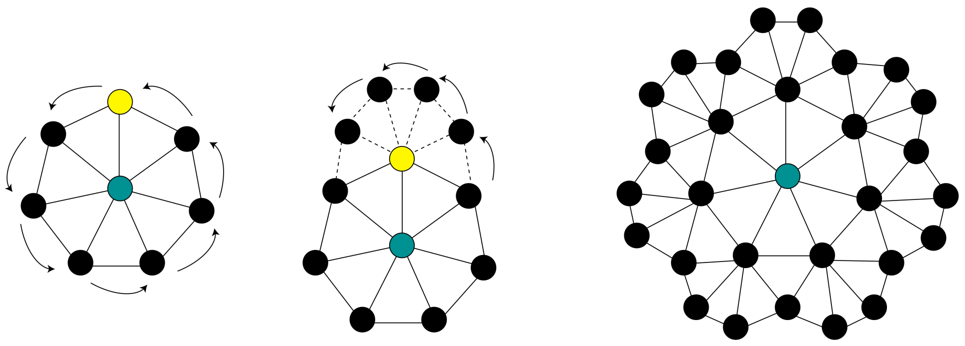


Figure 29: Creation of an H^{2}SOM grid using the Möbius transformation Depicted is H^{2}SOM grid with $\mathfrak{r} = 2$ rings and $s = 7$ neighbors.

- Determine the neural unit (also known as best matching unit (BMU)) with the minimum distance to the stimulus, i.e.

$$u_h^t = \arg\min_h ||u_h - x_j||^2$$

- Update the BMU with the iterative adaption rule

$$u_h^{t+1} = u_h^t + \epsilon^t * \mathfrak{n}^t(j,h) * (x_j - u_h^t)$$

  where ϵ^t is a learning rate and $\mathfrak{n}(j,h)$ is a neighborhood function.

- The neighbors of the winning neuron are also updated, but to a lesser degree, which is mediated by the neighborhood function.

In contrast to the Euclidean SOM a refined neighborhood function $\hat{\mathfrak{n}}(x_j, u_h^t)$ (cmp. Equation (4)) is used for the H^{2}SOM, taking into account the change from Euclidean to hyperbolic space.

$$\hat{\mathfrak{n}}^t(j,b) = \exp\left(-\frac{\left(2\text{arctanh}\left(\left|\frac{z_j - z_h}{1-\bar{z}_j z_h}\right|\right)\right)^2}{\sigma^2(t)}\right) \tag{4}$$

In addition, training is performed ring-wise, i.e. all the steps except for the creation of the grid structure are performed for each ring individually.

The number of neural units in the grid of an H^{2}SOM grows exponentially with the number of rings $\mathfrak{r}$. This leads to a more trustworthy mapping but dramatically increases the time required for the search of the BMU during training. Leveraging the hierarchical structure of the grid, a beam search is applied to approximate the global BMU in each training step. The search starts with the central neural unit as the initial BMU. For a beam width of $\mathfrak{w} = 1$, it continues by recursively choosing the BMU among the children of the last winning neural unit until it reaches the current periphery of the grid. The BMU determined for the last ring is an approximation of the global BMU. For values of $\mathfrak{w} > 1$ searching is done equivalently, but the children of $\mathfrak{w}$ different neural units are searched for the BMU. It has been shown in [Ontrup and Helge Ritter, 2005] that this strategy significantly accelerates the training while staying close to or even surpassing the embedding performance using the aforementioned global search.

The H^{2}SOM depends on parameters that need to be optimized. These are the number of rings $\mathfrak{r}$, the number of children s, the neighborhood adaption modifier σ and the learning rate ϵ^t. The algorithm is quite robust against changes in ϵ^t and σ but the parameters determining size $\mathfrak{r}, s$ are important.

H^{2}SOM classifier *Most of the following method has been published in Langenkämper, Goesmann, and Nattkemper, 2014.*

As mentioned above, the H^{2}SOM has not been developed for classification. It is an unsupervised learning algorithm, i.e. data without labels are grouped together in clusters. Nevertheless, it can be extended to be used as a classifier. This way, it is turned into a supervised classifier, i.e. labeled data is used to train a machine learning algorithm for classification. After training the H^{2}SOM, neural units can be labeled. This is done with a labeling function $\mathfrak{l}(u_h)$ that is defined on the Voronoi cell $V(u_h)$ of the training data for each neural unit

$$V(u_h) = \{x_j \in \Gamma^t | \Delta(x_j, u_h) < \Delta(x_j, u_c), \forall c \neq h\}$$

using a given metric Δ (in our case the Euclidean metric). We propose majority voting $\mathfrak{l}^{\mathsf{maj}}$ which is defined as

$$\begin{aligned} \mathfrak{l}^{\mathsf{maj}}(u_h) &= \arg\max_{\Omega_l}(\Psi(V(u_h), \Omega_l)) \\ \text{with } \Psi(V(u_h), \Omega_l) &= |\{x_j \in V(u_h) | \Omega_j = \Omega_l\}| \end{aligned} \tag{5}$$

This method labels a neural unit u_h with the majority of the labels of the attached data points. It assumes that most of the neural units are "pure", i.e. most labels in a Voronoi cell are the same. A labeled H^{2}SOM as described above can be used for classification. To label, i.e. to classify, an unknown feature vector x_j a classification function $\mathfrak{c}(x_j)$ is used. We propose different functions $[\mathfrak{c}^{\mathsf{nn}}(x_j), \mathfrak{c}^{\mathsf{thresh}}(x_j), \mathfrak{c}^{\mathsf{nbrs}}(x_j)]$, defined in the following.

The most straightforward function is to assign x_j to the label $\mathfrak{l}(u_h)$, which is assigned to the nearest neighbor u_h in the H^{2}SOM.

$$\mathfrak{c}^{\mathsf{nn}}(x_j) = \mathfrak{l}(u_h) \text{ with } h = (\arg\min_c \Delta(x_j, u_c)) \tag{6}$$

Furthermore, the Euclidean distance $\Delta(x_j, u_h)$ can be seen as a certainty measure that the BMU u_h is the correct association of x_j. Therefore, we define an arbitrary threshold β beyond which the association is assumed to be uncertain and therefore a special rejection label $\Omega_\emptyset$ is assigned. The value of β

is empirically determined. This function can be used to reject data that does not match the learned data distribution.

$$\mathfrak{c}^{\text{thresh}}(x_j) = \begin{cases} \mathfrak{c}^{\text{nn}}(x_j), & \text{if } \Delta(x_j, u_h) < \beta \text{ with } h = \arg\min_c \Delta(x_j, u_c) \\ \Omega_\emptyset, & \text{else} \end{cases} \tag{7}$$

The previous strategies determine the label in a Winner-Takes-All (WTA) manner. We can use the H^{2}SOM property that neighboring neural units, i.e. grid neighbors, share common properties, usually referred to as "neighborhood preservation". For the third proposed classification function (cmp. Equation (8)), the neighborhood of a BMU is evaluated, i.e. the neighbors of the BMU in the same ring. This smoothes out unlikely assignments with a large BMU distance and "taxonomic disagreement" of the neighborhood.

$$\mathfrak{c}^{\text{nbrs}}(x_j) = \begin{cases} \mathfrak{l}(u_{h+1}), & \text{if } (\Delta(x_j, u_{h+1}) + \Delta(x_j, u_{h-1})) \\ & < (3 * \Delta(x_j, u_h)) \wedge \mathfrak{l}(u_{h-1}) = \mathfrak{l}(u_{h+1}) \\ & \text{with } h = \arg\min_c \Delta(x_j, u_c) \\ \mathfrak{c}^{\text{nn}}(x_j), & \text{else} \end{cases} \tag{8}$$

Hierarchical Hyperbolic Self Organizing Linear Maps For the H^{2}SOM classifier, we made the assumption that the neural units are more or less "pure". If this is not the case a number of data points are just ignored, due to the majority voting used for labeling the neural units. The majority voting labels the neural unit with the most abundant label. All other labels are ignored. Therefore low abundant classes are much less likely to be considered in the model. In addition no normalized output is provided by any of the H^{2}SOM classifier variants. Therefore, we propose the Hierarchical Hyperbolic Self Organizing Linear Maps (H^{2}SOL [Langenkämper and Nattkemper, 2016]). The H^{2}SOL is a new classification algorithm based on the idea of the Hierarchical Hyperbolic Self Organizing Maps (H^{2}SOM) algorithm [Ontrup and Helge Ritter, 2005]. Like the H^{2}SOM classifier, an H^{2}SOM is trained and maps high-dimensional data onto the hierarchical grid of nodes in the hyberbolic plane. The H^{2}SOL uses a local linear classification function attached to each node u_h following the scheme proposed in [Nattkemper, Helge Ritter, and Schubert, 2001] (Equation (9)).

$$y'(x) = ||\theta_h + A_h(x - u_h)||^2 \tag{9}$$

with $\theta_h \in \mathbb{R}^{\text{out}}, A_h \in \mathbb{R}^{\text{in} \times \text{out}}, u_h \in \mathbb{R}^{\text{in}}, x \in \mathbb{R}^{\text{in}}$. The parameters θ_h, A_h are learned according to the following update rules (Equations (10) and (11)).

$$\Delta\theta_h = \epsilon_h * (y(x) - y'(x)) \tag{10}$$

$$\Delta A_h = \epsilon_h * (y(x) - y'(x)) * \frac{(x - u_h)^T}{||x - u_h||} \tag{11}$$

with ϵ_h being a learning rate.

Digression: Balanced extension to the H^{2}SOL

Data imbalance is a problem eminent in biological data. Different strategies to tackle these exist. For more detailed problem description and solutions have a look at Section 4.4. To compensate this problem in the H^{2}SOL, we propose the balanced extension of the H^{2}SOL. Instead of randomly choosing a training data element x_j with probability $p = \frac{1}{J}$, we assign a probability according to the frequency of the class $\Omega^{(l)}$ of that training data, i.e. x_j with probability $p = \frac{1}{|\Omega^{(l)}|*L}$ where L is the number of classes and $\Omega^{(l)}$ the l-th class. This way, samples of rare classes are presented more frequently to the classifier than with random sampling. This increasing their representation in the model and thus increasing the classification accuracy of these rare classes.

Results

In a first experiment, we conducted a 10 fold cross-validation study to compare the classifiers' performance on three datasets – Hausgarten, PAP and UVP5. It was taken care that each algorithms received the same training and validation folds to make sure the results are comparable. As we want to test different parameters for some algorithms we define an additional dataset, the optimization set Γ^o. For optimization and evaluation on the same dataset, a nested cross-validation (CV) is used to ensure that the validation set Γ^v is always truly independent and that a separation between the training set Γ^t, the validation set Γ^v and the optimization set Γ^o is always guaranteed. Otherwise, we would get so-called data-leakage [Kaufman et al., 2012] and thus bias the results. To prevent this, an outer and an inner CV are performed. The outer CV is performed as described in Section 3.1 and evaluates the classifier trained on Γ^t against the validation set Γ^v for each iteration of the CV. Another CV is performed to determine the optimal parameterization for the classifier. This CV splits Γ^t in $\mathfrak{k}$ folds. In each iteration, $\mathfrak{k} - 1$ folds are used as

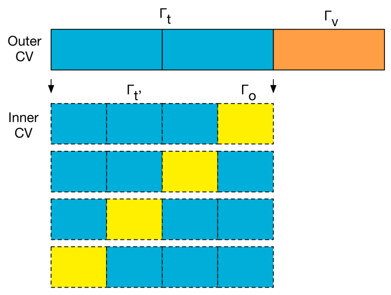


Figure 30: A nested cross-validation (CV) is used to perform optimization and evaluation on one dataset. The nested CV uses an outer loop (Outer CV) which is identical to the aforementioned CV. Here only one step of the outer CV is shown for brevity. On the training set Γ^t a further CV is performed in the inner loop (Inner CV) for optimization of the classifiers parameters for the training of Γ^t.

training set $\Gamma^{t'}$ to train a classifier and the remaining fold Γ^{o} is used for evaluating the classifier. For each parameter set, an inner CV is performed and the parameter set with the best average accuracy is chosen. This parameter set is used to train a classifier on Γ^{t} and evaluate on Γ^{v}. For each iteration of the outer CV as described in Section 3.1 this inner CV was performed anew, possibly yielding different parameterizations every iteration. For the H^{2}SOL and H^{2}SOM classifiers the number of rings and spread factor $(\mathfrak{r}, s) \in \{(2,7), (2,8), (2,9), (3,7), (3,8), (4,7)\}$ were evaluated. For the SVM, the penalty parameter, also known as slack variable $C \in \{1, 10, 100, 1000\}$ was evaluated. The KNN was used as a baseline and therefore, we omit the time-consuming parameter optimization. The number of nearest-neighbors k was empirically set to 3 throughout all experiments.

A ten-fold outer CV combined with a four-fold inner CV was performed. No pre-processing was used as we mainly wanted to compare the classifiers. Besides, the selected datasets don't feature severe aberrations, which would have a tremendous impact on the classification performance. As features MPEG-7 features were chosen due to their wide range of applicability and their good results in the feature comparison. As classifiers the K-Nearest Neighbor, H^{2}SOL, H^{2}SOM Classifier, H^{2}SOM Classifier with neighbor smoothing classification scheme, H^{2}SOM Classifier with rejection and Support Vector Machine were tested.

In a second experiment, we try to answer the question of how much data is needed and how well the classifiers perform with a certain amount of data. For Hausgarten let $\Gamma^{t}_{J'} = \{x_0, x_1, \ldots, x_{J'}\}$ with $J' \in \{5, 6, 7, \ldots, 1500\}$ be the training set for each experiment with an increasing number of training samples. For brevity, we formally define this only for the Hausgarten dataset. Analog to above we define $\Gamma^{t}_{J'}$ for PAP with $J' \in \{5, 105, 205, \ldots, 28305\}$ and for UVP5 with $J' \in \{5, 105, 205, \ldots, 7805\}$. As validation set we always use the remaining data samples Γ^{v} disjoint from $\Gamma^{t}_{J'}$, $\Gamma^{t}_{J'} \cap \Gamma^{v} = \emptyset$. The parameterization was gained from the first cross-validation experiment (see above). As the number of training data samples might be much lower in this experiment, the parameters for parts of these experiments might not be fit. However, an exhaustive parameter search for each experiment is computationally unfeasible. For the same reasons, we omit a cross-validation.

Hausgarten The parameterization is automatically determined for each fold as described in Section 3.4. Please note that multiple cross-validation runs were performed and thus multiple parameter sets per algorithm will be reported. For the H^{2}SOL for $(\mathfrak{r}, s)$ it is $2 \times (2,7), 2 \times (2,8), 3 \times (2,9)$ and $3 \times (3,7)$, and for the H^{2}SOM classifiers it is $1 \times (2,8), 2 \times (2,9), 2 \times (3,8)$ and $5 \times (4,7)$ and for the SVM for C it is 6×1000 and 4×100. When looking at the average F_1-scores in Table 4 we see that the SVM outperforms all other algorithms by at least four percentage points. When looking at the single class performances

	KNN	H^2SOL	H^2SOMClassifier	H^2SOMClassifier(NbrS)	H^2SOMClassifier(Rej)	SVM
small round sponge	0.82	0.82	0.82	0.82	0.85	0.88
Kolga hyalina	0.75	0.71	0.61	0.58	0.62	0.83
Elpidia heckeri	0.81	0.82	0.75	0.75	0.78	0.85
Bathycrinus cf. carpenteri	0.83	0.79	0.69	0.68	0.73	0.84
bathycrinus stalks	0.48	0.42	0.29	0.27	0.32	0.48
burrowing purple anemone	0.87	0.87	0.86	0.86	0.88	0.94
small white anemone	0.88	0.85	0.85	0.85	0.89	0.9
shell	0.89	0.77	0.77	0.75	0.81	0.87
shrimp	0.95	0.98	0.95	0.95	1.0	1.0
avg	0.83	0.8	0.77	0.76	0.8	**0.87**

Table 4: Hausgarten - F_1-scores of the classifiers presented in this section on the Hausgarten dataset. The H^2SOM-Classifier(Rej) rejected 12.78% of validation data. A more detailed table additionally showing recall and precision can be found in the Appendix (Table 18).

(Table 4) and the confusion matrices (Figure 31, the H^2SOM classifiers were omitted for brevity) we see the same confusion pattern and similar single class performances for all three classifiers, with some very minor deviations. Besides KNN, H^2SOL and SVM perform equally well on almost all classes with around 0.8 F_1-score, except for bathycrinus stalks and shrimp. The bathycrinus stalks are a hard task as these are thin elongated objects without a distinctive color. The orientation of the stalks changes throughout the dataset and because the dataset is quite small, there might be not enough samples per orientation to classify these satisfiability. Shrimp is the class which could be classified best with an F_1-score of 0.95 to 1.0, which is not surprising as the big red shrimps really differ from other classes in that dataset. The SVM has the best performance for all classes. The H^2SOM classifiers all underperform for some classes such as *Kolga hyalina*, *Bathycrinus cf. carpenteri* and bathycrinus stalks. Although they perform worse for some classes, overall they are only slightly worse in average performance. This is expected because the capacity of their model is limited, i.e. the H^2SOM classifiers can only model a limited amount of data and are likely to suppress smaller or overlapping classes. This is also underlined by the parameterization of the H^2SOM classifiers which use far larger networks in contrast to the H^2SOL, which benefits from the addition of the local linear classifiers attached to the neural units to model more data.

In the second experiment, we answer the question of how much data is needed for each classifier to perform the task accurately. The results are shown in Figure 32. All classifiers roughly follow the same trend. At 100 samples, all classifiers already surpass 65% accuracy. The SVM surpasses the 80% accuracy at 369 samples, the KNN at 579 and the H^2SOL at 795 samples. The SVM is known to achieve quite reasonable results with few data samples, which is also shown in this experiment. The KNN can also deal with a very limited amount of data but usually reaches a plateau phase quiet early, which surprisingly does not happen here. The H^2SOL is not that fit for limited data but should accel with a larger amount of data but still remains below its capabilities here.

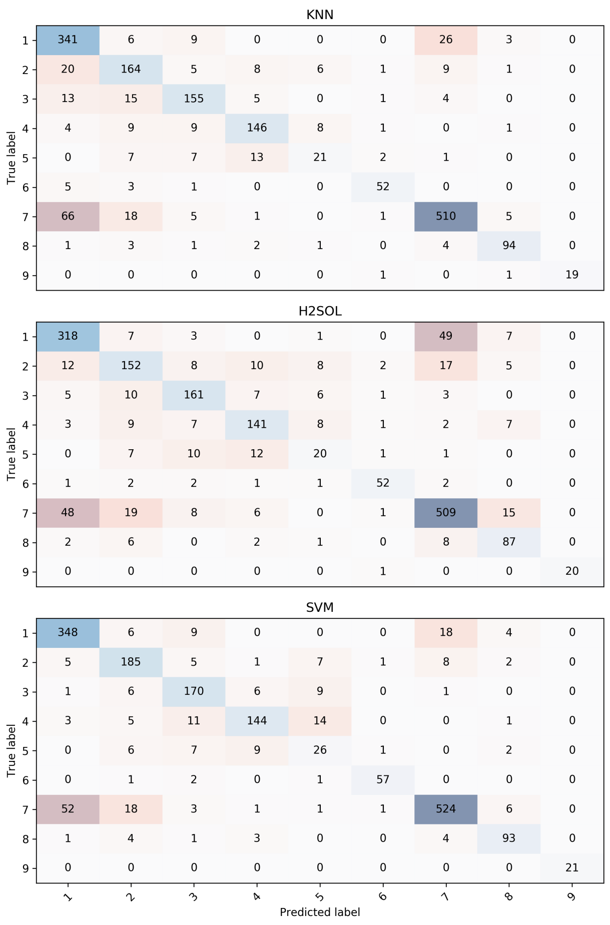


Figure 31: Confusion matrizes for the Hausgarten dataset 1) small round sponge, 2) *Kolga hyalina*, 3) *Elpidia heckeri*, 4) *Bathycrinus cf. carpenteri*, 5) bathycrinus stalks, 6) burrowing purple anemone, 7) small white anemone, 8) shell 9) shrimp

PAP The parameterization is automatically determined for each fold as described in Section 3.4. For the H^{2}SOL for $(\mathfrak{r}, s)$ it is $10 \times (2, 7)$, and for the H^{2}SOM classifiers it is $4 \times (3, 8)$ and $6 \times (4, 7)$ and for the SVM for C it is 10×1000. For the PAP dataset the results are shown in Table 5. In terms of average performance, the H^{2}SOL outperforms the KNN by 7 percentage points, the SVM even by 8 percentage points and the H^{2}SOM classifiers by more than 16 percentage points. The SVM has a slight lead in all classes, except *Amperima*, cnidaria 2, 8 and 10 and *Ophiuroidea* where the H^{2}SOL outperforms the rest. KNN and the H^{2}SOM-Classifier(Rej) perform best at classifying the cnidaria 16 class, whereas the SVM performs best for *Amperima* and all other H^{2}SOM classifiers and the H^{2}SOL perform best at classifying *Ophiuroidea*.

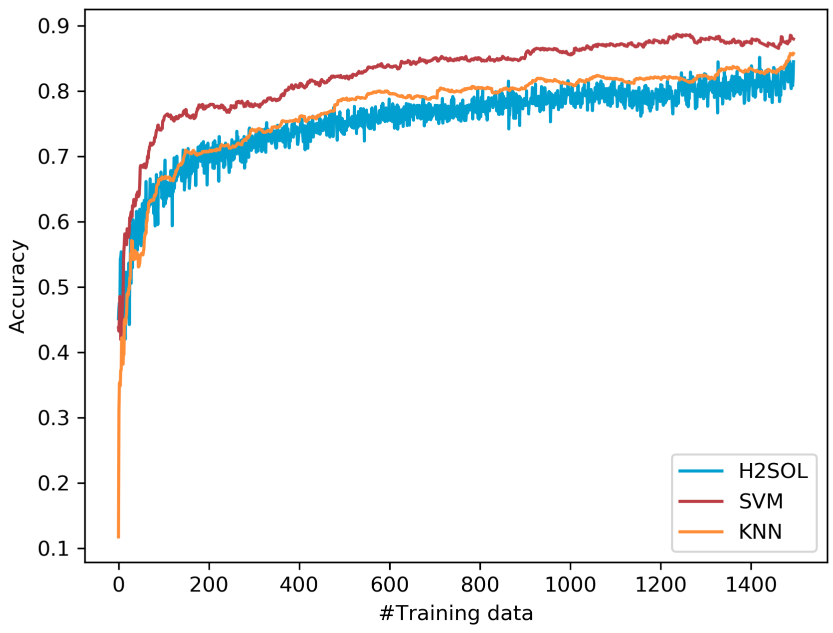


Figure 32: Performance of different classifiers on the Hausgarten dataset The accuracy for the Hausgarten dataset is shown for different classifiers, for different amounts of training data, showing the impact of limited training data.

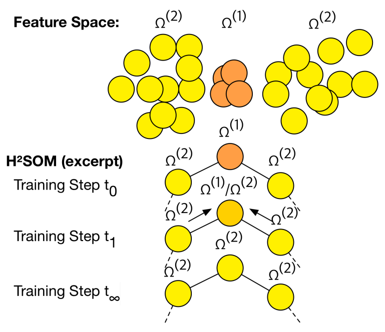


Figure 33: Neighborhood preservation in the H^{2}SOM. For more details have a look at the text.

For the H^{2}SOM classifiers we see a lot of classes with an F$_1$-score of zero, showing the limited classification capabilities for this kind of multi-class classification task with imbalanced data distribution. It highlights the inability of the H^{2}SOM classifiers to correctly model classes with a low abundance, which is due to the limited model size and the majority voting during labeling, i.e. the usually large amount of data is transformed to a rather small number of neural units. Because majority voting is used to label the neural units, low abundant classes tend to be neglected and are not well or even not at all represented in the final model. In addition, the H^{2}SOM was designed to be used for visualization purposes it features the so-called neighborhood preservation. Neighborhood preservation tries to preserve the neighborhood of the input data in the low dimensional space it projects to, thus also featuring only minor changes between neighboring neural units. This leads to so-called interpolation neural units, as depicted in Figure 33. The few samples of $\Omega^{(1)}$ are at first represented in the H^{2}SOM at t_0 but are then successively superseded by samples of $\Omega^{(2)}$, thus forming an interpolation neural unit. The H^{2}SOL bypasses this problem by introducing the local classifier, thus representing all data. The F_1-score might not be enough to have a look at in this case. Although the SVM is outperformed by the H^{2}SOL and the KNN in terms of F_1-score it has a superior precision with 0.68 compared to 0.62 (H^{2}SOL) and 0.56 (KNN) (see Appendix Table 19). However, the recall is significantly worse than that of the other two competitors – 0.49 (SVM) compared to 0.66 (H^{2}SOL) and 0.57 (KNN).

When looking at the confusion matrices (Figure 34), the patterns of KNN and H^{2}SOL are quite similar, but the confusion matrix of the SVM differs significantly. KNN and H^{2}SOL confuse class *Amperima* for cnidaria 2 and vice versa as well as cnidaria 2 for *Ophiuroidea* and vice versa. In addition, many classes are mistaken to be *Amperima* and many are falsely classified as *Ophiuroidea*. The SVM also classifies many samples to be *Ophiuroidea*, although they are not. But instead of confusing *Ophiuroidea* with cnidaria 2 it confuses them with cnidaria 9. In addition, cnidaria 2 is mistaken for cnidaria 9 and ophiuroidea disk. Furthermore, many samples of the *Amperima* class are classified incorrectly. Interestingly enough, the SVM hardly makes the misclassifications that are present in the other two classifiers. In addition, misclassifications like classifying class $\Omega^{(1)}$ as $\Omega^{(2)}$ but not vice versa are quite common in contrast to the other two classifiers. This is likely due to the different mechanisms used for classification. The KNN evaluates the very local data distribution. The H^{2}SOL also evaluates the local data distribution but looks at much more data therefore being not that prone to local changes. The SVM evaluates the data distribution of one class against the rest and uses a voting scheme on the winning class, which is likely accountable for the different classification results. Interestingly the different cnidaria morphotypes, which are quite similar in morphology, are not confused significantly except for cnidaria 2 and cnidaria 9 with the SVM.

Figure 35 shows the classification accuracy for the PAP dataset with an increasing number of training samples. The accuracy of the KNN quickly rises and reaches a plateau phase at about 10000 data points, which is quite typical for the KNN, especially with larger datasets with a large number of classes, as the local evaluation is quite error-prone with overlapping data distributions of many classes. The H^{2}SOL rises slower but steadily until the end of the experiment, which is also to be expected as it needs a certain amount of data to accel. The SVM fluctuates at first, reaches a temporary plateau phase and then slowly rises. In contrast to the other two algorithms it underperforms.

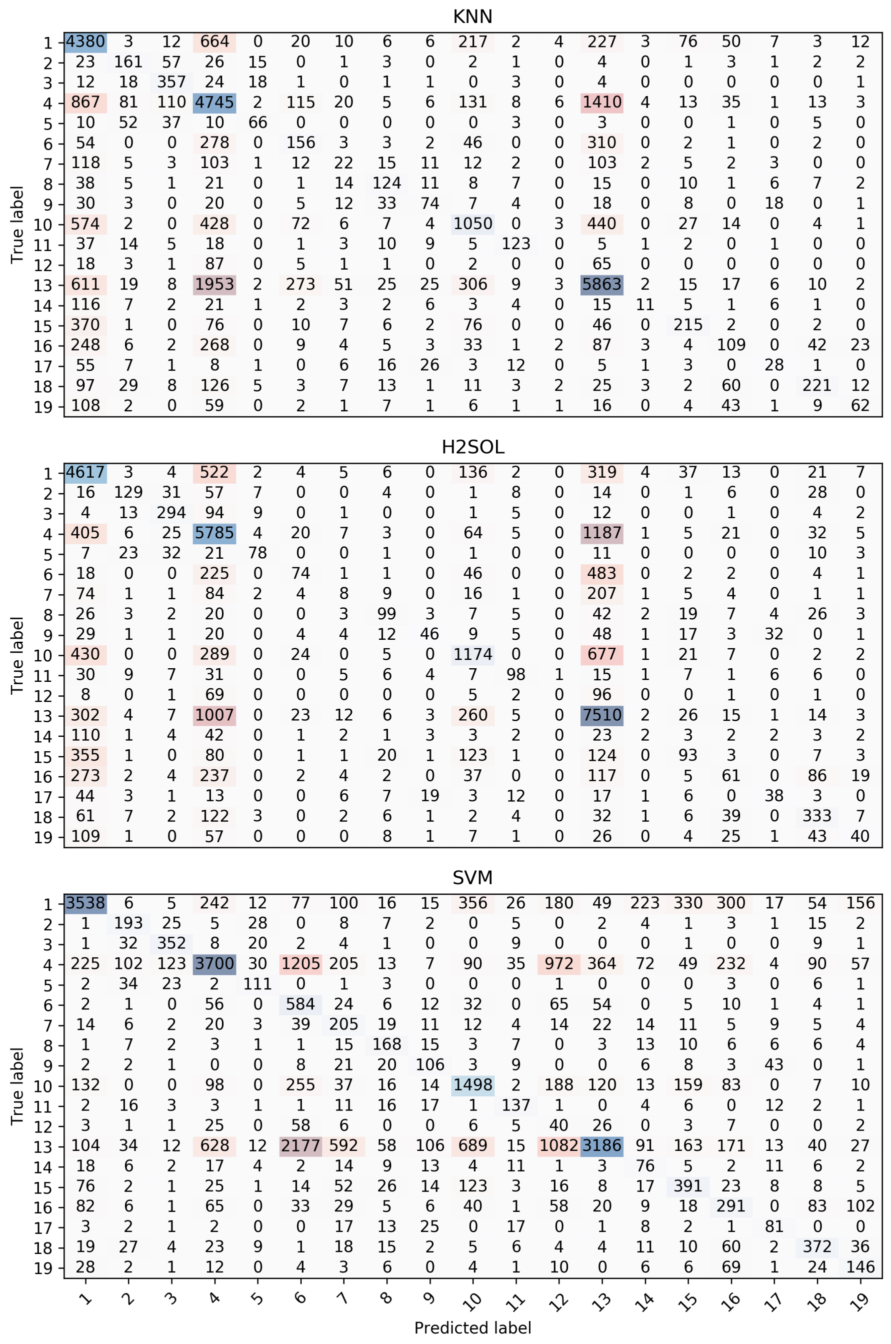


Figure 34: Confusion matrizes for the PAP dataset 1) *Amperima*, 2) cnidaria 10, 3) cnidaria 16, 4) cnidaria 2, 5) cnidaria 8, 6) cnidaria 9, 7) crinoid 1, 8) crinoid 2, 9) *Echiura*, 10) *Foraminifera*, 11) *Oneirophanta*, 12) ophiuroidea disk, 13) *Ophiuroidea*, 14) *Peniagone*, 15) polychaete 1, 16) *Porifera*, 17) *Pseudostichopus villosus*, 18) stalked tunicate, 19) *Tunicata*

	KNN	H^2SOL	H^2SOMClassifier	H^2SOMClassifier(NbrS)	H^2SOMClassifier(Rej)	SVM
Amperima	0.65	0.73	0.59	0.56	0.6	0.71
cnidaria 10	0.45	0.51	0.04	0.04	0.05	0.49
cnidaria 16	0.68	0.69	0.6	0.6	0.65	0.7
cnidaria 2	0.57	0.71	0.49	0.47	0.5	0.59
cnidaria 8	0.44	0.53	0.38	0.34	0.4	0.53
cnidaria 9	0.2	0.15	0.0	0.0	0.01	0.22
crinoid 1	0.07	0.03	0.0	0.0	0.0	0.23
crinoid 2	0.45	0.42	0.1	0.08	0.11	0.49
Echiura	0.35	0.29	0.0	0.0	0.0	0.35
Foraminifera	0.46	0.52	0.25	0.18	0.25	0.54
Oneirophanta	0.59	0.5	0.23	0.14	0.25	0.52
ophiuroidea disk	0.0	0.0	0.0	0.0	0.0	0.03
Ophiuroidea	0.66	0.75	0.62	0.61	0.62	0.49
Peniagone	0.09	0.02	0.0	0.0	0.0	0.2
polychaete 1	0.36	0.17	0.07	0.06	0.08	0.39
Porifera	0.18	0.12	0.04	0.03	0.04	0.27
Pseudostichopus villosus	0.22	0.3	0.05	0.04	0.05	0.42
stalked tunicate	0.47	0.53	0.36	0.31	0.38	0.55
Tunicata	0.28	0.19	0.0	0.0	0.0	0.33
avg	0.55	**0.62**	0.45	0.43	0.46	0.54

Table 5: PAP - F_1-scores of the classifiers presented in this section on the PAP dataset. The H^2SOM-Classifier(Rej) rejected 10.60% of validation data. A more detailed table additionally showing recall and precision can be found in the Appendix (Table 19).

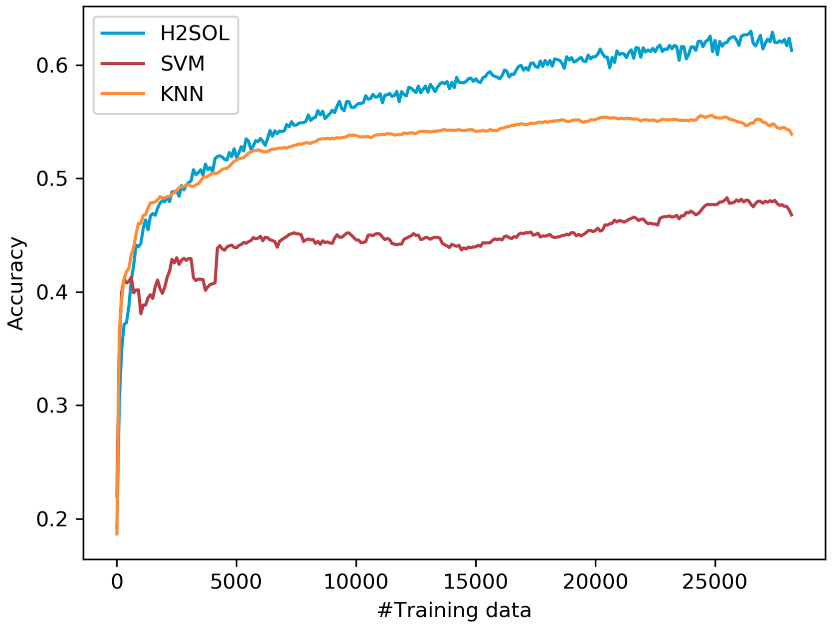


Figure 35: Performance of different classifiers on the PAP dataset The accuracy for the PAP dataset is shown for different classifiers, for different amounts of training data, showing the impact of limited training data.

UVP5 The parameterization is automatically determined for each fold as described in Section 3.4. For the H^{2}SOL for $(\mathfrak{r}, s)$ it is $5 \times (2,7), 3 \times (2,8)$ and $2 \times (2,9)$, and for the H^{2}SOM classifiers it is $1 \times (3,7), 6 \times (3,8)$ and $3 \times (4,7)$ and for the SVM for C it is 10×1000. We have used a subset of 19 classes of the original UVP5 dataset because of its considerable number of classes and dataset size. Just as above MPEG-7 features were used. As the dataset was grayscale and MPEG-7 includes several color-specific features, we also evaluated features more fit for gray values. We therefore used the square histogram features with 18 bins (shf18) as proposed in Section 3.2. The parameterization for the cross-validation experiment and the shf18 features is also automatically determined. For the H^{2}SOL $(\mathfrak{r}, s)$ is $2 \times (2,7), 2 \times (2,8), 5 \times (2,9)$ and $1 \times (3,7)$, and for the H^{2}SOM classifiers it is $2 \times (3,8)$ and $8 \times (4,7)$ and for the SVM, C is 10×1000. The results of the cross-validation experiments are shown in Table 6. Let us first have a detailed look at the evaluation using the MPEG-7 features. The H^{2}SOL wins nine classes, the SVM six and the KNN four. The H^{2}SOM-Classifier(Rej) is en par with the SVM for the art turb class, however the good performance of H^{2}SOMClassifier(Rej) is also due to rejection of samples. For all classifiers the art turb class is the class that is classified best. As this class is just cloudy looking mist, which fills (most of) the patch, it is quite distinguishable compared to the rest of the classes. For the shf18 features the KNN and H^{2}SOL win 12 respective 11 classes, with most classes being a tie between both. Three classes are won by the SVM. The class which can be classified best is now det aggregate spherical. This is rather hard to explain. The class is just a black and roundish object, which is not in the center but positioned in the left of the image almost all of the time. A possible explanation would be that the histogram computed on the center of the image does only contain white pixels and this is unique in this dataset.

When looking at the confusion matrices for the UVP5 dataset using the shf18 features (Figure 36) we see that there are few classes that are misclassified to be det aggregate light and phyto tricho tuft by all classifiers. For the KNN and the H^{2}SOL it is the other way around, i.e. there are few samples predicted to be det aggregate light and phyto tricho tuft but are actually another class. For the SVM this is not true. Nonetheless, the SVM features fewer misclassifications for art bubbles, det aggregate dark and phyto tricho puff. Other than that, there is no general pattern observable and the mispredictions are spread over almost all classes. The confusion matrices for the UVP5 dataset using the MPEG-7 features can be found in the appendix (Figure 78). The general observations are true there, as well.

In Figure 37 the classification accuracy for the UVP5 dataset with an increasing number of training samples (MPEG-7) is shown. We can see that the performance of the H^{2}SOL and KNN rises quickly with slight advantages for the KNN at first and later for the H^{2}SOL. The performance of the SVM rises slower and reaches a plateau at about 4000 samples. However, it stays well below the other two algorithms. Nevertheless, the overall performance of all algorithms is

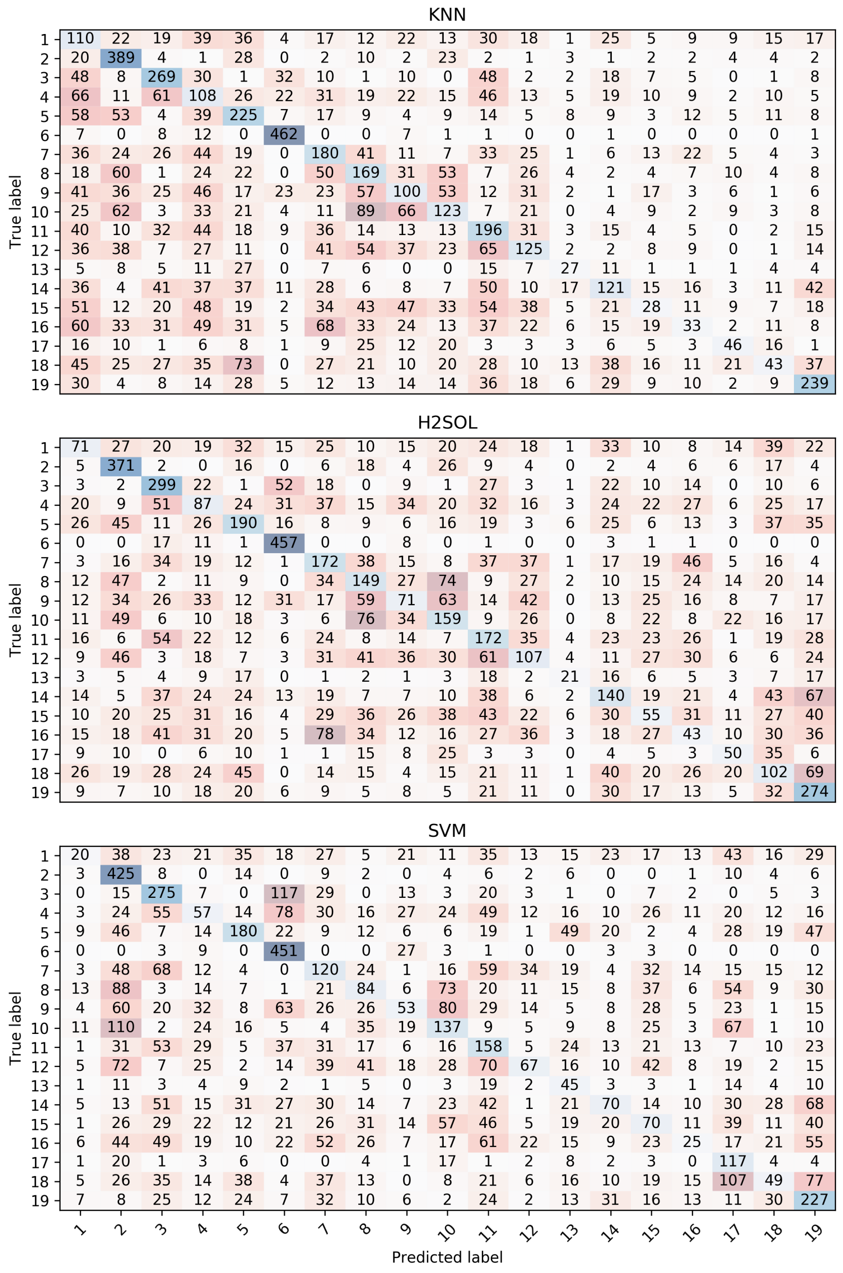


Figure 36: Confusion matrizes for the UVP5 dataset using shf18 features 1) art bubbles, 2) art turb, 3) det aggregate dark spherical, 4) det aggr dark spherical, 5) det aggregate dark, 6) det aggregate light, 7) det aggregate spherical, 8) det feces like fish, 9) det feces like munida, 10) det feces stick, 11) det fiber, 12) phyto tricho puff, 13) phyto tricho tuft, 14) pro rhizaria other, 15) pro rhizaria radiolaria 2, 16) zoo crust cop, 17) zoo crust cop like, 18) zoo gel carn medusa, 19) zoo moll

	KNN		H^2SOL		H^2SOMClassifier		H^2SOMClassifier(NbrS)		H^2SOMClassifier(Rej)		SVM	
	MPEG-7	shf18	MPEG-7	shf18	MPEG-7	shf18	MPEG-7	shf18	MPEG-7	shf18	MPEG-7	shf18
art bubbles	0.18	0.19	0.18	0.2	0.14	0.11	0.13	0.1	0.1	0.06	0.19	0.08
art turb	0.89	0.59	0.88	0.6	0.87	0.5	0.87	0.49	0.9	0.51	0.9	0.53
det aggr dark spherical	0.24	0.49	0.27	0.51	0.19	0.41	0.18	0.4	0.19	0.42	0.29	0.45
det aggregate dark	0.09	0.19	0.08	0.19	0.02	0.11	0.02	0.11	0.02	0.09	0.04	0.14
det aggregate light	0.1	0.39	0.07	0.39	0.06	0.31	0.06	0.31	0.06	0.28	0.02	0.39
det aggregate spherical	0.64	0.85	0.57	0.8	0.54	0.77	0.54	0.76	0.54	0.78	0.61	0.65
det feces like fish	0.16	0.33	0.15	0.33	0.13	0.25	0.13	0.26	0.14	0.25	0.14	0.23
det feces like munida	0.2	0.3	0.23	0.29	0.2	0.24	0.21	0.24	0.21	0.25	0.13	0.19
det feces stick	0.08	0.21	0.11	0.17	0.07	0.17	0.07	0.17	0.07	0.17	0.03	0.14
det fiber	0.16	0.26	0.19	0.31	0.22	0.3	0.22	0.29	0.23	0.31	0.29	0.27
phyto tricho puff	0.16	0.33	0.24	0.32	0.22	0.27	0.22	0.27	0.22	0.27	0.22	0.27
phyto tricho tuft	0.15	0.28	0.2	0.24	0.18	0.23	0.17	0.23	0.19	0.19	0.17	0.19
pro rhizaria other	0.07	0.22	0.0	0.22	0.0	0.06	0.0	0.06	0.0	0.07	0.08	0.2
pro rhizaria radiolaria collodaria solitary black	0.23	0.29	0.27	0.29	0.21	0.22	0.21	0.21	0.19	0.2	0.33	0.19
zoo crust cop	0.05	0.08	0.12	0.13	0.09	0.1	0.09	0.1	0.1	0.1	0.05	0.16
zoo crust cop like	0.06	0.1	0.07	0.1	0.05	0.09	0.05	0.09	0.06	0.09	0.03	0.08
zoo crust shrimp like	0.3	0.28	0.35	0.26	0.34	0.2	0.33	0.2	0.34	0.22	0.43	0.29
zoo gel carn medusa	0.08	0.13	0.14	0.21	0.11	0.16	0.11	0.16	0.07	0.15	0.11	0.13
zoo moll	0.2	0.51	0.32	0.46	0.27	0.34	0.27	0.34	0.28	0.36	0.31	0.38
avg	0.22	**0.32**	**0.24**	**0.32**	0.21	0.26	0.21	0.26	0.21	0.27	0.23	0.26

Table 6: UVP5 - F_1-scores of the classifiers presented in this section on the UVP5 dataset using MPEG-7 or shf18 features. The H^2SOM-Classifier(Rej) rejected 11.01% of validation data. A more detailed table additionally showing recall and precision can be found in the Appendix (Table 21 and Table 22).

quite low, as already seen in the Table 6. Here the fluctuations of the SVM can be observed as with the PAP dataset, albeit much less strongly.

Additionally, we compared the runtime performance of KNN, H^2SOL and SVM on the PAP dataset with a varying number of training and test samples. The results are shown in Figure 38. The training time (cmp. Figure 38a) for the SVM increases exponentially with the number of training samples. KNN does not build a model, thus no training is performed, but we rather measured the initialization time, e.g. for building a kd-tree. The setup of KNN, as well as, the training time of the H^2SOL are negligible compared to the SVM. The test time (cmp. Figure 38b) of SVM and KNN rises quickly with the amount of validation data to test. In comparison, the H^2SOL outperforms both other algorithms with the test time being one order of magnitude lower. The time for KNN and SVM is not only dependent on the amount of validation data, but also on the amount of training data. Because we split the dataset into training Γ^t and validation data Γ^v the amount of training data decreases, when the amount of validation data increases. Therefore it looks like the test times of SVM and KNN will run into a plateau.

Digression: Metagenome classification - AKE

Most of the following has been published in Langenkämper, Goesmann, and Nattkemper, 2014. Please note that the variables in this disgression are unrelated to the ones used before with the same symbol. The H^2SOM classifiers were originally developed for taxonomic classification in metagenomics. The resulting web-based metagenomic classifier AKE was published in Langenkämper, Goesmann, and Nattkemper, 2014. A short recap will be presented in the following. For creating a model, all full genomes from the NCBI database were gathered. For evaluation, the Acid

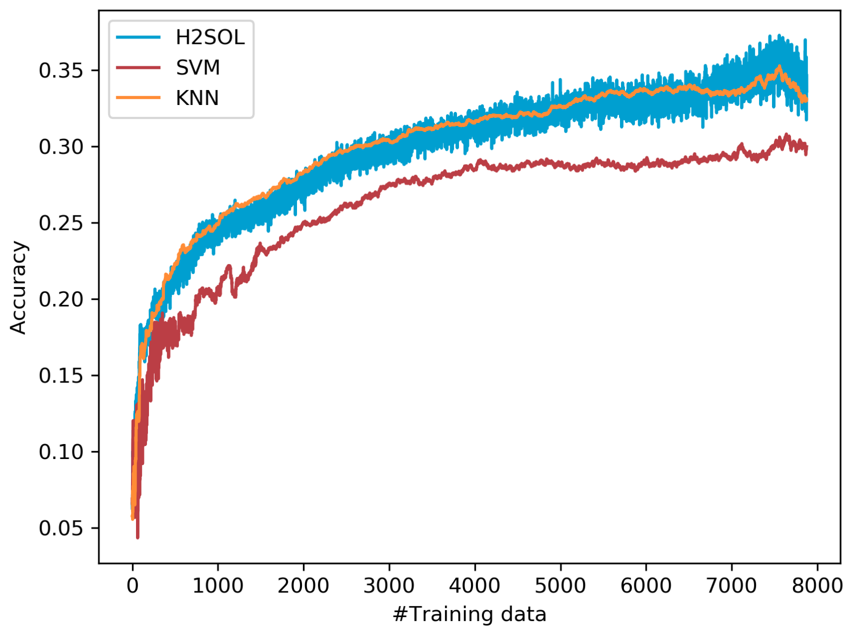
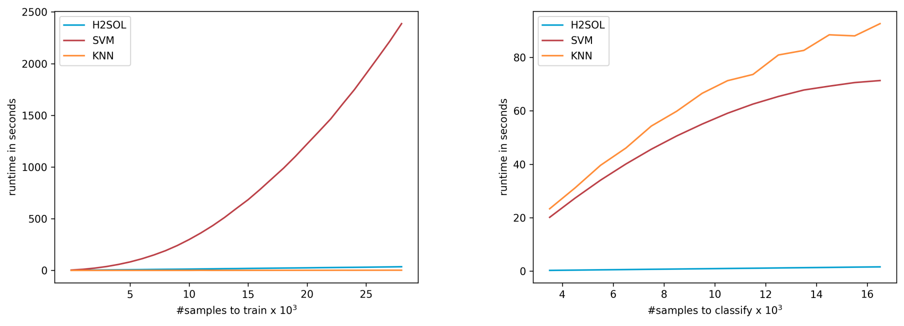


Figure 37: Performance of different classifiers on the UVP5 dataset using shf18 features The accuracy for the UVP5 dataset using shf18 features for different classifiers shows the impact of limited training data.



(a) Training time **(b)** Test time

Figure 38: Runtime comparison of KNN,H^{2}SOl and SVM learners on the PAP dataset *a)* shows the time for training of the algorithms. Please note that the KNN does not build a model, thus no training is performed, but we rather measured the initialization time, e.g. for building a kd-tree. *b)* shows the time for testing of the algorithms, i.e. classifying.

Mine Drainage (AMD) dataset [Tyson et al., 2004], as well as the Cow Rumen dataset, were chosen [Hess et al., 2011] To classify textual data like metagenomes, features have to be computed. For this purpose k-mer profiles with three different normalizations were used. A k-mer $\kappa_j(k, \Sigma)$ is a word of length k on an alphabet Σ. In this case $\Sigma = \{a, c, g, t\}$ is the DNA alphabet and therefore 4^k k-mers $\kappa_{j(j=0,\ldots,4^k-1)}(k, \Sigma)$ exist. A k-mer profile is the histogram of k-mers in a text. For more details on the design, feature extraction, classifier or visualization and online features have a look at the paper [Langenkämper, Goesmann, and Nattkemper, 2014]. The evaluation of different web-based taxonomic classifiers (see table below) shows that the runtime differs dramatically from a second (AKE), to minutes (PhylopythiaS), to an hour (NBC), to almost a week (WebCarma) due to algorithmic properties and implementation details. AKE is faster compared to the other applications because it only needs to compute the Euclidean distance between the descriptive model and the data that should be classified, whereas the others need to compute alignments (WebCarma) or apply decision functions (PhylopythiaS, NBC). The neural network used is especially suited to generate a hierarchical, compact, descriptive model, which allows fast queries. Although there might be methods reported to be equally fast and more accurate, to the authors' knowledge, there is no web-based solution that performs equally well in terms of execution time and accuracy for generic metagenome data. AKE is a fast taxonomic assignment tool for first visual inspection of whole metagenome datasets. Its web-based dynamic visualization allows fast analyses even on low performance computers without installation of software. Furthermore, the web-based approach enables a cooperative analysis of data with colleagues.

Algorithm	#Correct	#Wrong	#Rejections	Runtime (sec)
AKE	**902**	213	68	**0.43**
NBC	218	717	248	3480
PhylopythiaS	105	411	832	207
WebCarma	678	13	492	$6 * 10^5$

Table 7: Performance comparison of PhylopythiaS, NBC, WebCarma and AKE for the AMD dataset on phylum level.

3.5 Summary and list of contributions

In this section, we have established a pipeline for crafted computer vision systems. Various pre-processings, feature extractions, object detection systems and classifiers have been presented, discussed and evaluated. With the Unsharp Masking pre-processing, we could increase the classification accuracy by 8 basis points. Our newly proposed Super square features and Square histogram features have been shown to be competitive with established features in a test case, although they are slightly outperformed by the MPEG-7 CSD feature. We have shown that marine datasets can be classified automatically to a certain degree. The Hausgarten dataset can be classified satisfactory with an F_1-score of 0.87. The PAP dataset is more challenging, featuring more classes and with an F_1-score of 0.62 is a much harder task. Although the UVP5 dataset contains the same number of classes as the PAP dataset, the best classifier could only achieve an F_1-score of 0.32. Classification performance depends on a variety of factors. As our evaluations have shown, the number of classes and dataset size are possible influences as well as the complexity of the objects and the morphological similarity of the classes, i.e. similar-looking classes are likely to be confused by the classifier. Besides, we have investigated how much data is needed for each classifier to achieve satisfactory accuracy. Our experiments have shown that performance increases quite rapidly with an increasing number of training samples. Mediocre results can even be achieved after a fraction of the whole dataset is presented. A sufficiently large dataset is required to achieve good results. However, the number of samples depends strongly on the dataset. We have shown that the newly proposed H^{2}SOL and the newly proposed square histogram features are valuable assets in the automatic classification of marine datasets. The H^{2}SOL was the most accurate classifier in two out of three datasets. Although the H^{2}SOL was outperformed by the SVM for the Hausgarten dataset, training and classification times of the H^{2}SOL are much faster.

Contributions

Overview of pre-processing strategies

Different pre-processings relevant to marine image informatics were described and discussed.

Cross-validation study for pre-processing strategies
A cross-validation study was performed to measure the impact of some relevant pre-processings.

Overview of features
A variety of features relevant to marine image informatics were presented, as well as their applicability and usefulness for certain types of classes or image settings discussed.

Square histogram features
The square histogram features were proposed, with two histograms computed separately for the core and the remainder of an annotation patch.

Square median features/Super square features
The square median features and the super square features were proposed, where statistical measures on rings around the center are computed.

Cross-validation study for feature extraction
The feature extractions were evaluated on two different datasets using a cross-validation study, which showed considerable differences in applicability of features for the marine image domain. Furthermore, the newly proposed square histogram features and super square features proofed to be competitive for one of the two datasets.

Overview of object detection strategies
A quick overview of manual and computational methods for object detection was presented.

Entropy-driven ROI Detection
An automatic approach for ROI detection using Shannon entropy inspired by Johnson-Roberson, Pizarro, and S. Williams, 2010 was briefly sketched. The method was published in Langenkämper and Nattkemper, 2016.

Overview of classifiers
An overview describing and discussing the K-Nearest Neighbor, SVM, H^{2}SOL and H^{2}SOM classifiers was presented

H^{2}SOM classifiers
By introducing labeling of neural units in the H^{2}SOM algorithm and introduction of three labeling functions, we could transform the H^{2}SOM clustering algorithm to a supervised classifier. Parts of the methodology is published in Langenkämper, Goesmann, and Nattkemper, 2014.

H^{2}SOL
The H^{2}SOL classifier was proposed. It is a newly supervised classifier

based on the H^{2}SOM clustering algorithm. It uses local linear classifiers to improve the classification performance of the H^{2}SOM classifiers and furthermore provides a normalized output, i.e. belief values about the certainty of the classifier for its decision.

Cross-validation study for comparing classifiers

An extensive study comparing the presented classifiers on three different application scenarios was performed. The results were extensively discussed. The newly proposed H^{2}SOL and the square histogram features proofed to be a valuable asset for marine image informatics. Although the H^{2}SOL was only superior on two of three datasets regarding classification performance, the H^{2}SOl is orders of magnitude faster in training and one order of magnitude faster in classification.

How much data is needed?

An interesting question is that of how much annotated data samples are necessary to yield a certain classification accuracy. In experiments, where we increased the number of training data iteratively, we could systematically investigate that issue and analyze the advantages and disadvantages of the KNN, SVM and H^{2}SOL.

4

A deep learning approach to marine image analysis

Automatic classification systems are designed to solve problems, where humans are too slow to solve the problem in a reasonable time, or where humans are either unable to solve or are not particularly well fit to solve the problem. The latter means that there are a variety of problems that are so complex that the human mind is not particularly well suited to solve them. The paradox is that we are very capable of classifying objects, even if we have seen few representations of them. Nevertheless, the process behind this, i.e. the complete decision-making process is often beyond our reach. In particular, one could recognize that an object is probably a sea star and justify one's decision by reasoning that it resembles a sea star seen before, but this is a fairly high-level description of the neural process. From the visual input through our eyes, our brain extracts features and compares them with already known ones. But the process behind this, that is what features are computed and how the matching and reasoning is done is not transparent to us. Likewise, when solving such a problem using mathematics, the mathematical description of the visual image, i.e. a feature, is often high dimensional. The problem is that thinking in more than 2-3 dimensions while processing large amounts of data is hard or even impossible for us. Describing a perfectly fit feature set that captures only the most important information in arbitrary imagery, is therefore simply not possible for us by pure imagination.

One possible option to design features is to reenact nature, i.e. analyze how humans extract features and try to model this behavior using mathematics. Yet, this requires a complete understanding of the way we extract features, which is still quite limited today. Nevertheless, features based on Gabor filters are one such example. Another possibility would be to engineer features to describe the statistical values of the image data. However, this presupposes a modeling step that captures all aspects of the imagery in detail. This contradicts the hypothesis that our reasoning system is not transparent to us. Still, there are a variety of prominent engineered features, such as the MPEG-7 features or Gabor filters. In the crafted systems as presented above, some of these hand-selected and purpose-designed features were used. Before the advent of CNNs, handcrafted features were the prevalent feature extraction method. In contrast, CNNs use data-

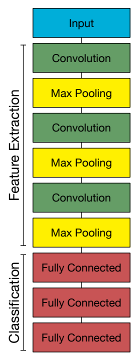


Figure 39: Depiction of a typical simple convolutional neural network. Multiple layers of convolution and pooling are followed by usually one or more fully connected layers. Feature extraction in CNNs is usually done by the convolutional filters, while the classification is usually done by the fully connected layers.

driven features learned from data, to model the same data to be discriminated by the subsequent classifier. Instead of dividing this process into parts, the data-driven feature extraction is integrated into the CNN. This avoids the manual optimization of the feature extraction and at the same time, enables the generation of more discriminative features. The disadvantage is that a lot of data is needed to learn from.

Convolutional Neural Networks (CNNs) have become popular in areas such as machine learning and computer vision due to the emergence of powerful graphics cards, huge datasets and software libraries supporting the training of CNNs. Although there are deep learning systems other than CNNs, this work focuses on CNNs due to their excellent suitability for image classification and detection.

In recent years, convolutional neural networks have completely changed the game of computer vision. Due to the great success in some of the important benchmark datasets as well as on real-world problems, convolutional neural networks are the prevalent contributions presented on machine learning or computer vision conferences and journals nowadays. Yet, it is still unclear whether this success can be transferred into the domain of marine image informatics. Usually, applications testing the CNNs against benchmark datasets or everyday images are presented and applications for special domains such as marine images are quite rare. Furthermore, marine images differ from everyday images in many respects.

- Compared to everyday images, the amount of data is still small.
- The classes in marine data are often imbalanced.
- The current can change the orientation of objects, thus arbitrary orientations might be possible.
- A limited number of views of one object are annotated, limiting the ability to classify similar objects, where photos were taken from a different angle.
- Lighting aberrations are much more severe, due to water specific light absorption and artificial lighting.
- Most research is focused on everyday objects, thus some assumptions in algorithms may not hold, e.g. that color is evenly distributed, while underwater images are often blueish.

In the following, we evaluate the use of CNNs mainly for classification, but also analyze interfaces and provide brief summaries for the application of CNNs for object detection and instance segmentation.

4.1 Feature extraction & classifiers revisited

For different applications, there is a multitude of different network architectures, the number of which is rapidly increasing. As part of this work, we have used several different Convolutional Neural Networks (CNNs) on a variety of datasets to estimate which network is best suited to what purpose in the context of marine imaging. The networks used are DenseNet-121 [G. Huang et al., 2017], Inception-V3 [Szegedy, Vanhoucke, et al., 2016], InceptionResNet-V2 [Szegedy, Ioffe, et al., 2017], MobileNet [Howard et al., 2017], NASNet [Zoph et al., 2017], ResNet50 [K. He, X. Zhang, et al., 2016], VGG16 [Simonyan and Zisserman, 2014] and Xception V1 [Chollet, 2017]. Please note that these are only a small selection of the published available networks. The selection was made for different reasons that are:

a Being fit for the problem, i.e. no exceptionally large models were chosen that would likely lead to overfitting.

b Availability and comparability, i.e. tested, reliable source code is available from one source.

c Pre-trained models for imagenet are available, i.e. these networks have been trained on imagenet and the respective model configurations, often simply called weights, are available for these models, which can be used to evaluate transfer learning.

d The models are either well-known or represent new innovative architectures.

VGG16 VGG16 [Simonyan and Zisserman, 2014] is one of the first deep convolutional networks. It exclusively consists of 3×3 convolutions, max pooling layers and fully connected layers (see Figure 40). It is regarded as a well researched, good baseline.

DenseNet-121 DenseNet-121 [G. Huang et al., 2017] consists of a repeating pattern of alternating dense blocks and transition blocks (see Figure 41).

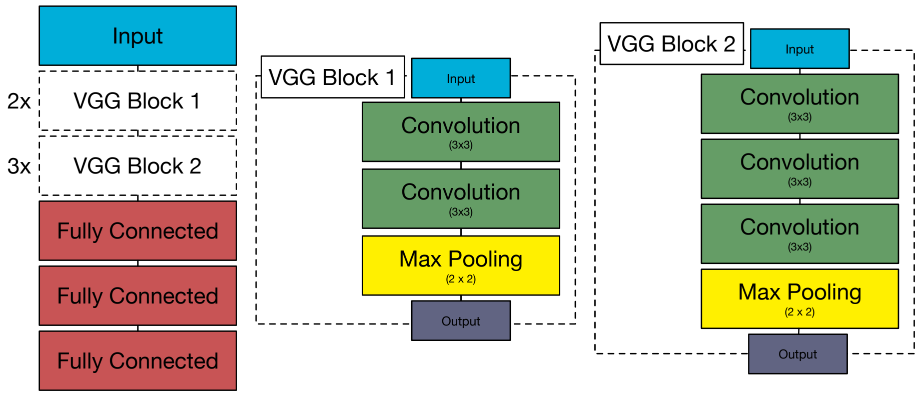


Figure 40: Architecture of VGG16 The architecture of VGG16 is dominated by two different repeating building blocks which only contain 3×3 convolutions and max poolings.

DenseNets are called like this because of the excessive use of bypass connections. After every dense block the output of that block is concatenated with the input of the same block, thus increasing the size of the input with each dense block. These dense connections give DenseNet-121 the name. The transition blocks reduce the size again by using 1×1 convolutions followed by an average pooling layer. 1×1 convolutions are a good way to either reduce or increase the dimensionality, i.e. the number of filters. Imagine you have an input of size $(\mathcal{B}, \mathcal{D}_0, \mathcal{H}, \mathcal{W})$, where $\mathcal{B}$ is the batch size, $\mathcal{D}$ is the dimensionality also known as depth, i.e. the number of filters and $\mathcal{H}$ and $\mathcal{W}$ are the height and width of the filter respectively. A 1×1 convolution with D_1 filters transforms the input

$$(\mathcal{B}, \mathcal{D}_0, \mathcal{H}, \mathcal{W}) \mapsto (\mathcal{B}, \mathcal{D}_1, \mathcal{H}, \mathcal{W})$$

The dense bypass connections allow for a strong gradient flow and better back-propagation of error. The concatenation of the bypass connection allows for more diversified features.

Inception V3 Inception V3 [Szegedy, Vanhoucke, et al., 2016] is the second increment of the original Inception architecture, also known as GoogleNet [Szegedy, Liu, et al., 2015]. Some refer to Szegedy, Liu, et al., 2015 as GoogLeNet and to Ioffe and Szegedy, 2015 as Inception V1, although Ioffe and Szegedy, 2015 also refers to the original Szegedy, Liu, et al., 2015 as Inception network. Therefore we will use the terms GoogLeNet and Inception V1 also for the Szegedy, Liu, et al., 2015 paper. Inception V3 is presented in the same paper as Inception V2 and includes several improvements such as the introduction of batch normalization in parts of the network, a better-fitted optimizer – the RMSProp optimizer – factorized 7×7 convolutions and

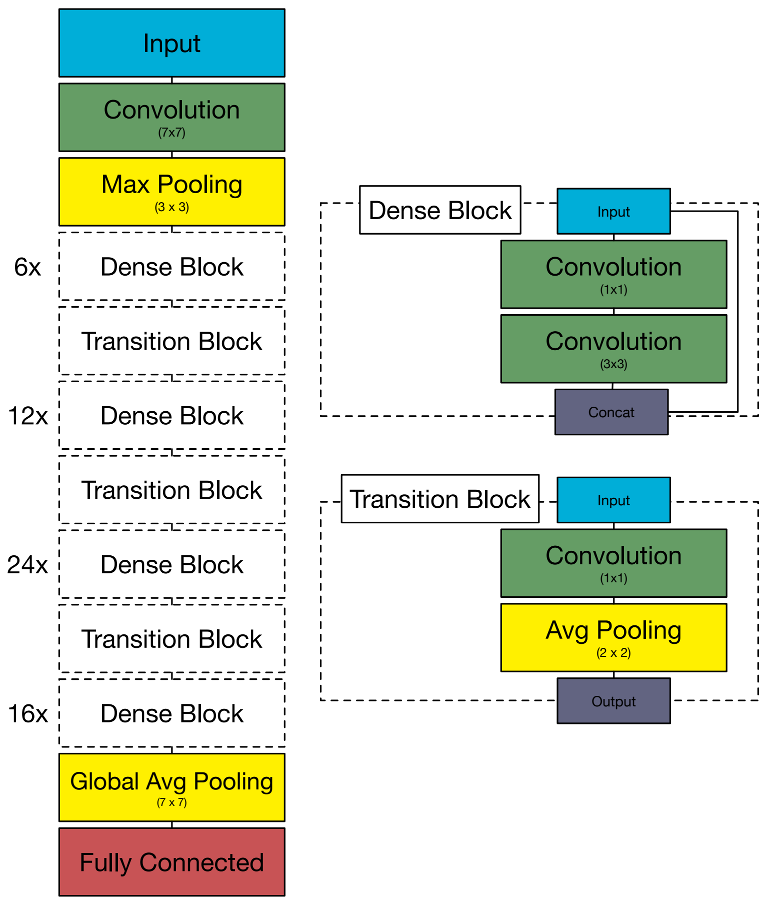


Figure 41: Architecture of DenseNet-121 DenseNet-121 consists of a repeating pattern of alternating dense blocks and transition blocks. 1×1 convolutions are used here as a means to increase or reduce the number of filters. Every layer in the network is preceded by a batch normalization which is not shown for brevity.

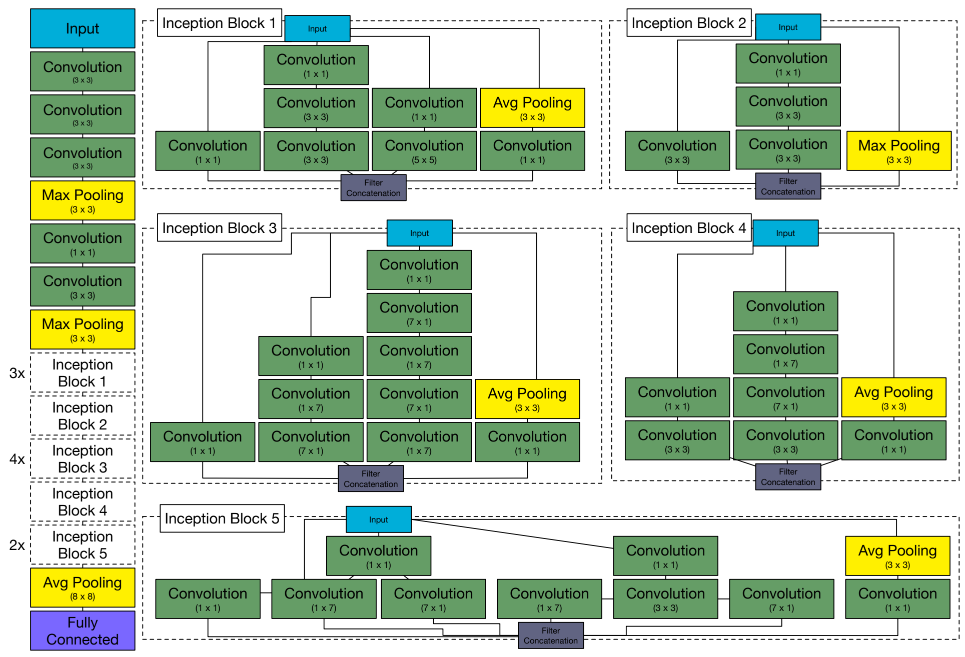


Figure 42: Architecture of Inception V3 Inception V3 is the incremental improvement of the Inception architecture. It features lots of batch normalizations as well as factorized 7×7 convolutions. All convolution layers are followed by a batch normalization, which is omitted due to brevity.

label-smoothing regularization. Batch normalization layers [Ioffe and Szegedy, 2015] normalize their input, i.e. each mini-batch is normalized by mean and variance of that batch. This has several advantages, e.g. it reduces the problem of exploding/vanishing gradients. In addition, it reduces covariance shifts and introduces a slight regularization. Details can be found in Ioffe and Szegedy, 2015. Factorizing convolutions with larger filter sizes into smaller ones, improves computational efficiency. A convolution of size 5×5 can be factorized into two 3×3 convolutions with the 5×5 convolution being $2.78\times$ more expensive [Szegedy, Vanhoucke, et al., 2016]. However, the spatial dependency of signals captured with a convolution is partly lost as spatially larger convolutions can encode dependencies, which are further away than spatially smaller ones. For more details on factorizing convolutions, RMSProp or label-smoothing regularization have a look at Szegedy, Vanhoucke, et al., 2016. The architecture is depicted in Figure 42. It consists of a so-called stem with convolutions, batch normalizations and max poolings followed by repetitions of five different inception blocks with different depths and widths, followed by an average pooling and a fully connected layer.

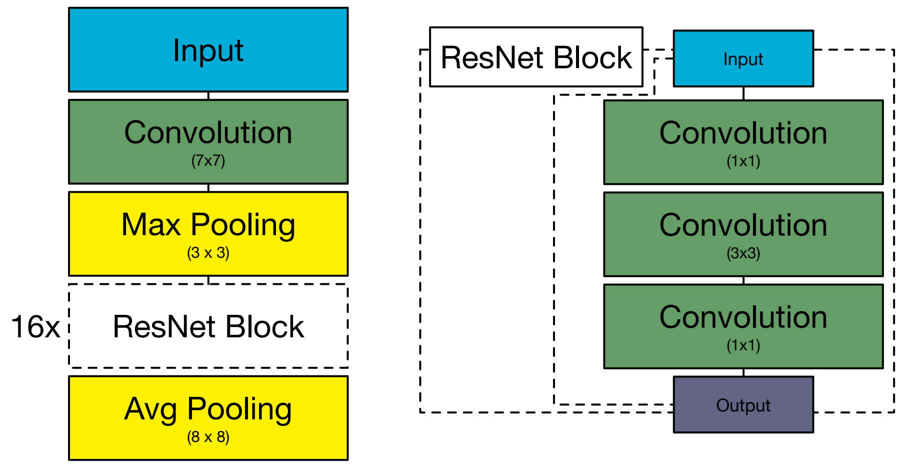


Figure 43: Architecture of ResNet50 With the introduction of residual connections in ResNet much deeper networks could be trained avoiding the vanishing or exploding gradients problem.

ResNet50 In recent years, models have become increasingly deeper to achieve a better classification performance. Unfortunately, the models are harder to train with increasing depth due to vanishing or exploding gradients [K. He, X. Zhang, et al., 2016]. In K. He, X. Zhang, et al., 2016 residual connections are introduced to compensate for this problem. This allows the training of significantly deeper models, which reach a depth of up to 152 layers and thus also increase the classification performance. For experiments, we used the smaller 50 layer ResNet50 introduced in the paper mentioned above. It contains 16 blocks each with a 1×1 convolution, which reduces the dimensionality, followed by a 3×3 convolution and again followed by a 1×1 convolution, which increases the dimensionality. The entire architecture is depicted in Figure 43.

InceptionResNet V2 InceptionResNet V2 is the fourth increment of the original Inception architecture and is introduced in the same paper as Inception V4 [Szegedy, Ioffe, et al., 2017]. The architecture is depicted in Figure 44 and is similar to previous inception variants, but introduces residual connections in the inception blocks. The authors claim that their system learns fasters and that it performs slightly better than a similar Inception architecture without residual connections.

MobileNet MobileNet [Howard et al., 2017] uses depthwise separable convolutions. These factorize standard convolutions into depthwise convolutions that apply a single filter to each input channel, and a 1×1 convolution combines the outputs of these depthwise convolutions. This drastically reduces computa-

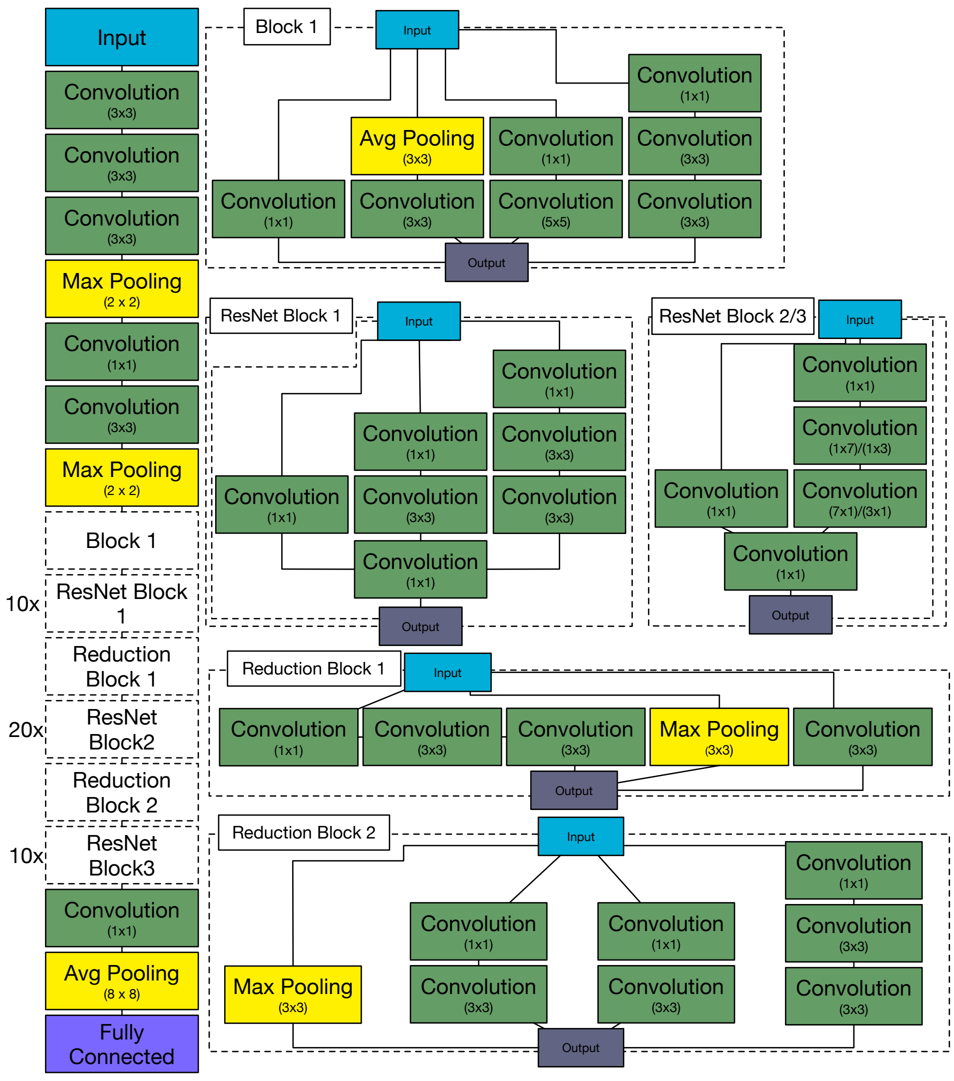


Figure 44: Architecture of InceptionResNet V2 InceptionResNet combines residual connections with inception blocks. All convolution layers are followed by a batch normalization, which is omitted due to brevity.

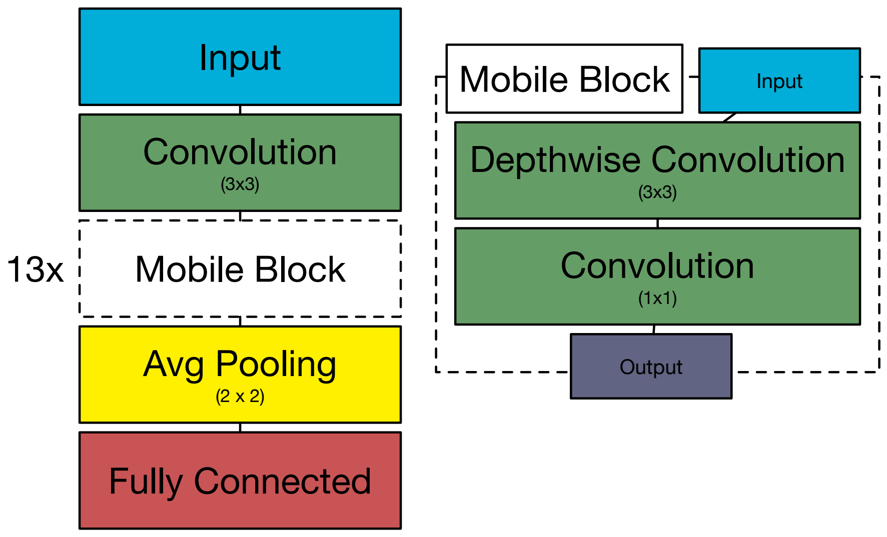


Figure 45: Architecture of MobileNet MobileNet is designed as a lightweight network, which can even be employed in mobile phones or single-board computers such as the NVIDIA Jetson.

tional costs and model size and is therefore often used for smaller scale models and devices, e.g. mobile phones. For a more detailed explanation and discussion of computational cost, see the original paper [Howard et al., 2017]. The architecture is rather simple as it consists of a 3×3 convolution followed by 12 blocks of depthwise separable convolutions, an average pooling and a fully connected layer (see Figure 45.

Xception V1 Xception V1 [Chollet, 2017] is introduced as a variation of the Inception V3 network, where the Inception modules have been replaced with depthwise separable convolutions. This slightly increases the classification performance and lowers computational cost. The architecture consists of two 3×3 convolutions each followed by a batch normalization, twelve Xception blocks, two 3×3 depthwise separable convolutions each followed by a batch normalization, a global average pooling and a fully connected layer (cmp. Figure 46).

NASnet In the NASNet paper [Zoph et al., 2017], the authors propose a method for automatically designing an architecture that matches the data one wants to classify. A recurrent neural network selects from a number of prominent CNN building blocks to create a network fit for the data. Furthermore, if this is too costly, measured in training time and data available, they present a methodology to scale up the architecture from a small training set to a larger one. They show this with scaling up from CIFAR-10 [Krizhevsky, G. Hinton,

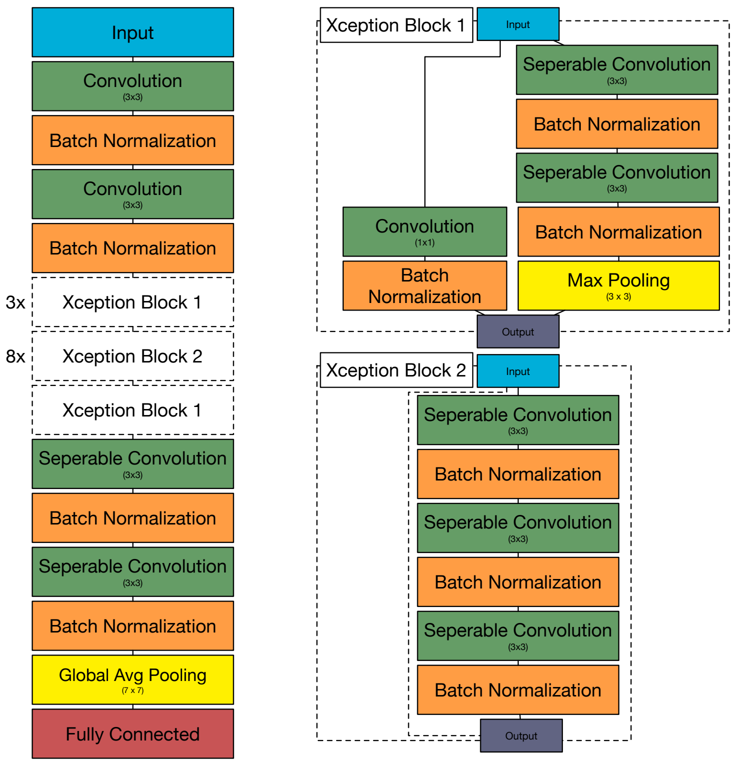


Figure 46: Architecture of Xception Xception introduces depthwise seperable convolutions into Inception V3 network.

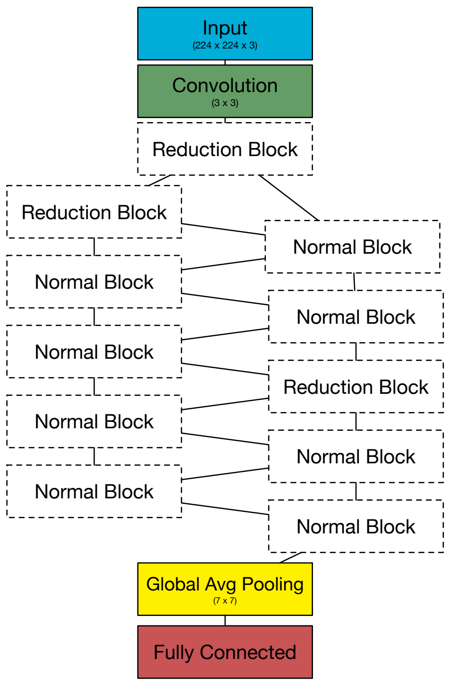


Figure 47: Architecture of NASNetMobile The architecture for the NASNetMobile convolutional neural network used in this thesis is depicted. In the NASNet paper the authors propose a method of automatically designing an architecture, therefore this structure is only one possibility. Due to brevity the structure of the reduction and normal blocks is not printed, but can be found in the original paper [Zoph et al., 2017].

et al., 2009], a dataset with ten different classes, to ImageNet, with 1000 classes. In our evaluation, a model automatically designed for ImageNet is used without algorithmically optimizing the architecture to fit our data. It consists of a 3×3 convolution followed by 3 reduction blocks and 3 normal blocks, using a particular link structure (cmp. Figure 47), preceded by a global average pooling layer and a fully connected layer.

Transfer learning Transfer learning is the process of taking a pre-trained network, i.e. a network that was trained on a huge dataset like ImageNet, to preset the weights and possibly adapt them further[Yosinski et al., 2014]. There are several different methods, all of which are coined transfer learning. We explore two of them.

The first method is to initialize the weights with pre-trained ones from a similar dataset or ImageNet and retrain all elements of the network, usually employing a lower learning rate. This makes it possible to initialize the weights with meaningful values instead of choosing random ones. This leads to faster convergence and usually to better results.

Another method would be to not only set the weights but to exclude multiple or even almost all layers except for the last ones from training. As a rule, the fully connected layers are trained, but the convolutional layers are excluded from training. Then the feature extraction of the convolutional layer is frozen and not learned anew, while the classification part of the fully connected layers is learned anew for each problem. By using pre-trained networks, the filters are already set to react to structural elements, color, or texture. It is assumed that on a fundamental level, the filters of two arbitrary networks are similar, e.g. most images feature edges or dominant colors. In addition, pre-training on a different dataset should increase the generalization performance since fewer parameters are trained and the network is not able to memorize the exact samples by heart. In addition, the learning speed of the network is increased significantly, as significant steps have already been done in pre-training. This assumes that the classifier creates meaningful, distinctive filters valid for both datasets, on a fundamental level. This may not be the case when very domain-specific data needs to be learned, e.g. the transfer from ImageNet to abstract paintings that do not resemble terrestrial objects. If there is still a slight similarity in the datasets, but it is not so pronounced, choosing the first option and initializing the weights with pre-trained data should be more promising. Please note that a lot more techniques to transfer from one model to another exists, but we limit our analysis to the two most common ones. For a current survey, have a look at C. Tan et al., 2018.

Data augmentation The term data augmentation describes the application of transformations to the training images to increase the amount of training data. These can be operations such as extracting sections from images, flipping, rotating, or Gaussian blurring. During training, not all the data is fed into the network in one piece, but instead, so-called batches are fed to the optimizer in order to incrementally learn from the data. During training, randomly selected transformations are applied to these batches to create volatile copies of them. Most deep learning frameworks are able to compute these augmentations on-the-fly on the current batch, i.e. the input annotation patches are transformed on the CPU while the GPU is still processing the previous batch. It has proven helpful in preventing overfitting and improving the performance of the classifier. To a certain extent, these transformations resemble pre-processing in the crafted networks. Instead of manipulating an entire image or all images, e.g. adjusting the illumination, only the annotation patches are manipulated. For a more in-depth overview have a look at Shorten and Khoshgoftaar, 2019.

Pre-processing Pre-processing was usually done to have more consistent data and therefore, also more valid training data. A condition that is often stated for deep learning is that enough training data is available. This implies that each class is presented under several conditions, e.g. even if the illumination conditions are poor, there is likely a sample in the dataset where the conditions are similarly poor. In theory, this makes most pre-processings superfluous. Therefore, apart from data augmentation, pre-processing is usually omitted. However, at the beginning of deep learning, when optimizers were not as stable as today, pre-processings to help the optimizers, e.g. whitening the images using ZCA (cmp. [Pal and Sudeep, 2016]) were widely used. Today this is also mostly omitted.

Experimental setup The aforementioned networks were evaluated on four different datasets, Hausgarten PS80/179-3, PAP, APEI6 and UVP5. These feature different challenges. The first three are benthic images with different image qualities, artifacts and visual appearances. UVP5 differs as it contains grayscale images of plankton that are reminiscent of light microscopy images. All of the above convolutional neural networks were evaluated on all of these datasets by splitting each dataset Γ in training Γ^t and validation data Γ^v with a ratio of 85% to 15%. Since no parameter tuning was performed, no optimization dataset was required and therefore more data was available for training and validation. Unlike the previous section, no cross-validation was carried out due to the rather long runtime of CNNs. For better comparability, the same split of training and validation data was used for each classifier, i.e. the training data was the same across the different classifiers. In addition to testing the plain classifiers, augmentation and transfer learning and a combination of both were evaluated for each classifier. The augmentation operations were flipping horizontally and 90, 180 and 270 degree rotation. For the transfer learning, presetting the weights and retraining of all layers were performed. We chose this way of transfer learning because in a pre-study (see appendix Table 23) it showed more consistent and significantly better results than retraining of the last layers only. Training was performed on an NVIDIA Tesla P100 for 15 epochs using Tensorflow 1.13.1 [Abadi et al., 2016].

Results In Table 8 the results for the Hausgarten PS80/179-3 for all classifiers including transfer learning (T), augmentation (A) and both (T)(A) are listed. Xception with transfer learning with 0.99 validation accuracy delivers the best results. The runner-ups are ResNet with transfer learning with 0.98, Xception with transfer learning and augmentation with 0.97 and InceptionResNet with transfer learning with 0.96. Interestingly, transfer learning with added augmentation seems to reduce the performance of the Xception network. The observation that augmentation causes a loss of performance can generally be observed with this dataset, except for VGG16 and NASNet. Both also have significantly lower validation accuracy than the competitors. Probably, VGG16 and NASNet need more data to function properly. Augmentation would provide this additional data and thus improve their performance. This is probably be-

Network	validation accuracy	training accuracy	training time	#parameters*10^6
DenseNet	0.74	0.88	251.37	7
DenseNet(A)	0.45	0.60	255.05	7
DenseNet(T)	0.92	0.98	249.21	7
DenseNet(T)(A)	0.88	0.85	254.57	7
Inception	0.79	0.86	228.85	22
Inception(A)	0.57	0.57	234.64	22
Inception(T)	0.92	0.98	287.14	22
Inception(T)(A)	0.67	0.79	293.65	22
InceptionResNet	0.81	0.91	680.67	54
InceptionResNet(A)	0.68	0.69	687.98	54
InceptionResNet(T)	0.96	0.99	679.25	54
InceptionResNet(T)(A)	0.90	0.92	686.52	54
MobileNet	0.79	0.95	**202.74**	**3**
MobileNet(A)	0.62	0.74	208.52	**3**
MobileNet(T)	0.95	0.99	203.61	**3**
MobileNet(T)(A)	0.94	0.94	207.88	**3**
NASNetMobile	0.27	0.94	526.39	4
NASNetMobile(A)	0.27	0.75	544.60	4
NASNetMobile(T)	0.26	0.99	530.67	4
NASNetMobile(T)(A)	0.30	0.95	535.93	4
ResNet	0.51	0.83	217.79	24
ResNet(A)	0.48	0.62	221.89	24
ResNet(T)	0.98	0.99	218.62	24
ResNet(T)(A)	0.60	0.92	221.89	24
VGG16	0.14	0.17	263.12	134
VGG16(A)	0.34	0.35	267.16	134
VGG16(T)	0.34	0.30	229.31	134
VGG16(T)(A)	0.34	0.35	233.35	134
Xception	0.86	0.95	732.62	21
Xception(A)	0.66	0.72	737.29	21
Xception(T)	**0.99**	**1.00**	732.14	21
Xception(T)(A)	0.97	0.94	737.28	21

Table 8: Comparison of different CNNs on the Hausgarten dataset. (A) stands for added augmentation, (T) for application of transfer learning and (T)(A) is the combination of both.

cause Hausgarten PS80/179-3 is a very small dataset that most classifiers can memorize. The augmentation introduces variation, which hampers the ability of the classifier to learn the data by heart. In addition, values of 0.99 for the validation accuracy for such a small dataset with a training accuracy of 1.00 also indicate that overfitting may have occurred.

The PAP dataset is a larger dataset with more classes. In Table 9 the results are displayed. MobileNet with transfer learning and augmentation, as well as InceptionResNet with transfer learning, achieve the highest accuracy of 0.96. The runner-ups all with 0.95 validation accuracy are DenseNet with transfer learning, DenseNet with transfer learning and augmentation, Inception with transfer learning, InceptionResNet with transfer learning and augmentation, MobileNet with transfer learning, ResNet with transfer learning, Xception with transfer learning and Xception with transfer learning and augmentation. Here the addition of augmentation does not seem to affect the performance at all or only marginally. All networks achieve almost equal performance of above 90% with a few exceptions with slightly lower accuracy, except VGG16, which performs significantly worse with an accuracy of 0.19 to 0.30. Although this network achieved state of the art performance at its time on ImageNet, which is significantly larger than PAP and has more classes, the community reports that it is known to be difficult to train with some datasets. Here it is likely that the PAP dataset is still too small.

The APEI6 dataset is similar to PAP in visual quality and size. Yet, it has slightly fewer classes. In Table 10 the results are listed. Here, several algorithms perform equally well with 0.95 validation accuracy. These are DenseNet with transfer learning, Inception with transfer learning, InceptionResNet with transfer learning, MobileNet with transfer learning, Mobilenet with transfer learning and added augmentation and Xception with transfer learning. Interestingly, the runner-ups for this dataset include networks trained from scratch, e.g. DenseNet and InceptionResNet each with 0.94. In addition, VGG16 performs almost competitively to other algorithms. All algorithms perform better than 0.86, which is remarkable.

The UVP5 dataset is different from the other datasets in that it does not contain benthic images, but light microscopy-like ones. In Table 11 all the results are displayed. MobileNet with transfer learning is the best-performing network here. Runner-ups are InceptionResNet with transfer learning with 0.87, and Inception with transfer learning, InceptionResNet with transfer learning and augmentation, MobileNet with transfer and augmentation, Xception with transfer learning and Xception with transfer learning and augmentation each with 0.86 accuracy. Some networks significantly underperform with this dataset reaching below 0.20 validation accuracy. Interestingly the only pattern observable is that transfer learning seems to increase performance except for VGG16. The validation accuracy of DenseNet drops down to 0.12 from 0.47 when augmentation is added. In contrast, ResNet without augmentation only has an accuracy of 0.19, but with augmentation of 0.64. One explanation could be that the augmentation increases features, which help the ResNet to classifier objects, but at the same time hampers the performance of DenseNet, which probably focuses on other features.

In Figure 48 we plotted the number of parameters against the validation accuracy (for results with added augmentation/transfer learning have a look at the appendix Figure 79, Figure 80, Figure 81). Although the number of parameters may limit the accuracy of a classifier, this does not seem to be the case here. VGG16, which has by far the most trainable parameters, is one of the least performant classifiers. Therefore, the architecture and interconnection of the parameters seem to be of major importance. This is in accordance with other works, e.g. Canziani, Paszke, and Culurciello, 2016. The results for the other classifiers do not indicate a relationship between the number of parameters

Network	validation accuracy	training accuracy	training time	#parameters
DenseNet	0.92	0.96	2971.10	7
DenseNet(A)	0.91	0.87	3005.90	7
DenseNet(T)	0.95	0.98	2991.26	7
DenseNet(T)(A)	0.95	0.93	3025.00	7
Inception	0.91	0.97	2728.95	22
Inception(A)	0.86	0.88	2793.43	22
Inception(T)	0.95	0.98	3437.51	22
Inception(T)(A)	0.94	0.93	3509.09	22
InceptionResNet	0.92	0.98	8137.58	54
InceptionResNet(A)	0.86	0.91	8219.23	54
InceptionResNet(T)	**0.96**	**0.99**	8128.95	54
InceptionResNet(T)(A)	0.95	0.95	8220.21	54
MobileNet	0.92	0.98	**2439.42**	**3**
MobileNet(A)	0.93	0.93	2487.02	**3**
MobileNet(T)	0.95	0.98	2456.92	**3**
MobileNet(T)(A)	**0.96**	0.95	2498.98	**3**
NASNetMobile	0.82	0.98	6266.81	4
NASNetMobile(A)	0.81	0.92	6398.94	4
NASNetMobile(T)	0.92	0.98	6350.39	4
NASNetMobile(T)(A)	0.80	0.94	6386.67	4
ResNet	0.91	0.94	2667.24	24
ResNet(A)	0.86	0.86	2663.85	24
ResNet(T)	0.95	**0.99**	2605.01	24
ResNet(T)(A)	0.90	0.93	2646.99	24
VGG16	0.30	0.29	3111.21	134
VGG16(A)	0.19	0.19	3162.21	134
VGG16(T)	0.30	0.29	2758.59	134
VGG16(T)(A)	0.30	0.31	2800.80	134
Xception	0.93	0.98	8829.13	21
Xception(A)	0.92	0.93	8886.17	21
Xception(T)	0.95	0.98	8818.83	21
Xception(T)(A)	0.95	0.95	8900.23	21

Table 9: Comparison of different CNNs on the PAP dataset. (A) stands for added augmentation, (T) for application of transfer learning and (T)(A) is the combination of both.

Network	validation accuracy	training accuracy	training time	#parameters
DenseNet	0.94	0.96	5229.59	7
DenseNet(A)	0.91	0.90	5317.64	7
DenseNet(T)	**0.95**	0.98	5223.09	7
DenseNet(T)(A)	0.91	0.90	5307.52	7
Inception	0.93	0.95	4779.70	22
Inception(A)	0.89	0.89	4895.60	22
Inception(T)	**0.95**	0.99	6016.45	22
Inception(T)(A)	0.89	0.89	6149.47	22
InceptionResNet	0.94	0.97	14252.27	54
InceptionResNet(A)	0.88	0.89	14305.63	54
InceptionResNet(T)	**0.95**	0.99	14267.52	54
InceptionResNet(T)(A)	0.94	0.94	14361.69	54
MobileNet	0.93	0.99	**4274.73**	**3**
MobileNet(A)	0.92	0.92	4353.78	**3**
MobileNet(T)	**0.95**	0.99	4293.50	**3**
MobileNet(T)(A)	**0.95**	0.94	4412.02	**3**
NASNetMobile	0.92	0.97	11141.63	4
NASNetMobile(A)	0.90	0.91	11337.17	4
NASNetMobile(T)	0.94	0.99	10993.07	4
NASNetMobile(T)(A)	0.90	0.94	11241.13	4
ResNet	0.92	0.96	4549.94	24
ResNet(A)	0.89	0.89	4648.15	24
ResNet(T)	0.93	0.99	4552.95	24
ResNet(T)(A)	0.90	0.92	4625.12	24
VGG16	0.86	0.86	5443.25	134
VGG16(A)	0.86	0.86	5526.50	134
VGG16(T)	0.86	0.86	4833.23	134
VGG16(T)(A)	0.86	0.86	4908.00	134
Xception	0.93	0.99	15455.71	21
Xception(A)	0.90	0.90	15575.38	21
Xception(T)	**0.95**	0.99	15492.28	21
Xception(T)(A)	0.94	0.94	15614.80	21

Table 10: Comparison of different CNNs on the APEI6 dataset. (A) stands for added augmentation, (T) for application of transfer learning and (T)(A) is the combination of both.

Network	validation accuracy	training accuracy	training time	#parameters
DenseNet	0.47	0.93	4006.61	7
DenseNet(A)	0.12	0.76	4103.40	7
DenseNet(T)	0.82	0.96	4015.40	7
DenseNet(T)(A)	0.79	0.82	4049.59	7
Inception	0.77	0.92	3682.74	22
Inception(A)	0.69	0.79	3755.54	22
Inception(T)	0.86	0.97	4626.80	22
Inception(T)(A)	0.83	0.84	4726.76	22
InceptionResNet	0.81	0.96	10946.40	54
InceptionResNet(A)	0.67	0.80	11055.92	54
InceptionResNet(T)	0.87	**0.98**	10953.63	54
InceptionResNet(T)(A)	0.86	0.87	11053.49	54
MobileNet	0.78	0.94	3299.50	**3**
MobileNet(A)	0.79	0.81	3355.56	**3**
MobileNet(T)	**0.88**	0.97	**3286.15**	**3**
MobileNet(T)(A)	0.86	0.86	3343.39	**3**
NASNetMobile	0.12	0.93	8512.89	4
NASNetMobile(A)	0.09	0.80	8693.17	4
NASNetMobile(T)	0.31	0.97	8520.24	4
NASNetMobile(T)(A)	0.25	0.85	8565.60	4
ResNet	0.19	0.95	3504.24	24
ResNet(A)	0.64	0.77	3582.54	24
ResNet(T)	0.77	**0.98**	3504.90	24
ResNet(T)(A)	0.79	0.85	3571.08	24
VGG16	0.12	0.12	4185.71	134
VGG16(A)	0.08	0.09	4252.73	134
VGG16(T)	0.12	0.12	3710.85	134
VGG16(T)(A)	0.12	0.14	3771.58	134
Xception	0.78	0.97	11896.78	21
Xception(A)	0.83	0.82	12024.57	21
Xception(T)	0.86	**0.98**	11908.81	21
Xception(T)(A)	0.86	0.86	12015.85	21

Table 11: Comparison of different CNNs on the UVP dataset. (A) stands for added augmentation, (T) for application of transfer learning and (T)(A) is the combination of both.

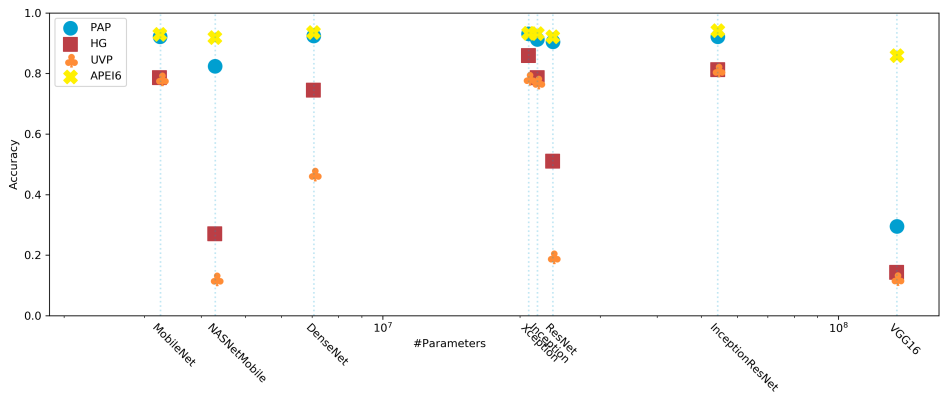


Figure 48: Validation accuracy and number of parameters of various deep learning classifiers For results of added transfer and augmentation please have a look at the appendix Figure 79, Figure 80, Figure 81.

and the validation accuracy. In contrast, MobileNet with the smallest number of parameters and InceptionResNet with the largest number of parameters except for VGG16 often perform better than their competitors.

When we compare the influence of transfer learning, augmentation, and the combination of both on the classification performance, we see that transfer learning provides the best results. In our setting augmentation often negatively affects the performance (see Figure 49, please note that only results for APEI6 are displayed). In five out of eight cases, the baseline even surpasses the variant with additional augmentation or transfer learning and additional augmentation. A curiosity is again VGG16, which performs equally well regardless of the additional technique used.

The time to train a network given the same amount of learning epochs and the same amount of training data should also be mainly affected by the number of parameters and computational efficiency of the operations used. As an example, the results of APEI6 are discussed, but the results for the other datasets are comparable. Most algorithms finish training in about 1.5 hours, except NASNetMobile, InceptionResNet and Xception, which take 3 to 4 hours of training (see Figure 50). InceptionResNet is computationally expensive because it contains many parameters. Furthermore, the input size is 299×299 instead of 229×229, as most other tested networks expect. NASNetMobile features some 7×7 convolutions, which are computationally demanding. Apart from that it has almost the least amount of parameters and uses depthwise separable convolutions, which are computationally less expensive than regular convolutions. It may be possible that the interconnections between the modules result in a bad layout of operations on the GPU, although this is only an educated guess. Xception makes heavy use of Batch Normalization, which are costly and the input size is 299×299, which is larger than most networks. It uses depthwise

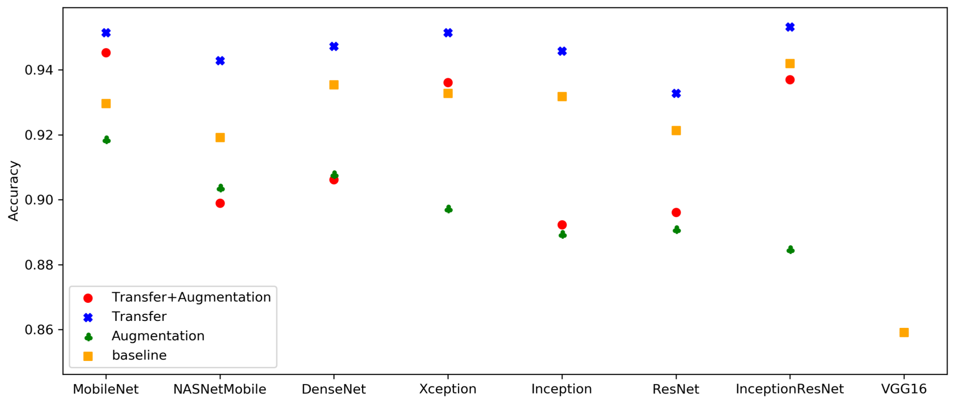


Figure 49: Transfer learning and augmentation Influence of transfer learning and augmentation on the accuracy of different deep learning classifiers trained on the APEI6 dataset.

separable convolutions, which should speed up the training. However, all of this does not explain why Xception is the slowest of all networks in training. The idea that depthwise separable convolutions are not implemented as well as regular convolutions can be discarded in our opinion, due to the fact that MobileNet, which makes heavy use of them is the fastest network. When comparing the runtimes of a network with added augmentation or transfer learning, we expect the variants with augmentation to be somewhat slower as they increase the number of training samples and thus the number of iterations per epoch. Except for VGG16, this holds true.

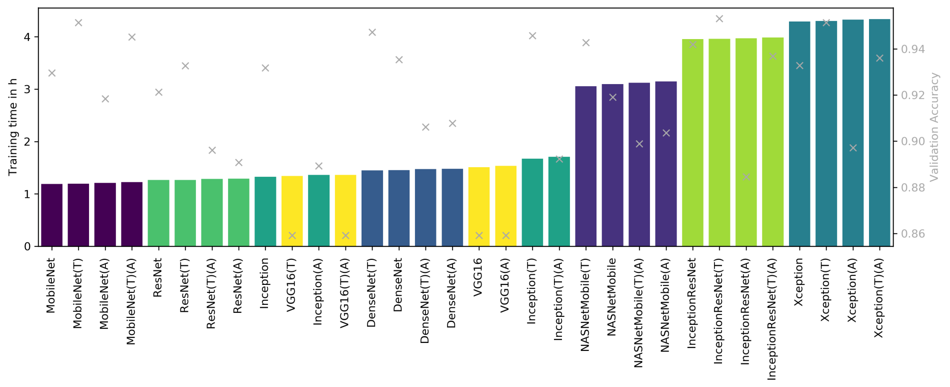
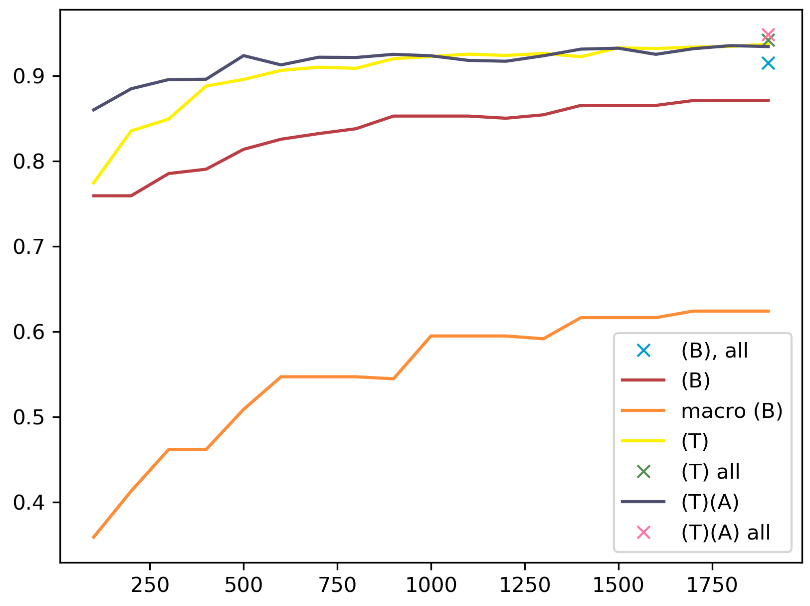


Figure 50: Training time and validation accuracy of various deep learning classifiers The boxes depict the training time and the gray x's show the accuracy.

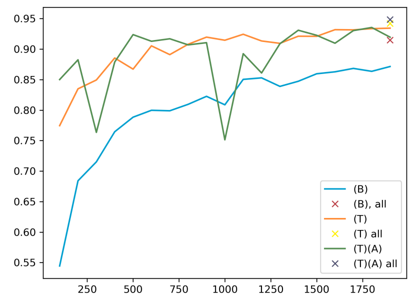
4.2 How much data is necessary for applying convolutional neural networks?

As shown above, CNNs provide superior performance, with their data-driven features, in contrast to the conventional crafted systems, as presented in the previous section (cmp. Section 3). Yet, CNNs are known to require a lot of data to achieve this kind of superior performance and therefore, cannot be employed for all problems. But how much training data is sufficient? To answer this question, we have divided the PAP dataset into training and validation datasets with varying sizes. For $J' \in \{200, 300, 400, \ldots, 2000\}$ let $\Gamma^t_{J'} = \{x_0, x_1, \ldots, x_{J'}\}$ be the training set for each experiment with a growing number of training samples. As validation set we always use the same validation set Γ^v disjoint from $\Gamma^t_{J'}$, $\Gamma^t_{J'} \cap \Gamma^v = \emptyset$ containing 15% of all samples of the PAP dataset. With each $\Gamma^t_{J'}$ a DenseNet121 was trained and evaluated against Γ^v. In addition to the use of the plain DenseNet121 classifier, the impacts of transfer learning and transfer learning with added augmentation on training with limited data were evaluated. PAP is a rather imbalanced dataset. This could lead to the classifier favoring one class when limited training data is available, i.e. the classifier, if not trained with enough data, may be tempted to always predict the majority class to achieve a fairly good classification performance. In order to assess this scenario, the macro-averaged accuracy is determined, i.e. the per-class accuracies are averaged. For each training set, the classifier was trained for 15 epochs. In Figure 51a you can see the best accuracy of all 15 epochs, no matter in which epoch it was produced. In a real-world case, where the validation data is unlabeled and unknown, this is, of course, not possible, but here it provides an estimate of the best possible case. Results for the accuracies yielded in the last epoch (cmp. Figure 51c), or for the epoch with the best training loss (cmp. Figure 51b, both pieces of information can be obtained, without knowing the validation data) are also shown below.

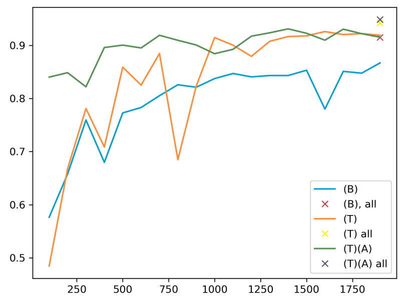
In Figure 51a we see that quite good results can be achieved, even with a very limited amount of training data. A stable plateau phase seems to be reached



(a)



(b)



(c)

Figure 51: Available Training data and accuracy Impact of the amount of training data on the validation accuracy. *a)* The epoch with the best accuracy is selected for visualization. *b)* The epoch with the least loss is selected for visualization. *c)* The accuracy yielded after the last epoch is shown. (B) stands for baseline and is without augmentation or transfer learning. (T) stands for transfer learning (A) for augmentation. (T)(A) is the combination of both. Macro (B) denotes the macro-averaged accuracy for the baseline experiment. The suffix "all" denotes that these accuracies are achieved when all available training is provided.

after 1500 training samples. Nevertheless, the provision of all available training data Γ^t shows a significant improvement. The addition of transfer learning and even more transfer learning with augmentation leads to better results, even with very few training samples. The results of the macro-average of the DenseNet classifier without transfer learning and augmentation (see Figure 51a macro (B)) suggest that some classes are not well predicted. This is no wonder when an average of 10 samples per class is available for 200 samples and 19 classes. If we consider the class imbalance, some classes may not be included at all in the training or validation data. The training and validation data are randomly sampled. Therefore not all classes are represented evenly. When looking at the more realistic results with the best loss during an experiment (Figure 51b) or the results after the last epoch (Figure 51c), we see that the results are not continuously rising as before, but fluctuate and only settle after about 1000 labeled training data. This proves that with limited training data, the results can theoretically be quite good and stable, as the best case evaluation shows, but these results might be misleading in a practical use case since no one can say which epoch delivers the best results beforehand.

4.3 Does size matter? Encoding annotation size in convolutional neural networks

It is natural that objects are not equally sized. An annotation keeps track of this by incorporating a size r in a square annotation $\mathcal{A}'_j$, but the input of a convolutional neural network is usually the same size, e.g. 229×229. There are multiple ways to get an annotation patch $\mathbb{I}_j$ to the desired size, suitable for processing with a CNN. Figure 52 illustrates several ways to manipulate an annotation patch $\mathbb{I}_j$ to have the desired size. The methods of Figure 52 are described in the text below, but not all possible cases are covered. Please refer to the figure then.

The usual way is to scale the patch to the desired size using interpolation (cmp. Figure 52 a)). A potential drawback is that the size information is not preserved after scaling, i.e. all objects are equally sized. Yet, it possibly might not matter to get rid of the size information at all. Another simple alternative that does retain the size information is to extract a patch of the desired size around the center of the annotation (cmp. Figure 52 b)). This has two main disadvantages. If the annotation is larger than the desired size, only a partial object is extracted. In addition, when an object is quite small, a lot of background with other objects or misleading information might be extracted. Another way to preserve size is to use padding, i.e. the object is scaled to the desired size if it is larger or equal than the desired size, or otherwise if it is smaller than the desired size, a padding, e.g. noise is added to fill in the remaining space (cmp. Figure 52 c)). The padding should mask the surrounding environment of the object. Nevertheless, it is unclear whether the classifiers have problems dealing with this artificial artifacts. Several different padding variants are possible, e.g. random noise, padding with a constant value, e.g. black, white or the image mean, or padding by mirroring the image. An alternative to padding to the desired size is to scale the image to half the desired size first and then pad the rest (cmp. Figure 52 d)).

In an experiment we compared different strategies to answer the question

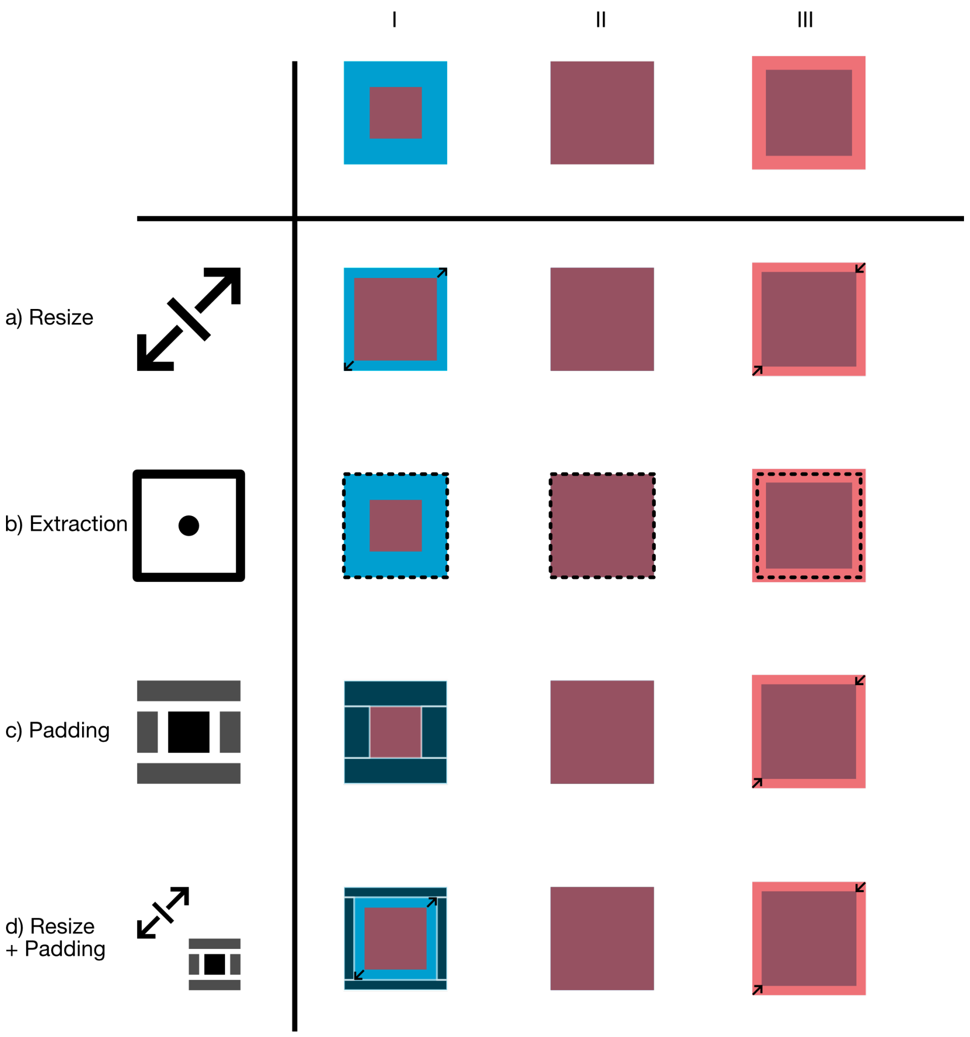


Figure 52: Different possibilities to get equally sized input for CNNs The blue square is the desired size, whereas the red square is the current size of the patch. The column I) illustrates what is done with patches which are smaller than the desired size. II) the patches are exactly the desired size. III) the patches are larger than the desired size. a) Resize: the patches are resized to the desired size. b) Extraction: a patch of the desired size around the center of the annotation is extracted. c) Padding: The patch is padded to get the desired size. d) Resize + Padding: The patch is resized to half of the desired size and then padded.

whether encoding size is beneficial for the classification performance and which method for scaling the images to the desired size is superior. With the annotations of the APEI6 images we created seven different datasets Γ_{resize}, Γ_{constant}, Γ_{noise}, Γ_{ambient}, Γ_{ramp}, Γ_{zero}, $\Gamma_{\text{padresize}}$ for the different methods to generate an annotation patch $\mathbb{I}_j$ from a square annotation $\mathcal{A}'_j$. Γ_{resize} refers to Figure 52 a), i.e. resizing the annotation patch $\mathbb{I}_j$ to the desired size, getting rid of the size information. Γ_{constant} refers to Figure 52 b), i.e. extracting an annotation patch of a constant size around the center of the annotation $\mathcal{A}'_j$. $\Gamma_{\text{noise}}, \Gamma_{\text{ambient}}, \Gamma_{\text{ramp}}, \Gamma_{\text{zero}}$ refer to Figure 52 c), using random noise (Γ_{noise}), using the mean of the edge pixels along one direction (Γ_{ambient}), using a linear ramp from the edge value to zero (Γ_{ramp}) or padding with zeros (Γ_{zero}). $\Gamma_{\text{padresize}}$ refers to Figure 52 d), i.e. first resizing and then using padding with random noise. Since each of these seven datasets is processed the same way, we describe the following process only once and refer to the dataset with just Γ for brevity. Γ is divided into a training set Γ^t and a validation set Γ^v with a ratio of 0.85 to 0.15. A MobileNet (see Section 4.1) is trained on Γ^t and evaluated on Γ^v. This experiment is repeated six times.

The results are displayed in Figure 53. While resize, noise and padresize perform almost equally well, constant is significantly lower in performance. This suggests that size information, at least for this dataset, may not be of great importance. Datasets where objects look the same and can only be distinguished by their size, e.g. diatoms, should benefit more from the inclusion of size information. For incorporating size information, padding with noise or padding and resizing appear to be equally suitable, while constant should be avoided. Using the constant method for small objects potentially extracts too much distracting information, such as parts of other objects, resulting in lower performance. The comparison of different padding variants (noise, ambient, ramp and zero) shows that noise and zero perform equally well concerning the mean value of the validation accuracy of the experimental runs. However, the results of zero are not that stable, i.e. fluctuate more, thus noise seems to work a bit better. Ramp and ambient achieve worse results with a lot more fluctuation.

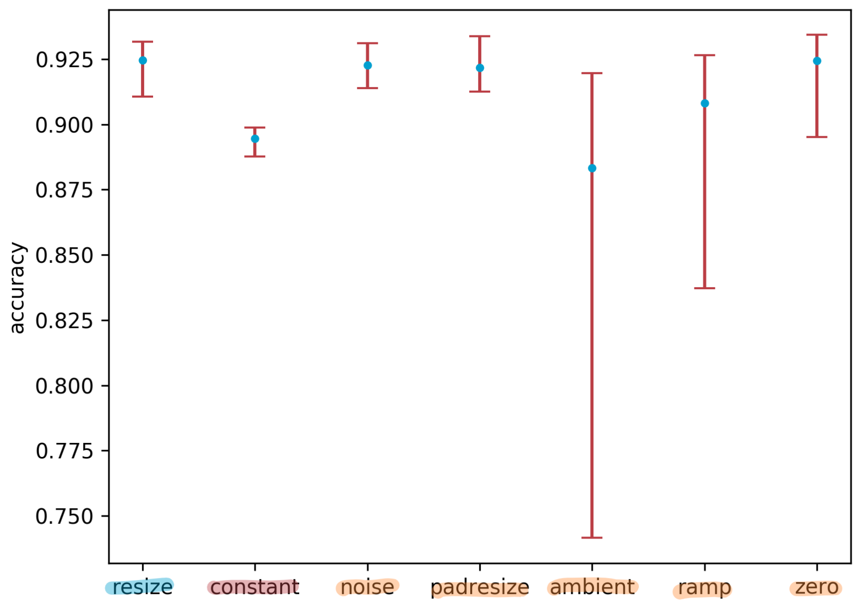


Figure 53: Classification accuracy for the different extraction methods Classification accuracy of a MobileNet on datasets reflecting different strategies to incorporate size information and one dataset (resize), which is void of size information. The blue dots represent the mean value of all six experimental runs and the red lines represent the range of results of the six experimental runs. The blue-marked dataset is void of size of information, while the red- and orange-marked ones incorporate it. The orange-marked ones use different padding variants.

4.4 Data scarcity and the data imbalance problem

Most of the following results have been published in Langenkämper, Kevelaer, and Nattkemper, 2019. They are based on the bachelor thesis of Robin van Kevelaer supervised by me.

Biological data, and in particular marine image data, has two problems. Data is scarce and certain classes are far more abundant than others. We refer to the first problem with the term *the data scarcity problem*. The data scarcity problem can be described as having too little data for a specific task to be fulfilled satisfactorily. From the definition, it can be deduced that how little data is too little depends on the task to be fulfilled and the questions to be answered. In our special case of training a classification or detection algorithm, the data scarcity problem can be further specified in more detail in the following: The data scarcity problem is present in a case when having more data would lead to a significant increase in classification/detection performance. Dependent on the algorithm, this threshold is rather high or low, e.g. a KNN needs very little data and usually runs into a plateau phase regarding classification performance rather quickly and therefore, a data scarcity problem is somewhat rare, while a CNN can benefit from a lot of training data. Marine images are also a special type of biological data. Data is scarce due to the high investment in ship cruises, equipment and complicated setup of the imaging system needed to acquire underwater imagery. The annotation problem is exacerbated by the high diversity - low abundance phenomena observed in the deep-sea. In addition, trained experts' time is limited and Citizen Science projects are challenging to establish, although public interest is generally high. Thus, the generation of data is not easy.

A special case of the data scarcity problem is the data imbalance problem, i.e. the common observation that 80%-90% of the data belong to a small subset of L' classes among the total number of L observed classes, with $L' << L$. In the data imbalance problem, not all data is scarce but only a subset, i.e. certain classes. Also, scarcity of data in certain classes and abundance for the others has to be seen with regard to the task to be fulfilled. An imbalanced dataset with L' much larger than L is imbalanced by the definition provided, but if providing more data for L does not lead to a significant increase in classifica-

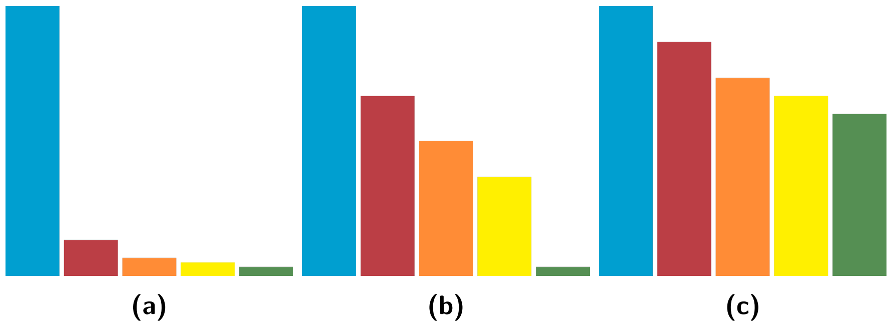


Figure 54: Depiction of different levels of class imbalance Examples of *a)* highly imbalanced, *b)* imbalanced and *c)* slightly imbalanced data. The colored bars represent different classes. The height of a bar is the abundance.

tion/detection performance, one would not describe this as a data imbalance problem. We define three types of data imbalance – highly imbalanced data, imbalanced data and slightly imbalanced data (cmp. Figure 54). The slight imbalanced data (Figure 54c), which likely does not lead to an imbalance problem, although there is data imbalance. Highly imbalanced data is characterized by L' being at least one order of magnitude larger than all other classes L, therefore $L' >> L$. Slightly imbalanced data is still imbalanced, but L' is only slightly more abundant than L, therefore $L' > L$. Imbalanced data lies in between these two categories. Please note that these categorizations are only rough categorizations and that the borders cannot be clearly defined. Nevertheless, such a categorization helps us in finding appropriate solutions for each case.

Different methods for compensating class imbalances exist. These are over- and undersampling of data [Chawla et al., 2002; H. He et al., 2008; Wilson, 1972], class weights, class aware loss functions, cost-sensitive learning [Elkan, 2001; Kukar, Kononenko, et al., 1998; Khan et al., 2018] and post-processing the output class probabilities also known as thresholding, which could be regarded as a special case of cost-sensitive learning [Lawrence et al., 1998; Richard and Lippmann, 1991]. Class weights, class aware loss functions, cost-sensitive learning are only applicable to certain algorithms. Sampling can further be divided into pre-processing- and online sampling strategies. The pre-processing strategies duplicate and modify the data. Pre-processing with over- and undersampling is applied before the classification is run and is therefore independent of the algorithm used. Therefore we will focus on pre-processing over- and undersampling in the following. Online sampling strategies are incorporated in algorithms using batches or randomly sampling the data, e.g. the balanced extension to the H^2SOL (see Section 3.4, Digression). The selection of the batches or the random sampling are modified to pick rare classes more often.

Prior work has been published to investigate the influence of class imbalance on machine learning algorithms, e.g. [Buda, Maki, and Mazurowski, 2017], but no investigation concerning the case of marine imaging is known to the author.

For a review about tackling class imbalance have a look at Haixiang et al., 2017.

Sampling strategies try to increase (oversampling) or decrease (undersampling) the data density in certain areas, where there is either not enough or too much training data present in the dataset to compensate for the data imbalance problem. Furthermore, it is common to combine oversampling and undersampling. Examples of oversampling methods are Random Oversampling [Batista, Prati, and Monard, 2004], ADASYN [H. He et al., 2008] and SMOTE [Chawla et al., 2002]. Examples for undersampling methods are Random Undersampling [Batista, Prati, and Monard, 2004], Edited Nearest Neighbours [Wilson, 1972] and NearMiss [Mani and I. Zhang, 2003].

Random oversampling (ROS) [Batista, Prati, and Monard, 2004] is a simple method designed to balance a dataset. With this method, annotation patches I_j belonging to the classes that are to be oversampled are drawn with replacement and added to the dataset until the desired class sizes are reached. This results in a larger dataset containing some of the I_j multiple times.

Synthetic Minority Over-sampling Technique (SMOTE) [Chawla et al., 2002] is an algorithm that generates synthetic samples from existing ones of a dataset. It was originally formulated for two-class classification problems. If $s'(\Omega^{(l)})$ new samples are to be created for a class $\Omega^{(l)}$, $s'(\Omega^{(l)})$ data points $\left\{x_{j=1,..,s'(\Omega^{(l)})}\right\} \subset \Gamma$ of this class are randomly selected. For each of these x_j, the K nearest-neighbors in Γ are computed forming $x_{k=1,...,K}$, while K is a hyper-parameter and must be determined. One of these K nearest-neighbors x_k is selected randomly. The new sample is determined to be:

$$x'_j = x_j + \lambda * (x_k - x_j) \tag{12}$$

with λ $[0, 1]$ being a random number.

ADASYN is similar to SMOTE, but it improves the generation of new data adaptively to the data distribution. This means that for minority samples which are near many majority samples and thus are likely to be misclassified, more synthetic samples are generated. To achieve this the K nearest-neighbors $x_{k=1,...,K}$ of a minority class sample x_j are computed. For data point of the minority class x_j we determine the ratio of majority data samples $\varrho_j = \frac{|\{x_k|\Omega(x_k)=\Omega_{\text{maj}}\}|}{K}$ in the neighborhood of x_j. By normalizing ϱ_j we yield a density distribution $\hat{\varrho}_j = \frac{\varrho_j}{\sum_{\varrho_{j'}}}$, which determines the number of synthetic data points to generate $\hat{\varrho}_j * s(\Omega^{(l)})$, where $s(\Omega^{(l)})$ is the number of samples of class $\Omega^{(l)}$. The generation of a synthetic data point x'_j in the vicinity of x_j is done analogously to SMOTE by randomly choosing an x_k with $\Omega(x_k) = \Omega_{\text{min}}$ and placing it on a direct line between the two data points x_j and x_k (see Equation (12)).

Random undersampling (RUS, [Batista, Prati, and Monard, 2004]) is a simple undersampling method that randomly removes features x_j (or annotation patches I_j) belonging to all classes that are to be subsampled until the desired sample size $s(\Omega^{(l)})$ is reached.

Image transformations used for data augmentation can also be applied as part of an oversampling method that can be employed to balance an imbalanced training dataset as proposed by Perez and Wang, 2017. To balance the dataset, the transformations are applied to patches $\mathtt{I}_j$ from minority classes that are to be oversampled. Transformations used here are a 90-degree rotation, Gaussian blur with $\sigma = 1$ and flipping using one of both image axes. Transformations are chosen randomly. In the following, this oversampling method is referred to as *Transformation Oversampling* (TROS).

All the used sampling methods introduced above require a desired sample size $s(\Omega^{(l)})$ for each class $\Omega^{(l)}$. The term *balancing rule* will be used to describe the rule that defines which sample size $s(\Omega^{(l)})$ a class should be sampled to.

Three different rules for oversampling are introduced in this section $\{\delta_{100}, \delta_{50}, \delta_{50,75,100}\}$.

The most intuitive rule is δ_{100} that sets the sample size $s(\Omega^{(l)})$ to the sample size of the majority class $s(\Omega_{\text{maj}})$.

$$\delta_{100} : s(\Omega^{(l)}) = s(\Omega_{\text{maj}}) \tag{13}$$

When resampling highly imbalanced datasets, the synthetically generated samples are only derived from a small number of samples. Thus it may be the case that at some point generating more samples does not significantly increase the accuracy of the classifier, due to the redundancy in the data. Additionally, there may be a loss of classification performance on the majority classes if all classes are sampled to the same size.

A solution for this may be oversampling rare classes to a size of $\frac{1}{2}s(\Omega_{\text{maj}})$ and keep the larger classes at their original size.

$$\delta_{50} : s(\Omega^{(l)}) = \begin{cases} \frac{s(\Omega_{\text{maj}})}{2} & \text{if } s(\Omega^{(l)}) < \frac{s(\Omega_{\text{maj}})}{2} \\ s(\Omega^{(l)}) & \text{else} \end{cases} \tag{14}$$

This rule may increase the classification accuracy of the rare classes keeping that of the common classes reasonably high, thus preventing a high loss of average precision per class caused by misclassification of common classes.

Using the third rule $\delta_{50,75,100}$ the sample sizes $s(\Omega^{(l)})$ are divided up into three ranges.

$$\delta_{50,75,100} : s(\Omega^{(l)}) = \begin{cases} \frac{1}{2}s(\Omega_{\text{maj}}) & \text{if } s(\Omega^{(l)}) \leq \frac{1}{4}s(\Omega_{\text{maj}}) \\ \frac{3}{4}s(\Omega_{\text{maj}}) & \text{if } \frac{1}{4}s(\Omega_{\text{maj}}) < s(\Omega^{(l)}) \leq \frac{1}{2}s(\Omega_{\text{maj}}) \\ s(\Omega_{\text{maj}}) & \text{else} \end{cases} \tag{15}$$

In addition, two rules $\{\hat{\delta}_{75}, \hat{\delta}_{50,100}\}$ combining oversampling with (random) undersampling are evaluated. The first rule δ_{75} completely balances the dataset, but decreases the variety of the largest classes by removing a certain share of their training samples randomly. Many of the synthetic minority class samples are derived from a small number of annotations. Because of this, the variance of these classes may be smaller than the variance of the majority classes even after oversampling. Applying this rule may reduce this difference.

$$\hat{\delta}_{75} : s(\Omega^{(l)}) = \frac{3}{4} s(\Omega_{\text{maj}}). \tag{16}$$

The other rule introduced here is adapted from a combined undersampling and oversampling approach introduced in Chawla et al., 2002. The method mentioned there includes undersampling the majority class to half the size and oversampling the minority class to $s(\Omega_{\text{maj}})$ in a two-class classification problem. This is extended to the multiclass classification problem at hand. The desired sample sizes $s(\Omega^{(l)})$ are computed as follows:

$$\hat{\delta}_{50,100} : s(\Omega^{(l)}) = \begin{cases} \frac{s(\Omega_{\text{maj}})}{2} & \text{if } s(\Omega^{(l)}) \geq \frac{s(\Omega_{\text{maj}})}{2} \\ s(\Omega_{\text{maj}}) & \text{else} \end{cases} \tag{17}$$

The classification results are evaluated using the macro-averaged recall, precision [Sokolova and Lapalme, 2009] and F_1-score [Ferri, Hernández-Orallo, and Modroiu, 2009]. Macro-averaging means that the respective measure is first computed separately for each class, then the arithmetic mean of the per-class measures is computed to obtain a performance measure that is suitable for equally weighting all classes regardless of their sample sizes. If the average of the class-wise measures were weighted by class size, as usual, low scores for small classes would lower the average much less, while for common classes the loss would be much stronger. This is important to assess whether a classifier can classify rare classes as well as common classes.

The macro-averaged recall R_{macro} is defined as

$$R_{\text{macro}} = \frac{1}{L} \sum_{l} R(\Omega^{(l)})$$

where $R(\Omega^{(l)})$ denotes the recall of class $\Omega^{(l)}$.

The macro-averaged precision P_{macro} is defined as

$$P_{\text{macro}} = \frac{1}{L} \sum_{l} P(\Omega^{(l)})$$

where $P(\Omega^{(l)})$ denotes the precision of class $\Omega^{(l)}$.

To evaluate the overall classification performance, the macro-averaged F_1-score $F_{1,\text{macro}}$ is computed. It is defined as

$$F_{1,\text{macro}} = \frac{1}{L} \sum_{l} F_1(\Omega^{(l)})$$

with $F_1(\Omega^{(l)}) = \frac{2R(\Omega^{(l)})P(\Omega^{(l)})}{R(\Omega^{(l)})+P(\Omega^{(l)})}$ where $F_1(\Omega^{(l)})$ is the class-wise F_1-score, which is the harmonic mean of $P(\Omega^{(l)})$ and $R(\Omega^{(l)})$, with both values weighted equally.

For evaluation of these sampling strategies, i.e. sampling algorithms combined with balancing rules, we performed a cross-validation experiment with three folds on a dataset from the PAP area (please note that a subset of the PAP dataset different to the one presented before was used). Instead of doing a time-consuming nested CV, each training fold is further split into a training set Γ^t (90% of the current training fold) and an optimization set Γ^o (10% of the current training fold). As classifier AlexNet was chosen because of its simplicity and rather small size, fit for a more in-depth inspection of layer weights. Instead of the usual input size of $228 \times 228 \times 3$ we reduced the input size to $64 \times 64 \times 3$ to increase the speed of the sampling algorithms. The network structure is depicted in Figure 55.

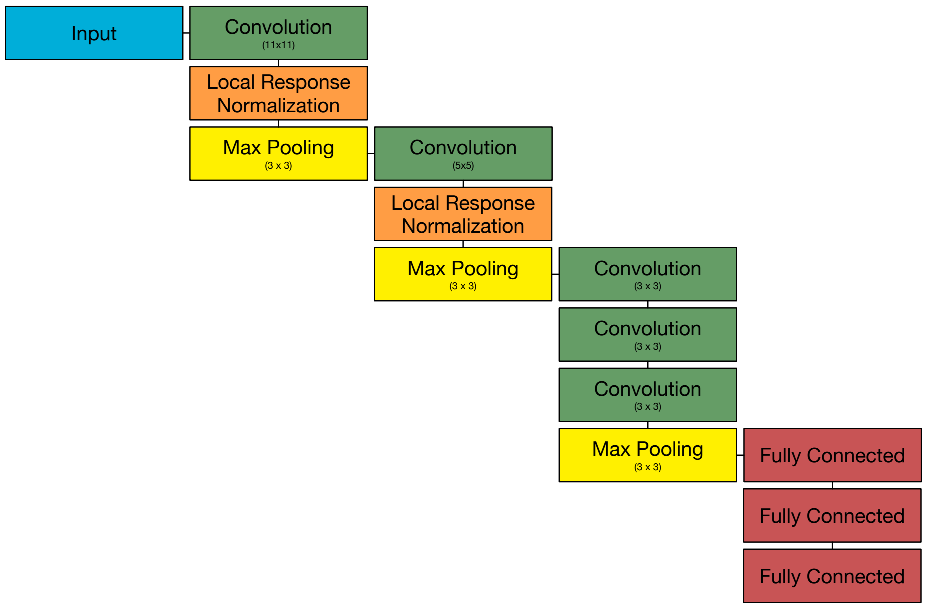


Figure 55: Architecture of AlexNet AlexNet is the first deep convolutation network. Unlike today's network it uses a large convolutional filter (11×11). In addition, it is not very deep.

In Table 12 a-c), the results of the AlexNet classification using the different balancing rules compared to the classification results without any sampling (baseline) are shown. It is evident that sampling helps in increasing the classification performance significantly. The best results are achieved using TROS with the $\delta_{50,75,100}$ rule, which results in an increase of roughly nine basis points for the $F_{1,\text{macro}}$ score. SMOTE oversampling combined with random undersampling is almost as good comparing the $F_{1,\text{macro}}$ score (8 basis point) but achieves a much

$F_{1,\text{macro}}$		SMOTE	ADASYN	ROS	TROS
baseline	0.6868				
δ_{50}		0.7571	0.7404	0.7416	0.7651
$\boldsymbol{\delta_{50,75,100}}$		0.7525	0.7432	**0.7445**	**0.7766**
δ_{100}		0.7581	**0.7434**	0.7433	0.7621
$\hat{\delta}_{75}$		**0.7653**			0.7607
$\hat{\delta}_{50,100}$		0.7652			0.7578

(a)

R_{macro}		SMOTE	ADASYN	ROS	TROS
baseline	0.6585				
δ_{50}		0.7225	**0.7082**	0.7266	0.7892
$\delta_{50,75,100}$		0.7332	0.7070	0.7249	0.7900
δ_{100}		0.7250	0.7067	**0.7280**	0.7767
$\hat{\delta}_{75}$		0.7317			0.7888
$\boldsymbol{\hat{\delta}_{50,100}}$		**0.7400**			**0.7961**

(b)

P_{macro}		SMOTE	ADASYN	ROS	TROS
baseline	0.7345				
$\boldsymbol{\delta_{50}}$		**0.8159**	0.7915	0.7688	0.7495
$\delta_{50,75,100}$		0.7907	0.7985	**0.7739**	**0.7691**
δ_{100}		0.8087	**0.8016**	0.7689	0.7563
$\hat{\delta}_{75}$		0.8153			0.7425
$\hat{\delta}_{50,100}$		0.8065			0.7334

(c)

	$F_{1,\text{macro}}$	R_{macro}	P_{macro}
baseline	0.6868	0.6586	0.7345
Only DA	0.7213	0.6989	0.7751
DA, SMOTE, δ_{50}	0.8000	0.7903	0.8206
DA, TROS, $\delta_{50,75,100}$	0.7919	0.7847	0.8030
DA, SMOTE, $\hat{\delta}_{75}$	**0.8145**	0.8110	**0.8248**
DA, SMOTE, $\hat{\delta}_{50,100}$	0.8120	**0.8136**	0.8157

(d)

Table 12: AlexNet classification results. a-c) Comparison of different balancing rules and various sampling algorithms. Best results per row for a-c) are shown in boldface. Additionally, these per row results are color-coded according to ranking. d) Influence of additional data augmentation. For d) the results per row are color-coded as usual.

higher macro precision than the method mentioned above (81.5% vs. 76.9%) at the cost of a much lower macro recall value (74% vs. 79,6%). ADASYN and ROS are underperforming, therefore the undersampling experiments were not executed for these algorithms.

classe	baseline F_1	DA, SMOTE, $\hat{\delta}_{75}$ F_1	class	baseline F_1	DA, SMOTE, $\hat{\delta}_{75}$ F_1
*Amperima**	0.9378	$\nearrow$ 0.9662	*Oneirophanta*	0.812	$\nearrow$ 0.9098
cnidaria 1*	0.9683	$\nearrow$ 0.9784	*Ophiuroidea**	0.932	$\nearrow$ 0.9630
cnidaria 2	0.731	$=$ 0.7299	ophiuroidead disk	0.3916	$\uparrow$ 0.6063
cnidaria 3	0.8043	$\uparrow$ 0.9437	*Peniagone*	0.4663	$\uparrow$ 0.6766
cnidaria 4	0.78	$\nearrow$ 0.8179	polychaete	0.7276	$\uparrow$ 0.8956
cnidaria 5	0.8968	$\nearrow$ 0.9013	*Porifera*	0.5811	$\uparrow$ 0.7051
crinoid 1	0.5423	$\uparrow$ 0.8179	*Pseudosticho*	0.3807	$\uparrow$ **0.7852**
crinoid 2	0.6063	$\uparrow$ 0.7493	stalked tunicate	0.684	$\uparrow$ 0.7913
Echiura	0.5178	$\uparrow$ 0.7095	*Tunicata*	0.4244	$\uparrow$ 0.5829
*Foraminifera**	0.864	$\nearrow$ 0.9459			
			$F_{1,\text{macro}}$	0.6868	0.8145
			F_1	0.8841	0.935

Table 13: Single Class F_1-scores including total macro and weighted F_1-score. The most abundant classes are marked with *. $\uparrow$ denotes an improvement of 10 basis points or more, $\nearrow$ an improvement of up to 10 basis points and $=$ a value which increased/decreased not more than 1 basis point.

The results of combining the sampling methods with data augmentation are shown in Table 12 d). Here the runner-up from above – SMOTE combined with the $\hat{\delta}_{75}$ balancing rule is the best, which gains an additional five basis points using data augmentation. Interestingly, data augmentation without sampling only gains 3.5 basis points compared to the baseline. The single class performances are listed in Table 13. The most significant improvements using sampling are made for the minority classes, e.g. cnidaria 3, which improves by 14 basis points or crinoid 2, which even improves by 27 basis points. This impressively shows the impact of our sampling strategy. Although the increase in F_1-score is only five basis points, the $F_{1,\text{macro}}$-score improves by 13 basis points. In addition, as can

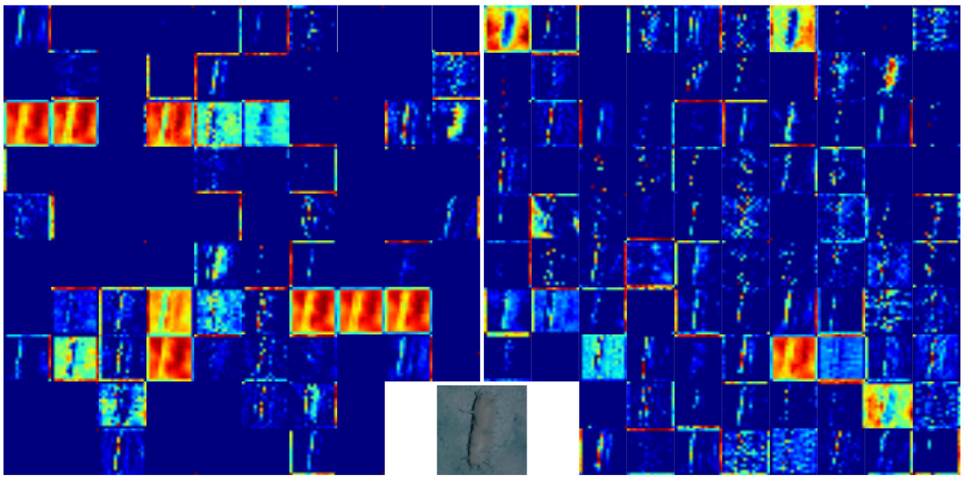


Figure 56: Example plots of the activations of the first layer of AlexNet The left image shows the activations for pure data augmentation and the right image of data augmentation combined with the best sampling approach (cmp. Table 12d). The visualization uses a heatmap color scheme, i.e. blue colors show regions with no activity and red areas are most active.

be seen in Figure 56 according to the activations, the AlexNet classifier tends to gain generalization performance, using oversampling in combination with data augmentation compared to using pure data augmentation. In this figure, we can see that more, and also more unique filters are active and that the filters generated by the convolutional neural network are detecting more edges, or small details like the tentacles of the holothurian, rather than memorizing the whole holothurian.

Digression: Sampling for non-deep learning

In addition, we investigated the influence of SMOTE and ADASYN oversampling on the SVM classifier. The SVM results are listed in Table 14. It turns out that oversampling impairs performance. This is unfortunately inherent of the way the SVM classifies and in that SMOTE and ADASYN generate data. The SVM tries to find a separating hyperplane between data points of different classes. SMOTE and ADASYN introduce new data points between the data points of differently labeled data (cmp. Equation (12)). Therefore, the data is placed close to the separating hyperplane, increasing the number of support vectors (cmp. Figure 57) needed to establish a hyperplane that separates the differently labeled data, while still not gaining any better scores. This results in overfitting of the classifier.

	SMOTE			ADASYN		
	R_{macro}	P_{macro}	$F_{1,\text{macro}}$	R_{macro}	P_{macro}	$F_{1,\text{macro}}$
baseline	**0.5571**	**0.6223**	**0.5796**	**0.5571**	**0.6223**	**0.5796**
δ_{50}	0.5541	0.6095	0.5729	0.5525	0.6074	0.5711
$\delta_{50,75,100}$	0.5528	0.6123	0.5735	0.5494	0.6045	0.5684
δ_{100}	0.5503	0.6036	0.5680	0.5475	0.5986	0.5643

Table 14: SVM results: Best results are shown in boldface.

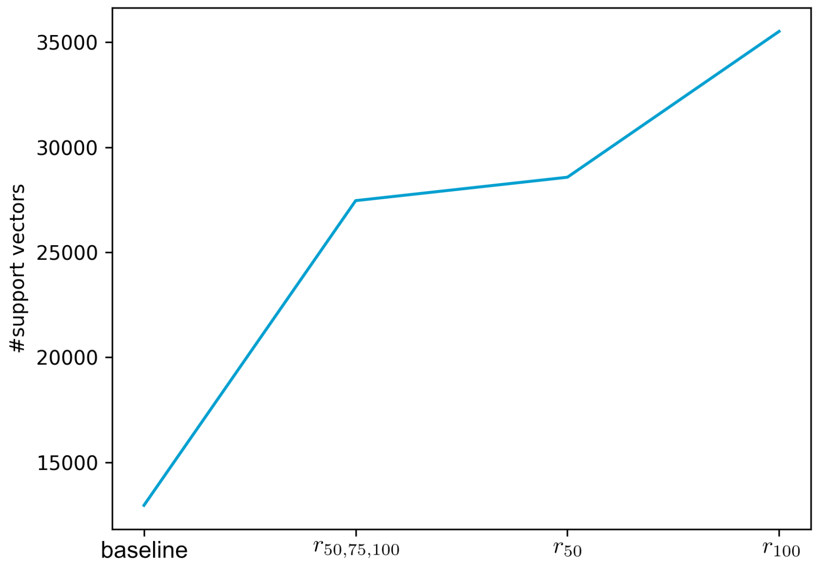


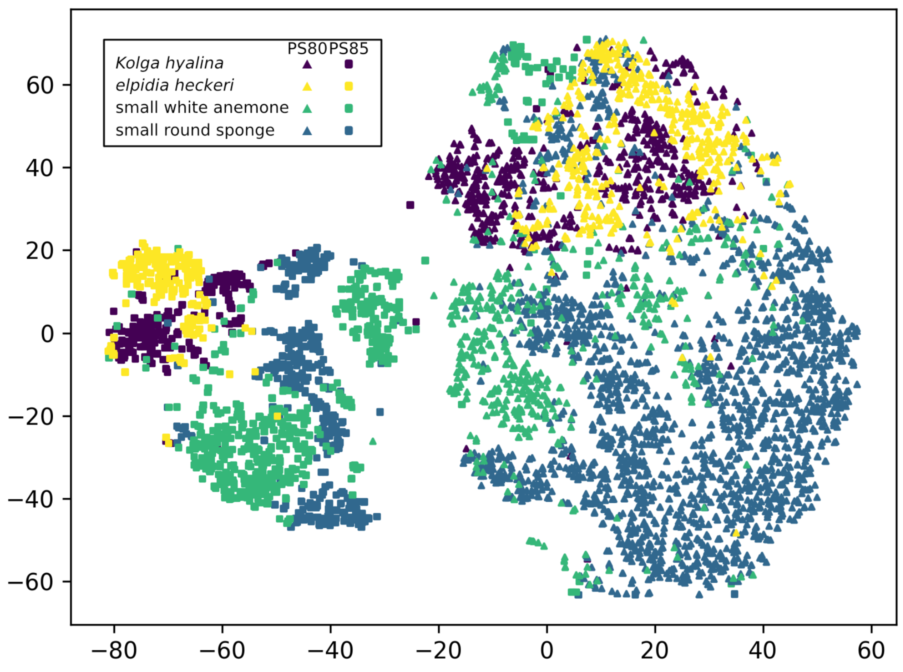
Figure 57: SVM and the balancing rules Number of support vectors compared to the balancing rule applied.

4.5 Application of learners to new data and the concept drift problem

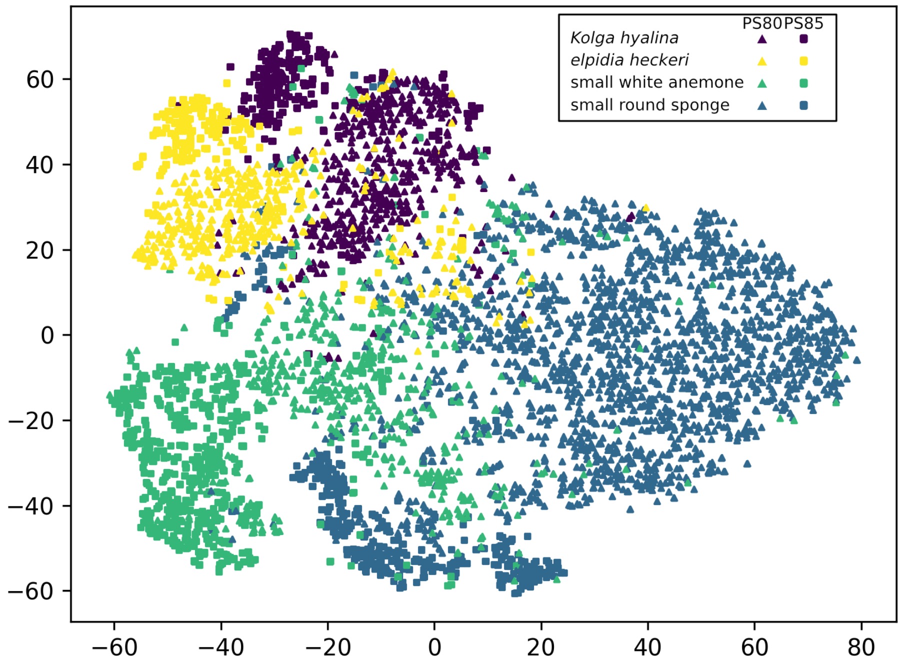
So far, we have trained networks on one part of a dataset and tested them with another part of the same dataset. However, often we have data from a new cruise, potentially recorded with different gear, from a different year or even a different region. The challenge here is that the distribution of data could change. Therefore our learned model might not be valid anymore. The idea that a concept that was learned is not valid anymore and the new data follows a new distribution is called concept drift. To read more about concept drift have a look at [Žliobaitė, Pechenizkiy, and Gama, 2016; Barros and Santos, 2018].

We have analyzed concept drift on two datasets from the Hausgarten region. One of them is PS80/179-3 (PS80) from 2012, which was already introduced earlier (see Section 2.1) and PS85/474-1 (PS85) from 2014. We used a DenseNet (cmp. Section 4.1) to extract features from the output of the last convolutional layer, for four of the main classes of the Hausgarten area. These features were transformed to 2D using t-SNE (cmp. Section 2.1) for visualization purposes. In Figure 58a the result is displayed. The two datasets (triangles PS80 and squares PS85) are separated, independent of the class, which indicates that a concept drift is present. If there were no concept drift, we would expect the classes to be clustered instead of the datasets, i.e. each class would be separated from the rest instead of one dataset being separated from the other one. Due to the concept drift, it is to be expected that a classifier may find it difficult to classify one dataset if trained on the other one. We will refer to this concept drift as being severe. A severe concept drift exists if the data can be divided into two clusters, where each cluster is formed by one dataset.

The reason for the concept drift is in the data being fundamentally different, i.e. having a different distribution. Feature extraction can compensate or aggravate the situation. For example, if features are scale-invariant, they can compensate for different distances to the seafloor in the data to counteract the concept drift. Therefore, we used another CNN, ResNet50 (cmp. Section 4.1),



(a) DenseNet



(b) ResNet50

Figure 58: Concept drift for different feature extractions Dimensionality reduction using t-SNE on features extracted from the last convolutional layer of *a)* DenseNet, *b)* ResNet50 of a subset of Hausgarten datasets PS80/179-3 (PS80) and PS85/474-1 (PS85). Same colors represent the same class, while same symbols represent the same dataset.

and repeated the same experiment as above, to see if a severe concept drift also occurs. The results are presented in Figure 58b. Although the datasets do not overlap completely, classes form clusters rather than the datasets. This would be defined as mild concept drift, i.e. datasets do not overlap, but classes form clusters. Even though the concept drift is only mild, classification of one dataset, while training on the other one may still be challenging without adjusting for the concept drift.

There are multiple methods to deal with concept drift, e.g. a) using an ensemble of networks, b) Fine-tuning the model, c) retraining the last layer or retraining the last fully connected layers, which we will present in the following.

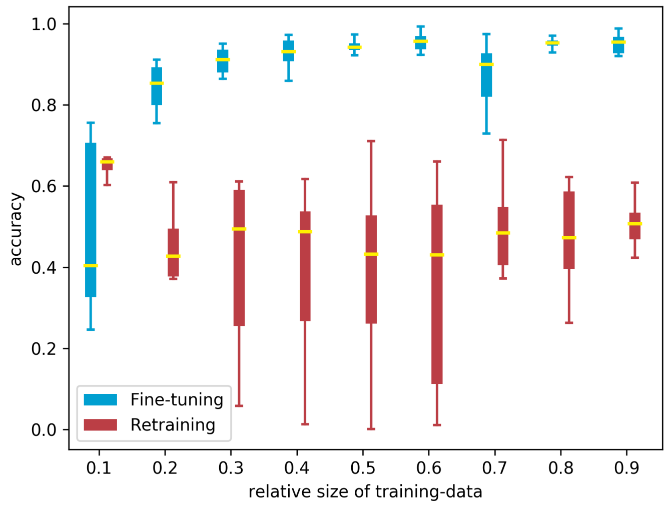
a) Using an ensemble is most useful when there are two or more concepts and we need to classify data, where we do not know from which concept it originates. This is because an ensemble incorporates many classifiers, where each classifier usually learns the data from one dataset. In our case, it is more common that we have existing datasets on which we can train, and a new dataset we want to classify. Ensembles may be not that well fit for this case because we know that the data is new and no data from the previously learned distribution has to be predicted. Therefore transforming the already learned model seems to be more promising.

b) Retraining the last layer or the last fully connected layers is used to keep the feature extraction part as it is, i.e. the convolutional layers, as these should be quite similar between the datasets. The last layer or all fully connected layers, if there are multiple, are retrained with the regular learning rate to match the new problem.

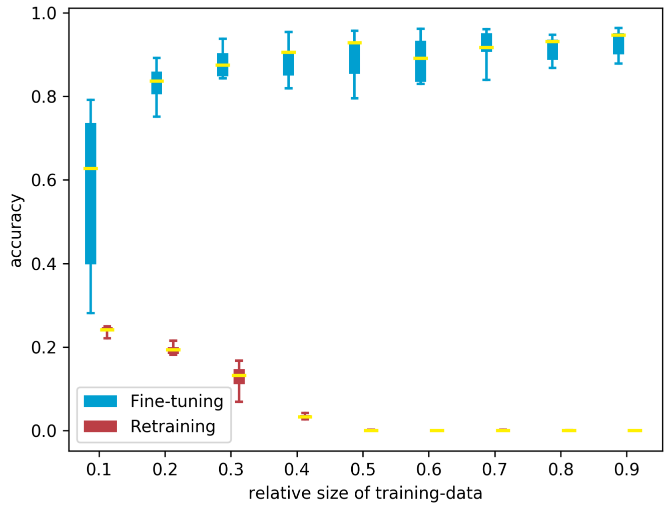
c) Fine-tuning the model means that the entire trained model is retrained but with a significantly lower learning rate. This way, the weights of the trained model are used as initialization and fine-tuned with the new data. As a rule of thumb, $\frac{1}{10}$ of the previous learning rate is taken as the new learning rate.

Both methods (b) and c)) have their theoretical advantages and disadvantages, i.e. the retraining of the last layer or last layers (b)), in theory, requires less data since fewer parameters are trained. In addition, this method should be able to handle severe concept drifts as the prediction part of the network is retrained to adapt to the new data. This however, is under the assumption that the feature extraction part is already fit for both concepts. Fine-tuning (c)) the model allows small incremental improvements to be made to the complete model, leading to a better overall mapping of the data to the model. However, the lower learning rate only allows for smaller adaptions compared to b).

At first, we performed a baseline experiment, where we computed the accuracy achieved when no compensation was performed, i.e. a network was trained on PS80 and PS85 was classified. In the following, we have conducted an experiment comparing both methods – b) retraining and c) fine-tuning. We compared both methods for the severe concept drift problem presented in Figure 58a as well as the mild concept drift problem presented in Figure 58b to compare the effects on both settings. The annotations $\mathcal{A}_j$ of PS80 of the four



(a)



(b)

Figure 59: Classification accuracy comparison for different methods tackling *a)* severe or *b)* mild concept drift To compensate the concept drift we add a percentage of all data (3604 annotations) from the second, new dataset to the training data. We compare retraining of the last layer to fine-tuning the whole network for *a)* tackling severe concept drift using Densenet as shown in Figure 58a or *b)* tackling mild concept drift using ResNet as shown in Figure 58b. The experiment was repeated 10 times. Using boxplots we show the variation of these repeated experiments.

selected classes were used as training set Λ^t to train a DenseNet or a ResNet50 respectively. Each model was either b) retrained or c) fine-tuned. In addition, we performed nine experiments in which we simulated different degrees of data availability of annotations in the new dataset PS85, i.e. 10% to 90% annotated data were used for b) retraining or c) fine-tuning. The remaining data, which was not used for b) retraining or c) fine-tuning, was used as a validation set Λ^v to assess the performance on the new data. All experiments were repeated 10 times randomly sampling the data to assess the stability of the methods.

If no training data from PS85 was used, i.e. no compensation for concept drift was performed, the classification accuracy for DenseNet is 0.58. For ResNet50 it is 0.18. This is interesting as the dimensionality reduction would suggest that classification with ResNet50 without compensation would be much more difficult than with DenseNet.

The results of the severe concept drift using DenseNet features are displayed in Figure 59a. We see that if we use 10% of the training data $\Gamma^t_{(PS85)}$ of PS85, the results for the c) retraining experiment are more stable, although higher values could be achieved by b) fine-tuning. For more than 10% of the available training data $\Gamma^t_{(PS85)}$ b) fine-tuning is favored over c) retraining, as better and more stable results are obtained.

The results of the mild concept drift using ResNet50 features are presented in Figure 59b. The results for b) fine-tuning are very similar to the results of the previous experiment. c) retraining is significantly underperforming and produces inferior results, which supports the hypothesis that b) fine-tuning should be favored. Comparing the experiments for mild and severe concept drift, we can conclude that both can be successfully tackled with b) fine-tuning the model leading to similar results.

4.6 Object detection revisited

Citizen Science revisited

Most of the following results are included in the paper Langenkämper, Simon-Lledó, et al., 2019.

Fully automatic detection of objects in marine image collections may seem unfeasible in many scenarios. Here, we demonstrate a promising alternative to a computational approach: Citizen Science, which has quickly gained popularity in recent years [Kullenberg and Kasperowski, 2016]. In the context of image analysis, citizen scientists (CS) are non-domain experts performing a detection or classification task usually with a web-application. Some recent works analyze the use of CS to increase the power of deep learning classification [Sullivan et al., 2018]. In the context of this work, the CS would be asked to perform the task of general object detection, i.e. finding all interesting objects without focusing on a single object class as most CS annotation studies do.

Since fully automatic detection is not an option and the time of domain experts is limited, this motivates a new strategy for the detection and classification of objects in marine images. First, citizen scientists are screening the large image collections for regions of interest (ROI) and mark those with a circle of adjustable size. In parallel, a subset of the data is evaluated by an experienced domain expert, i.e. a marine biologist, who assigns labels to objects, thereby providing a training set for a deep learning classifier. After the citizen scientists have marked all ROI in the complete image collection, the trained classifier is applied to all ROI for taxonomic/morphological classification (cmp. Figure 60 a)). This way, the expensive domain expert time could be used more efficiently.

In this study, we investigate the potential of such error-prone Citizen Science object detections in combination with powerful deep learning classifiers. In the first step I), we conducted a primer-experiment as a small test experiment for Citizen Science-based marine image analysis. The images were chosen to reflect typical examples and use cases. This primer-experiment gave us valuable insights into typical errors made by citizen scientists and protocol specifications for the main experiment. In step II), we performed a larger Citizen Science study including ten CS each of which manually evaluated the same set of ten images. The resulting data from I), II) and the gold standard annotations provided by an expert are used in order to simulate the outcome of a large-

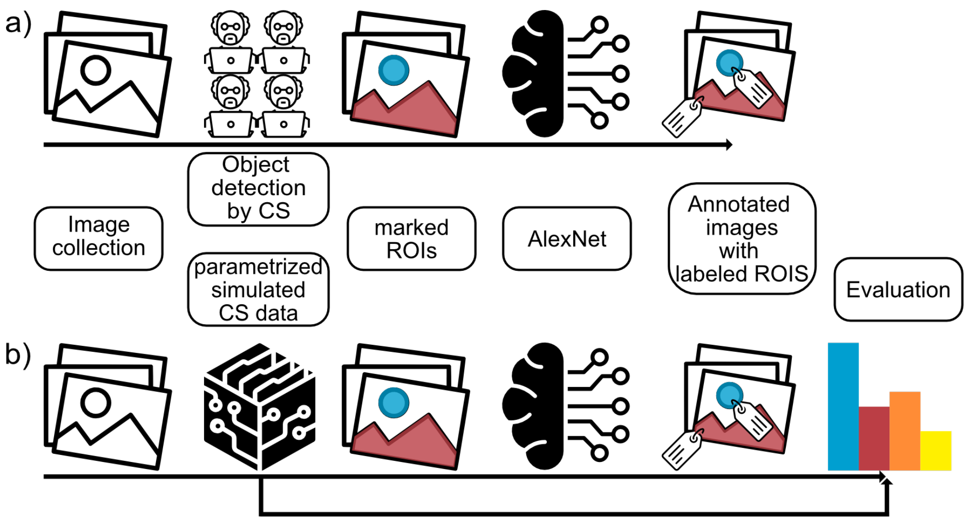


Figure 60: Workflow for our Citizen Science experiments *a)* Workflow of a Citizen Science-based annotation study. *b)* Workflow of our parametrized simulated Citizen Science study.

scale CS study in step III) resembling the errors usually done by CS. In step IV) we used the simulated annotations as input for a deep learning architecture trained on the expert gold standard annotations to answer the question if CS object detection is a valid/feasible replacement for expert object detections as a method to drive the attention of the trained deep learning classifier. By varying our simulation parameters, we were able to investigate the effects of different types of inaccuracies in the CS annotations, such as inaccurate position (IP) or inaccurate radius (IR) (cmp. Figure 60 b)).

Domain experts annotated a volume of images $\mathcal{V} = \{\mathcal{I}_{i=1,\ldots,10052}\}$. For this section, we are extending the definition of the annotation to a square annotation instead of a point annotation by including a radius r (i.e. half of the side length). A square annotation is defined as a tuple $\mathcal{A}'_j = (n, m, r, \mathcal{I}_i, \Omega)$. All other notations related to annotations in this section are not changed for the sake of brevity and clarity but implicitly are meant to be used with square annotations, i.e. $\Lambda = \{\mathcal{A}'_{j=0,\ldots,J}\}$. All annotations are split into a training set Λ^t and validation set Λ^v with a split of 90% to 10%.

I) Citizen Science Primer-experiment ($\mathcal{CSP}$) In the Citizen Science primer-experiment, eight CS detected and marked objects in two hand-selected images. In contrast to the expert, the CS only needed to detect and mark objects without providing a label, thus a CS annotation is defined as a tuple $\mathcal{A}^C_{\hat{j}} = (n^C, m^C, r^C, \mathcal{I}_i)$. The set of all CS annotations is denoted with $\Lambda^{\mathcal{CSP}}$. One image featuring visually easier to spot objects and one showing harder to spot small objects were presented. The annotations were gathered using the Biigle 2.0 [Langenkämper, Zurowietz, et al., 2017] annotation system. The CS were given a video instruction on how to use the annotation tool and were provided with the example images shown in Figure 16, as well as the instruction to look for interesting objects (excluding stones/nodules and sand). Furthermore, the CS were advised to zoom in to identify smaller objects. Further instructions were not provided intentionally. One could, for instance, provide a specific protocol to use a specific zoom level or to label a specific part of each object. We intentionally omitted this because this would have involved the classification of objects and the application of a specific protocol, which would have lead to errors. In addition, we wanted to investigate whether relatively good results could be achieved without the use of a protocol. If this were possible, it would make the realization of a CS study easier and reduce the necessary effort, i.e. more people could be recruited without the need for preparation. We evaluated the CS annotations $\Lambda^{\mathcal{CSP}}$ against the expert annotations Λ, depicted in Figure 61.

II) Citizen Science Study ($\boldsymbol{\mathcal{CSS}}$) In the Citizen Science Study hereinafter referred to as $\mathcal{CSS}$, we asked ten CS to annotate every image, i.e. provide a CS annotation $\mathcal{A}_{\hat{j}}^{\mathcal{CSS}}$ without a label, to image regions, of a hand-selected set of ten images. Each image was inspected by all ten CS. All annotations of the $\mathcal{CSS}$ are denoted with Λ^{CSS}. Motivated by the observations in the primer-study we eliminated protocol differences by asking the expert and the CS to label like the intuitive CS protocol from the $\mathcal{CSP}$ experiment, i.e. encircling the whole object. The instructions to the CS and the execution of the study were identical to the $\mathcal{CSP}$-experiment.

III) Simulated Citizen Scientists ($\boldsymbol{\mathcal{S}}$**im**$\boldsymbol{\mathcal{CS}}$) Based on the observations in the primer-experiment $\mathcal{CSP}$ and the CS study $\mathcal{CSS}$ we simulated CS object detection for all images of volume $\mathcal{V}$. This enabled us to assess the final classification results for these annotations, as we could use the domain experts' original annotations as a gold standard. To render one set of simulated CS detections, the domain experts' annotations of the validation set Λ^v were taken and transformed $(n, m, r, \mathcal{I}_i, \Omega) \mapsto (\hat{n}, \hat{m}, \hat{r}, \mathcal{I}_i, \Omega)$ to simulate inaccurate positioning (IP) or radius selection (IR) different from the experts' annotations. To simulate annotations suffering from inaccurate positioning (IP), we computed simulated artificial annotations $\mathfrak{A}_p^o = \{(\hat{n}, \hat{m}, r, \mathcal{I}_i, \Omega)\}$ resembling the behavior of citizen scientists, with o being the minimum overlap with the respective original annotation $\mathcal{A}_j$ in percent. To create $\mathfrak{A}_p^o$ in the vicinity of the expert annotations of the validation dataset Λ^v, we sample new positions $(\hat{n}, \hat{m})$ according to the following equations,

$$\hat{n} = n + Z_n * r, \tag{18}$$

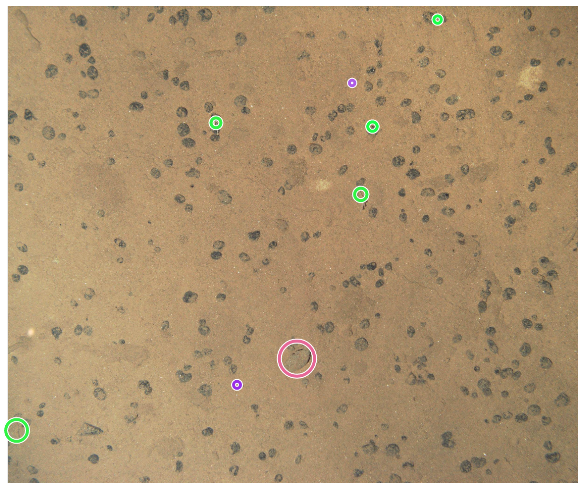
$$\hat{m} = m + Z_m * r \tag{19}$$

with

$$Z_n \sim \mathcal{U}(-(1-\sqrt{o}) * 2, (1-\sqrt{o}) * 2)$$

and

$$Z_m \sim \mathcal{U}(-(1-\sqrt{o}) * 2, (1-\sqrt{o}) * 2)$$


(a)

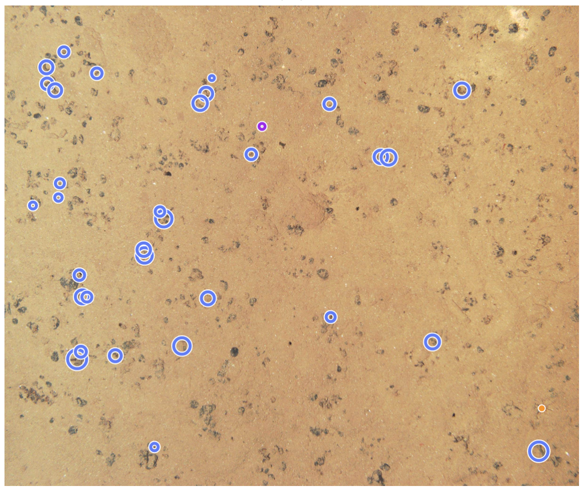

(b)

Figure 61: Images used for the CS primer-experiment Images used for the CS primer-experiment including the expert annotations, whereas the color indicates the taxa/morphotype assigned by the expert. Image *a)* features larger easier to spot objects, while image *b)* mostly features hard to spot *Porifera*.

with $\mathcal{U}(a,b)$ being the uniform distribution on the interval $[a,b]$. We explicitly only use the validation set as a basis to generate new positions to prevent mixture of training and validation set, also known as data leakage.

For evaluation, three different validation sets with different minimum overlaps $\mathfrak{A}_p^{87.5}, \mathfrak{A}_p^{75}, \mathfrak{A}_p^{50}$ were generated. For each expert annotation in the validation set Λ^v four different citizen scientist annotations were sampled, i.e. $|\Lambda^v| * 4 = |\mathfrak{A}_p^o|$.

To simulate inaccurate setting of the annotation radius (IR), we generate further sets $\mathfrak{A}_r^s = \{(x, y, \hat{r}, I_n, \Omega)\}$. These overestimate the radius by a maximum of 10% and 25% respectively ($\mathfrak{A}_r^{0.1}, \mathfrak{A}_r^{0.25}$) or underestimate it by a maximum of 10% and 25% ($\mathfrak{A}_r^{-0.1}, \mathfrak{A}_r^{-0.25}$), but keep the positions of the experts' annotations.

$$\hat{r} \quad = \quad (1 + Z_r) * r, \quad Z_r \sim \begin{cases} U(0,s) & s > 0 \\ U(s,0) & s < 0 \end{cases} \tag{20}$$

In addition, we simulate false positive (FP) annotation produced by CS. Therefore for each image (of the validation set Λ^v), we generate two random background annotations of the size 64×64, which are non-overlapping with expert annotations, i.e.

$$\mathcal{A}^{bg} = (x^b, y^b, 64, I_n, \Omega = Background)$$

with $\Delta((x^b, y^b), (x, y)) > r + 64)$, $\Delta(\cdot,\cdot)$ the Euclidean metric, and (x^b, y^b) a randomly chosen center fulfilling the above condition. These form $\mathfrak{A}^{bg}$.

Furthermore, we generated a dataset $\mathfrak{A}^{\mathcal{CSP}}$ resembling the results of the CS primer-experiment $\mathcal{CSP}$. Therefore we removed outliers and simulated deviations in position and radius using Gaussian distributions with parameters according to the CS primer-experiment data (see Figure 63 b)).

Deep learning classifier A classifier $f(x) = \Omega^{(l)}$ is trained on a training set Λ^t. In this case, we used the well-established AlexNet [Krizhevsky, Sutskever, and G. E. Hinton, 2012] (see Figure 55) because of fast and robust classification. Training was performed using the Caffe deep learning framework [Jia et al., 2014] and took 19 minutes on an NVIDIA Titan X. The network was trained for 30 epochs. Classifying the simulated data with the trained model took less than one minute for each dataset. The study compares the results of $f(\Lambda^v)$ and $f(\mathfrak{A}_p^{87.5})$, $f(\mathfrak{A}_p^{75})$, $f(\mathfrak{A}_p^{50})$, $f(\mathfrak{A}_r^{0.1})$, $f(\mathfrak{A}_r^{-0.1})$, $f(\mathfrak{A}_r^{0.25})$, $f(\mathfrak{A}_r^{-0.25}), f(\mathfrak{A}^{\mathcal{CSP}})$ as well as the ability to recognize background patches $f(\mathfrak{A}^{bg})$ in the following section.

Analyzing the CS primer-experiment from step I) we observed four kinds of errors or inaccuracies. First, the CS have produced false positives (FP), i.e. objects of no interest, or second, missed objects, so-called false negatives (FN). Third, in the case of true positive detections, the CS sometimes marked inaccurate positions (IP) of the annotations, or fourth, an inappropriate circle size (IR) was chosen.

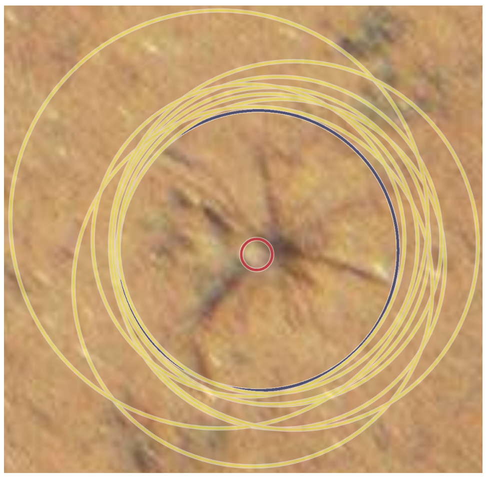
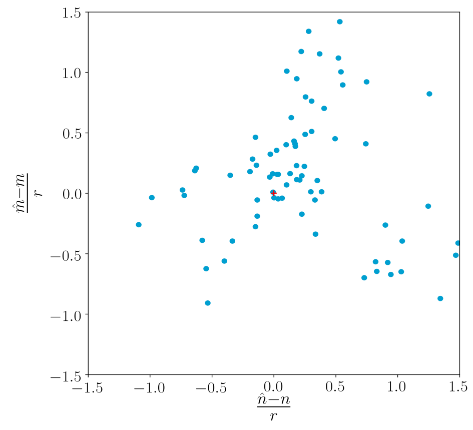


Figure 62: Difference of expert annotation, CS annotation and protocol-corrected expert annotation Results of the annotation of a *Porifera* in the CS primer-experiment $\mathcal{CSP}$. The expert annotation is in red while the CS annotations are in yellow and the protocol-corrected expert annotation is in green. The radii selected by the CS are more than ten times the expert annotation radius.

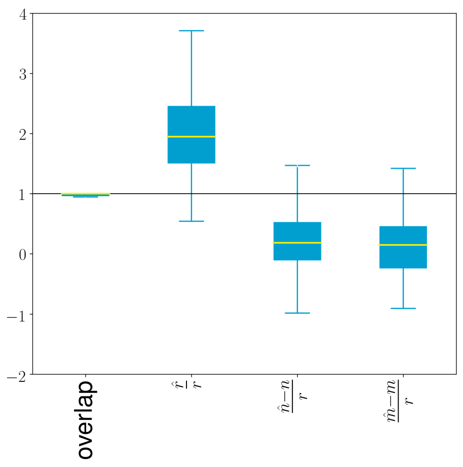
The CS primer-experiment $\mathcal{CSP}$ produced 228 annotations in total. Of these, 82 (36%) were valid, i.e. had an overlap with one of the expert annotations $\mathcal{A}_j$. This results in 146 (64%) false positives (FP). Of the 42 expert annotations, 21 (50%) were found and 21 (50%) were not found (FN). All of the 21 objects which were not found were of the same class (*Protista*). In the image featuring visually easier to spot objects, 8 out of 8 expert annotations were found. This shows how the quality of object detection is dependent on the difficulty of the task. There is a high deviation in the position of the CS annotations from the expert annotations in the CS primer-experiment $\mathcal{CSP}$ (Figure 63 a) and b). It can be seen that in Figure 63 a) most annotations are quite close to the expert annotations, but some large deviations can also be observed (cmp. inaccurate position (IP)).

Every person annotating image collections either implicitly or explicitly uses a protocol, i.e. a model on how to mark an object with a circle marker. The CS primer-experiment showed that the CS had implicitly used a protocol other than that of the expert. Experts marked objects not only to detect them but also to quantify their mass, thus sometimes only marking part of an object. In contrast, citizen scientists were focused on encircling the whole object, thus resulting in a change of center as well as a significantly higher radius (see Figure 62). In a second evaluation, we transformed the set of expert annotations Λ to the protocol-corrected set of expert annotations Λ'. To this end, the CS annotation protocol was applied to each expert annotation, i.e. the expert annotations were manually modified to enclose the entire object, also shifting the center



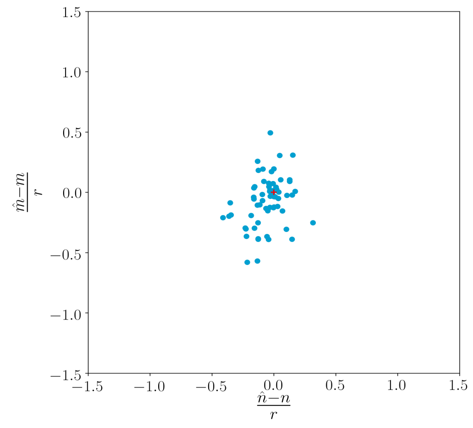
$\frac{\hat{m}-m}{r}$
$\frac{\hat{n}-n}{r}$

(a)



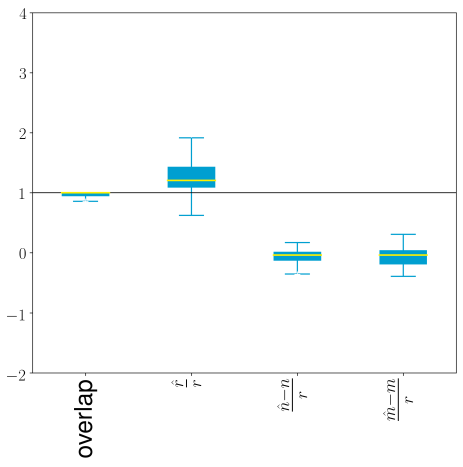
overlap
$\frac{\hat{r}}{r}$
$\frac{\hat{n}-n}{r}$
$\frac{\hat{m}-m}{r}$

(b)



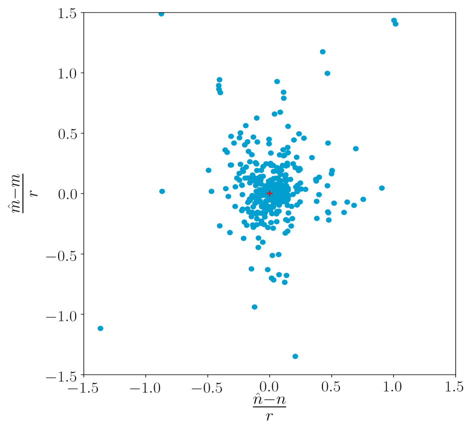
$\frac{\hat{m}-m}{r}$
$\frac{\hat{n}-n}{r}$

(c)

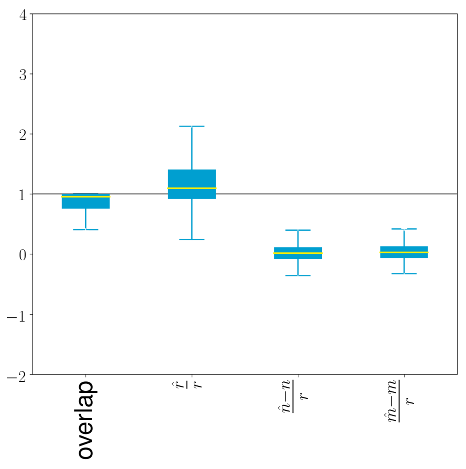


overlap
$\frac{\hat{r}}{r}$
$\frac{\hat{n}-n}{r}$
$\frac{\hat{m}-m}{r}$

(d)



(e)



(f)

Figure 63: Margins for flaws of CS annotations *a)* shows the deviation in x- and y-direction. Depicted are the deviations in the respective direction divided by the radius, i.e. $\frac{\hat{n}-n}{r}$. Each blue dot represents a CS annotation $\mathcal{A}^C_{\hat{j}}$ of the Citizen Science Primer-experiment $\mathcal{CSP}$. Please note that five outliers are not depicted in favor of a more detailed visualization. The red cross is just for orientation and depicts no deviation at all. *b)* shows the distribution of several measurement values of the CS data. Overlap displays the overlap of the expert annotation with the CS annotation. Relative radius $\frac{\hat{r}}{r}$ is the ratio of the radius $\hat{r}$ of the CS annotation and the radius r of the expert annotation. *c)* and *d)* show the same information on the same scale as a) and b) respectively but are protocol-corrected (see text for more details on the correction). *e)* and *f)* show the results for the Citizen Science Study $\mathcal{CSS}$. Each dot represents a CS annotation $\mathcal{A}^{\mathcal{CSS}}_{\hat{j}}$ of the $\mathcal{CSS}$.

of the annotation if necessary (cmp. Figure 62). The results are shown in Figure 63 c)-d). The protocol differences can be confirmed comparing the CS annotations with the protocol-corrected expert annotations (cmp. Figure 63 a)/b) and c)/d)). The protocol differences are also observable when looking at the radii in Figure 63 b)/d), which are mostly overestimated (cmp. inaccurate radius (IR)). Furthermore, we noticed that the CS rather increase the radius to encircle the whole object than running a risk of not capturing everything. This behavior could also be owing to the lack of experience in operating annotation software on a regular basis, i.e. finding the correct center of an object as well as selecting an appropriate zoom level for the task (also cmp. Figure 62). Owing to the rather generous radii provided by the CS annotators, most annotations have complete overlap with the expert annotations.

The CS study $\mathcal{CSS}$ produced 804 annotations $\mathcal{A}^C$ in total. Of these 366 (46%) were valid, i.e. had an overlap with an expert annotations $\mathcal{A}_j$. This results in 438 (54%) false positives (FP). Of the 108 expert annotations, 70 (65%) were found and 38 (35%) were not found (FN). In Figure 63 e) and f) we show the resulting deviations of the expert annotations compared to the CS annotations as observed in the CS study $\mathcal{CSS}$.

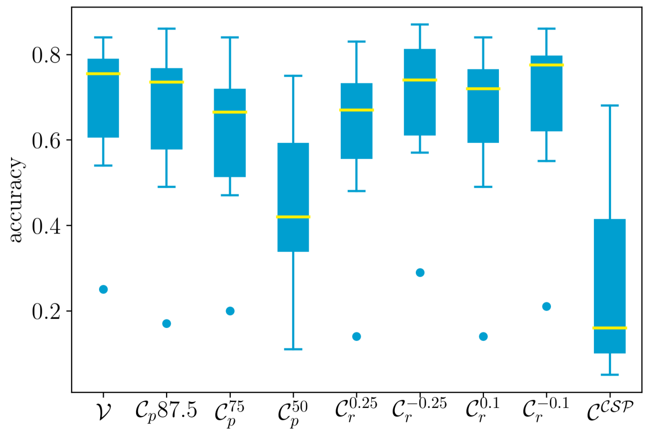
In Table 15 we show the classification accuracies for the different datasets starting with the validation data Λ^v of the expert annotations (first column, Table 15).

		inaccurate position (IP)			inaccurate radius (IR)				
	Λ^v	$\mathcal{A}_p^{87.5}$	$\mathcal{A}_p^{75}$	$\mathcal{A}_p^{50}$	$\mathcal{A}_r^{0.25}$	$\mathcal{A}_r^{-0.25}$	$\mathcal{A}_r^{0.1}$	$\mathcal{A}_r^{-0.1}$	$\mathcal{A}^{\mathcal{CSP}}$
Protista	0.84	$\nearrow$ 0.86	$=$ 0.84	$\downarrow$ 0.72	$\searrow$ 0.83	$\nearrow$ 0.87	$=$ 0.84	$\nearrow$ 0.86	$\downarrow$ 0.55
Porifera	0.82	$=$ 0.82	$\searrow$ 0.79	$\downarrow$ 0.75	$\searrow$ 0.81	$\searrow$ 0.79	$\nearrow$ 0.84	$\searrow$ 0.80	$\downarrow$ 0.68
Osteichthyes	0.71	$\downarrow$ 0.60	$\downarrow$ 0.50	$\downarrow$ 0.46	$\downarrow$ 0.57	$\uparrow$ 0.86	$\nearrow$ 0.72	$\uparrow$ 0.77	$\downarrow$ 0.05
Mollusca	0.76	$=$ 0.76	$\downarrow$ 0.69	$\downarrow$ 0.33	$\downarrow$ 0.66	$\nearrow$ 0.78	$\downarrow$ 0.72	$\nearrow$ 0.78	$\downarrow$ 0.10
Echinodermata	0.79	$\downarrow$ 0.73	$\downarrow$ 0.69	$\downarrow$ 0.38	$\downarrow$ 0.74	$\downarrow$ 0.70	$\nearrow$ 0.77	$=$ 0.79	$\downarrow$ 0.16
Crustacea	0.54	$\downarrow$ 0.49	$\downarrow$ 0.47	$\downarrow$ 0.21	$\downarrow$ 0.48	$\nearrow$ 0.57	$\downarrow$ 0.49	$\nearrow$ 0.55	$\downarrow$ 0.10
Cnidaria	0.79	$\searrow$ 0.77	$\downarrow$ 0.73	$\downarrow$ 0.61	$\downarrow$ 0.71	$\nearrow$ 0.82	$\downarrow$ 0.75	$\nearrow$ 0.81	$\downarrow$ 0.37
Bryozoa	0.75	$\searrow$ 0.74	$\downarrow$ 0.64	$\downarrow$ 0.54	$\downarrow$ 0.68	$\downarrow$ 0.67	$\downarrow$ 0.69	$\downarrow$ 0.71	$\downarrow$ 0.43
Arthropods	0.25	$\downarrow$ 0.17	$\downarrow$ 0.20	$\downarrow$ 0.11	$\downarrow$ 0.14	$\uparrow$ 0.29	$\downarrow$ 0.14	$\downarrow$ 0.21	$\downarrow$ 0.09
Annelida	0.57	$=$ 0.57	$\searrow$ 0.55	$\downarrow$ 0.36	$\searrow$ 0.55	$\nearrow$ 0.59	$\searrow$ 0.56	$\nearrow$ 0.59	$\downarrow$ 0.16
Avg	0.62	$\searrow$ 0.59	$\downarrow$ 0.55	$\downarrow$ 0.41	$\downarrow$ 0.53	$\nearrow$ 0.63	$\searrow$ 0.59	$=$ 0.62	$\downarrow$ 0.24
Weighted Avg	0.77	$=$ 0.77	$\downarrow$ 0.73	$\downarrow$ 0.59	$\downarrow$ 0.73	$\nearrow$ 0.78	$\searrow$ 0.75	$\nearrow$ 0.78	$\downarrow$ 0.41

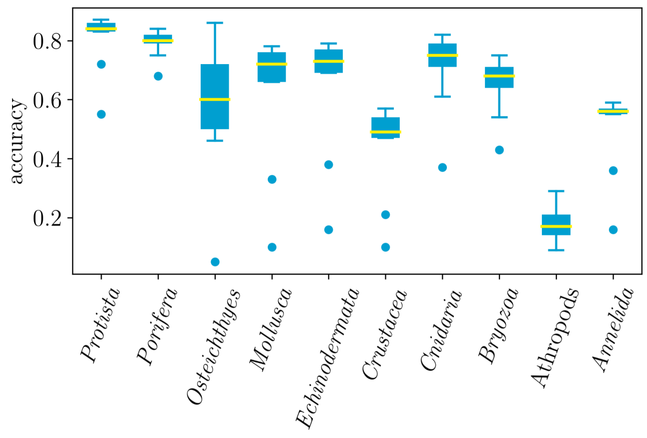
Table 15: Classification accuracy for the expert, the simulated citizen scientist annotations, and the CS primer-experiment. The arrows are just for illustration purposes. Differences of more than 3 basis points compared to the results of Λ^v are depicted with $\downarrow$ or $\uparrow$. Differences of less than 3 basis points compared to Λ^v are depicted with $\searrow$ or $\nearrow$.

Our main interest lay on the question, to what extent the position and size of the CS annotation influences the classifier's performance. Looking at the impact of the positioning inaccuracy (IP) on classification accuracy in Table 15 columns 2-4, we see a tendency that a decreasing minimum overlap yields a decreasing classifier accuracy. While *Protista* even gains 2 basis points when the minimum overlap changes to 87.5% due to positioning inaccuracy (IP), *Echinodermata* looses 6 basis points and *Osteichthyes* even looses 11 basis points. An explanation for this could be that *Protista* seem to have a unique texture everywhere, whereas *Osteichthyes* and *Echinodermata* have prototypical features at specific parts, e.g. the front and fins of *Osteichthyes* or the spines and the round shape of some *Echinodermata* (cmp. Figure 16). When shifting the position from the center of the object to the periphery, this information can be partly lost or distorted. In Figure 65 (left column) this is illustrated for a *Crustacea*. When the minimum overlap decreases the classification results get worse. The classification results for *Annelida* and *Protista* are not affected as much as the ones for the other organisms (cmp. Table Table 15 $\mathcal{A}_p^{75}$).

Looking at the experiments where the radii of the annotations have been varied (cmp. inaccurate radius (IR), Table 15, columns 5-8) we observed an increase in classification accuracy for decreasing radii for most taxa. A larger radius seems to spoil the classifier's performance. Although the overall accuracy

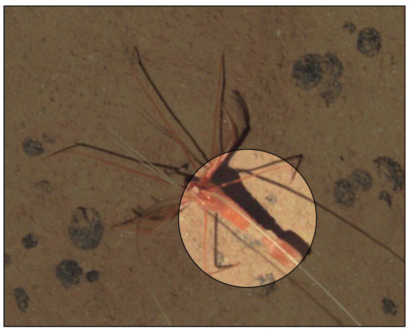


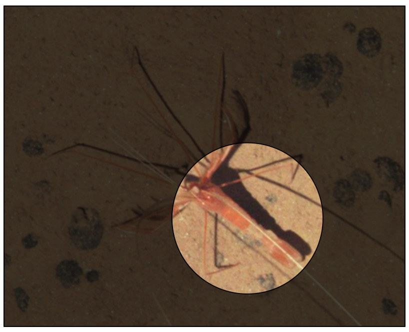
(a)

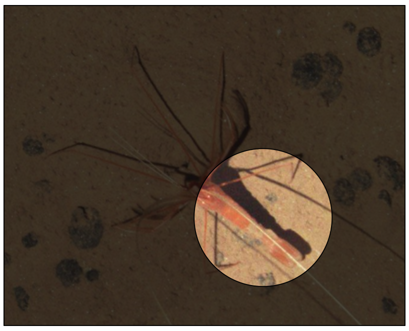


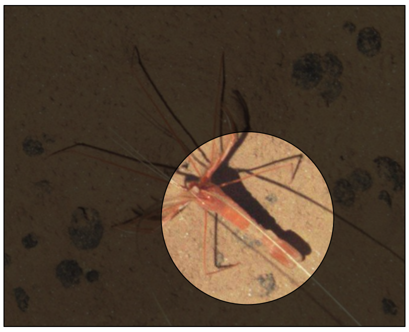
(b)

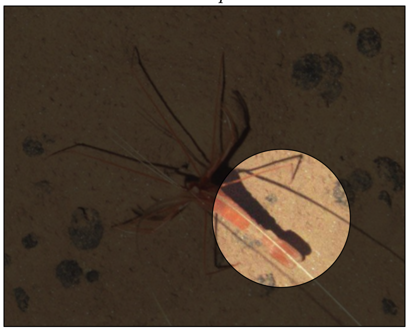
Figure 64: Overview of classification accuracies aggregated by study and by class Classification accuracy for the expert, the simulated citizen scientist annotations, and the CS primer-experiment. *a)* The accuracies are aggregated by experiment *b)* Each boxplot represents the aggregated results from each dataset $(\Lambda^v, \mathcal{A}_p^{87.5}, \ldots)$ for each class.

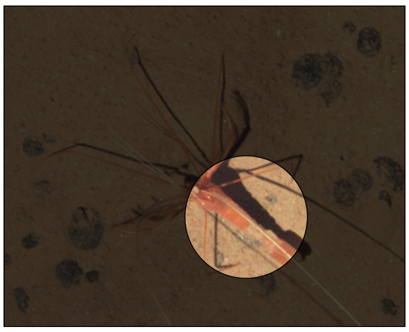

(a) Expert annotation

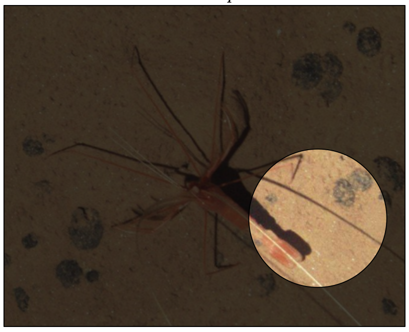

(b) $\mathfrak{A}_r^{0.1}$


(c) $\mathfrak{A}_p^{87.5}$


(d) $\mathfrak{A}_r^{0.25}$


(e) $\mathfrak{A}_p^{75}$


(f) $\mathfrak{A}_r^{-0.1}$


(g) $\mathfrak{A}_p^{50}$

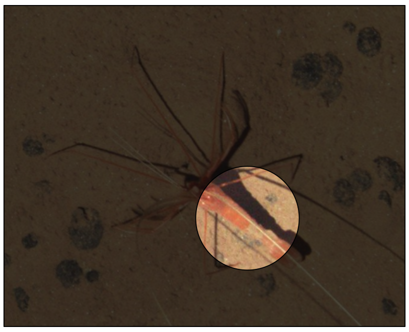

(h) $\mathfrak{A}_r^{-0.25}$

Figure 65: Inaccuracies in position (IP) and radius (IR) For illustration purposes we show the impact of different inaccuracies in position (IP) and radius (IR) on an annotation with the label *Crustacea*.

decreases/increases, the per-class accuracies behave quite differently, i.e. some classes seem to profit from the change while for others the accuracy drops. In contrast to the positioning inaccuracy (IP), the classifier appears to be more robust when the circle radius is changed (IR).

When looking at the classification results of $\mathfrak{A}^{\mathcal{CSP}}$ (cmp. Table 15, last column), we see a significant decrease in performance compared to the other experiments. Yet, this was somewhat expected because of the significant differences in the CS annotations compared to the expert annotations. Interestingly *Porifera* again only looses 14 basis points in classification accuracy in contrast to all other classes losing up to 66 basis points.

The different taxa feature individual degrees of difficulty/robustness (see Figure 64) with Arthropods being the most complex classification task. We can also see that some taxa are more robust to changes than others in general. While *Protista*, *Porifera* and *Annelida* only show small changes, especially *Osteichthyes* show considerable variations in classification accuracy when changing the simulation parameters and thus the inaccuracies.

Although the experiments $\mathcal{CSP}$ and $\mathcal{CSS}$ show a quite high number of FP, these could be identified in the classification step and therefore compensated to some degree. The correct classification of FP as $f(\mathfrak{A}^{bg}) = Background$ was performed with an accuracy of 96.78%.

In our study, we have identified and analyzed the four common annotation parameters reflecting the differences between a CS derived annotation and an expert-derived one. Those are false positives (FP), false negatives (FN), inaccurate position (IP) and inaccurate radius/size (IR).

In the primer-experiment (eight citizen scientists detected objects in two images), we observed these problems and showed that large deviations, compared to expert annotations, can occur (IP and IR). As shown with the protocol-corrected expert annotations (cmp. Figure 63 c) and d)), these deviations mostly result from protocol differences, which could be minimized by providing the citizen scientists with more detailed instructions as shown in the CS study ($\mathcal{CSS}$). In addition, the study showed that if a small crowd of untrained citizen scientists were assigned a rather complex task such as general object detection in marine images, a significant amount of data can be generated ($\mathcal{CSP}$: 228 annotations, $\mathcal{CSS}$: 804 annotations), but not all data is meaningful, i.e. a lot of false positives (FP) are annotated. Furthermore, because of the complexity of the objects to be annotated, some objects are not found at all (FN).

We simulated datasets featuring different degrees of IP, IR and FP and used these as input to the deep learning classifier. We observed that incorrect radii (IR) are less severe than a position shift of the center with the correct radius (IP). Besides, some objects are more sensitive to deviations than others, most likely owing to their shape/structure. Objects incorrectly annotated as being interesting but belonging to a background class (FP) can be recognized with 96.78% accuracy by the classifier, which seems like a good enough value to compensate for this kind of error.

The above observations motivate the following recommendations for CS data collections in the context of machine learning applications:

1. CS should be provided with clear instructions and examples about the way, the position and radius of the object should be marked.
2. The correct position seems more important than the correct radius.
3. Annotation radius should be selected not larger than necessary and should also be described by examples.
4. If CS annotations are used for classification, a smaller crop of the ROI can be tested for more accurate classifications.

The answer to the question of whether citizen scientists annotations are a feasible replacement for marine object detection in combination with deep learning classifiers has to be answered dichotomously. On the one hand, the results of the CS primer-experiment $\mathcal{CSP}$ suggest that if a significant divergence with expert annotations exists there is a considerable impact, which is shown by the predictions of our simulations. On the other hand, if the deviations are not that huge like in the $\mathcal{CSS}$, a minor impact in classification performance is noticeable.

Furthermore, measures to counter error-prone CS annotations using variations in the deep learning architecture or the training procedure, e.g. feeding simulated data to the classifier or increasing the size of the pooling layer might help in minimizing problems. For example, in this study we have shown that FP can be minimized by training the deep learning classifier to recognize background samples.

We conclude that if the CS study is well designed (see 1.-4. above, also cmp. [Schoening, Osterloff, and Nattkemper, 2016]), citizen scientist annotations are a valuable asset and errors can be within the limits of the other simulated data. Doing a joint annotation session or limiting the type of objects to be detected by a group of citizen scientists, thus splitting the task into object categories would also be useful to make the task more manageable for the citizen scientists and minimize errors. If this training is done thoroughly, it can be concluded that citizen scientist annotations are a compelling way to solve the marine object detection problem.

Digression: Deep learning and object detection

The following method has been published in Zurowietz, Langenkämper, Hosking, et al., 2018.

In Zurowietz, Langenkämper, Hosking, et al., 2018, we propose Maia (Machine learning Assisted Image Annotation), a two-step object detection approach. It is designed to minimize manual labor while still performing at a sufficient level. Most of the work for this paper has been done by my colleague Martin Zurowietz, but some preparatory work and ideas were brought in by me. The basic idea is that in the first step an autoencoder network, i.e. a network that tries to reproduce its input, learns the background of the images. An autoencoder $f^{\text{Auto}}(x) = f^{\text{dec}}(f^{\text{enc}}(x))$ consists of two parts - the encoder $f^{\text{enc}}(x)$ that learns a latent feature representation $f^{\text{enc}}(x)$ of the input and the decoder $f^{\text{dec}}(x)$ that attempts to reconstruct x from $f^{\text{enc}}(x)$. Since the deep-sea is visually rather dull in appearance and the abundance of life is rather low, a large number of random samples can be taken extracted, which usually will show a rather homogeneous background. We now use this data to train an autoencoder. Because the capacity of an autoencoder to learn is limited, rare occurrences of anything other than the background are suppressed to a certain degree. We can now apply the autoencoder to the original image. Parts of the image that contain no background, i.e. the desired objects, cannot be reconstructed well enough, which is shown by a large reconstruction error. These errors indicate that an object has been found.

In a second step, we use these object proposals of the autoencoder to train a Mask R-CNN, which is a region proposal network coupled with a convolutional neural network used for object detection and so-called instance segmentation. Instance segmentation segments the object inside the bounding box from the background.

In addition, step one and two can be performed independently. Performing only step one, a lot of object proposals can be generated quite quickly. However, all objects distinct from the background are presented. If there are already enough training samples available, e.g. by previous manual annotation of the dataset, the second step can be performed directly to find objects that were not previously annotated. The workflow and a short description of MAIA are shown in Figure 66. This is a fairly brief summary and some steps of the process have been omitted for brevity – for more details, please refer to the paper.

Results: MAIA's performance was evaluated on three different datasets (JC77 (cmp. Section 7), PAP (cmp. Section 2.2) and SO242 (cmp. Section 7)). The results are listed in Table 16.

Dataset	recall	precision	F_2
JC77	91.7	50.5	78.8
PAP	77.2	14.3	41.1
SO242	83.4	26.1	58.0

Table 16: Maia Results

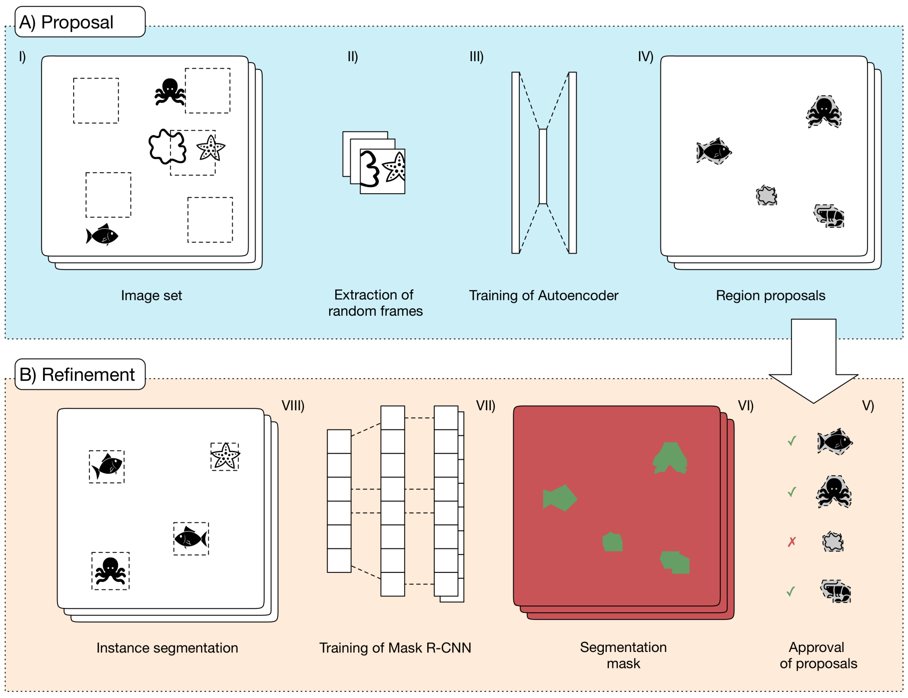


Figure 66: Maia workflow Maia is a two-step approach. *a)* In the first step, possible region proposals are determined using an autoencoder in an unsupervised way. *b)* In a second step, these proposals are used to train a supervised Mask R-CNN network. This way, objects that are not of interest can be filtered. Furthermore, the performance of the detection method is further improved by sorting out unwanted objects and feeding them to a more sophisticated algorithm for object detection. I) We start with an unannotated image set. II) Random frames of a predetermined size are extracted. III) An autoencoder is trained with these frames. IV) The autoencoder is applied and regions that do not resemble the background are proposed as interesting. Postprocessing is done to improve these proposals further. V) The proposals are manually controlled and either approved or discarded. VI) Segmentation masks are extracted, marking interesting objects as foreground (green) and the rest as background (red). VII) A Mask R-CNN network is trained with these segmentation masks. VIII) The Mask R-CNN is applied to the Image set yielding an instance segmentation, i.e. bounding boxes with segmented objects in them.

4.7 Summary and list of contributions

In this section, we have discussed the techniques that have emerged with the advent of deep learning, especially convolutional neural networks. These had a tremendous impact on the world of machine learning and computer vision, as previously unsolvable problems, such as learning hundreds to thousands of classes, could be solved. First, we introduced and described some prominent convolutional neural networks and compared their classification performance on four different marine datasets. The most successful classifiers were Xception, MobileNet, DenseNet, InceptionResNet. They achieve over 90% accuracy on all datasets except for UVP5, where the best ranking classifier still achieves an accuracy of 88%. That is an increase of roughly 50 basis points compared to the shallow learners. For Hausgarten and PAP, the distance to the shallow learners is not as significant, but it is still 12 basis points for Hausgarten and 34 basis points for PAP. We have further analyzed whether the application of transfer learning or augmentation, or both in combination, achieve better performance. In conclusion, it can be said that classifiers which applied transfer learning performed better than other variants. In most cases, transfer learning in combination with augmentation achieved a similar performance as transfer learning. Augmentation without transfer learning would sometimes even decrease performance, but sometimes lead to improvements. In addition, we have looked at the effectiveness of using the available parameters, whereby MobileNet stood out by the use of the fewest parameters and its superior or comparable performance. The same applies to the time it takes to train a network where MobileNet takes the least time. Since MobileNet has been optimized for small-scale systems and uses the least amount of parameters, this is not surprising.

In a further study, we investigated how many data samples are needed to train a CNN to achieve a certain performance level. We were able to show that even with only a very limited amount of training data, i.e. down to 100 samples, a performance of up to 85% can be achieved. Unfortunately, the results are not that stable, i.e. differences between different runs lead to significantly different results. These fluctuation stabilize with approximately 1000 samples.

The input of a CNN is usually of fixed size. The usual way to transform an annotation patch I_j to this size is to resize it, thus eliminating the size information. We analyzed whether the inclusion of size information in a CNN is beneficial to the classifier's performance by comparing different methods respecting size to the established resizing method, disregarding size information. No method proved to be superior to resizing. However, the padding with noise method proved to be competitive with the resizing method. Nevertheless, settings where size information plays a more important role, such as with diatoms, may come to different conclusions and show that padding with noise may even be superior to resizing.

Another common problem is that of data scarcity and data imbalance. We have defined the terms and concepts in the context of marine image informatics and proposed sampling methods and balancing rules to tackle these problems. We were able to show that significant improvements of 13 basis points on average and 40 basis points for a single class improvement could be achieved by over- and undersampling in combination with data augmentation. Furthermore, we could show that this approach leads to a better feature representation in the filters of an AlexNet, i.e. more filters are active and they show more general concepts, e.g. edges, instead of memorizing the input.

The data distributions of datasets, captured at the same location but at a later date or with different gear, might change. This change in data distribution is called concept drift. We analyzed that different feature extractors can lead to different degrees of concept drift. We compared fine-tuning the entire model to retraining of the last layer to compensate for concept drift. Fine-tuning proved to be significantly better than retraining for both feature extractors.

Deep learning has been made popular with considerable successes in classification, yet, it is not limited to that. We presented MAIA - a combination of an autoencoder network for unsupervised detection and Mask R-CNN for supervised instance segmentation. The autoencoder makes it possible to identify possible objects of interest efficiently and without manual input. These proposals can be further refined by manual inspection. The refined proposals are then used as training data for the Mask R-CNN, which detects and segments objects of interest in marine images with a reasonable recall. Although precision is not very high for all tested datasets, at least almost all objects are found. Additional unwanted objects can be discarded manually. Even though this process involves manual labor, it still saves a lot of time compared to the established method of fully manual annotation by experts.

Although automatic object detection might be feasible for some settings, Citizen Science seems to be a promising alternative for others. Nevertheless, the use of citizen scientists leads to some inaccuracies, i.e. flawed annotations. We conducted a small primer study to analyze typical errors and their margins for CS annotations. Problems such as inaccurate radius and inaccurate position were identified and error margins could be determined. In a more extensive CS study, we were able to confirm these results and simulate CS annotations on a large scale. These enabled us to analyze the impact of flawed CS annotations on the downstream deep learning classification. Some impacts are quite significant, but if the deviations from the expert annotations are not that large, the impacts are manageable.

Contributions

Comparison of deep learning Architectures
Although few papers discussing deep learning applications to marine imaging exist, we are the first to compare an extensive number of networks, including the application of data augmentation and transfer learning. Especially in deep-sea marine imaging, publications are quite rare.

Validation accuracy and number of parameters
The efficiency in using parameters is analyzed for various networks.

How much data is necessary? Deep learning imposes the assumption that enough data is available. However, in marine imaging, annotated data is rather scarce. In addition, how much data is enough? We analyzed how little data is sufficient to likely be able to classify to a certain accuracy and how reliable these results are.

Annotation size We investigated if annotation size matters and what method fits best to incorporate it.

Tackling data scarcity and imbalance In biological images, especially in marine imaging, some classes are scarce and there is generally an imbalance in the distribution of samples to classes. We evaluated several approaches to tackle this problem. In addition, we proposed and evaluated balancing rules, which determine the number of samples per class to improve the classification performance. Besides, we could show that augmentation in combination with sampling strategies provides a significant improvement in classification accuracy.

Concept drift We analyzed different degrees of concept drift and showed that they can be tackled using fine-tuning of a model, even with few annotations from a new dataset. In contrast we could show that retraining is not a viable option, which might be a domain specific finding.

Citizen Science object detection and deep learning In a first-time study, we combine Citizen Science for detecting objects in marine imaging with deep learning classification. Using the CS data, we could simulate enough data to analyze the errors expected to occur in a real-world experiment. Furthermore, we provide recommendations for designing a Citizen Science study, resulting from the results of our error assessment.

Maia We propose MAIA a hybrid approach to effective and efficient object detection using a combination of an autoencoder with a Mask R-CNN network.

5

Annotation assistance systems for marine image informatics

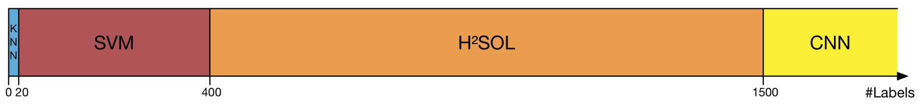


Figure 67: Static structure model for COATL The classifiers used in COATL and the number of annotations when they are switched are shown.

Some of the following results have been published in Langenkämper and Nattkemper, 2016.

Marine image informatics is driven by the need to annotate large amounts of images. With new annotation projects, this ideally happens fully automatically after an initial training phase. In the last section, we analyzed that deep learning-based systems perform this task quite well, but require a reasonable amount of data to predict reliably. In contrast, crafted systems (Section 3) do not perform as well as CNNs but need less data. Therefore we need a means to quickly collect annotations and at the same time, compensate for the error-prone automatic annotations of crafted systems, while the data is still scarce. So if the annotation process cannot be automated from the beginning, it would at least be useful to help the annotators with this time-consuming task.

Staged classifier architecture For classification, we have to create a classifier that is suitable for the diverse data of marine science. However, due to the characteristics of marine data as well as the way it is collected and processed, there is a multitude of different scenarios that can hardly be handled by one single classifier. Therefore, we propose a staged approach, i.e. a variety of classifiers which are specialized in one scenario are employed for a specific range of labeled training data available. The current classifier is switched at a certain point. At the beginning of an annotation process, there are no annotations and no computer assistance can be provided. With few labeled training data, a classifier handling this problem is needed. With a growing number of labeled training data, a simple classifier will soon be overburdened. Switching to a more complex classifier improves classification performance. However, this also has an impact on the runtime. We first employed the staged approach for the Hausgarten dataset in Langenkämper and Nattkemper, 2016, where we proposed a web-based system called COATL. It uses a static structure (see Figure 67), i.e. the amount of data when to switch from one classifier to another is pre-defined. We start with a KNN classifier, which is especially suitable for very few annotations, and switch to an SVM, which performs reasonably well with few data. The runtime of an SVM increases rapidly and so we switch to the H^2SOL until we have enough data to train a CNN architecture.

Digression: COATL mit CNNs

COATL uses a derivative of the Google Inception V1 network [Szegedy, Liu, et al., 2015]. The actual architecture is depicted in Figure 68 and consists of an inception layer, followed by a 2×2 max pooling layer and a local response normalization followed by a 7×7 convolutional layer followed by a 2×2 max pooling layer and two fully connected layers with 1024 and 128 hidden units. Dropout [G. E. Hinton et al., 2012] with a probability of 0.5 is used in between the two hidden layers. Rectified linear units (ReLUs) are used for nonlinearity [Nair and G. E. Hinton, 2010]. ReLUs are applied to the output of each layer. The Adam Optimizer [Kingma and Ba, 2014] is used for training. This rather small network was used because the input images of Hausgarten are quite small. Therefore we experimented with different architectures and the one described above achieved the best results in preliminary studies. Yet nowadays, image resolution is usually higher and thus larger, more sophisticated networks can be applied.

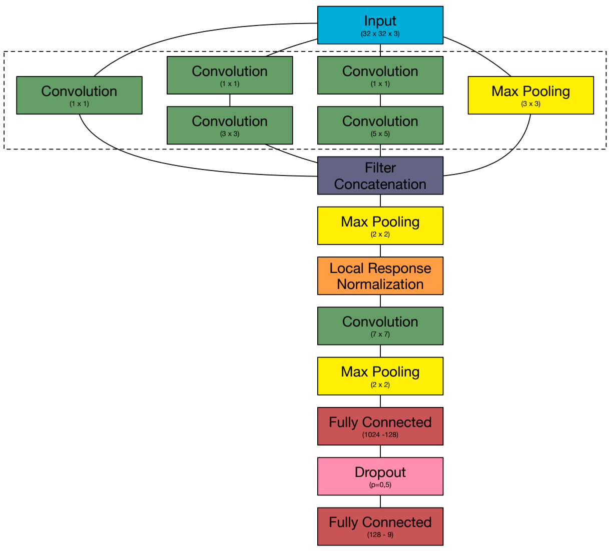


Figure 68: CNN structure as used in COATL The COATL CNN is a simple small convolutional network designed with smaller sized datasets in mind.

Digression: Memory Modules

A big problem with very few data and many classes is that most classes are very rarely presented to the classifier. It is possible for humans to memorize objects from one or very few samples, but for a machine learning system, this is a non-trivial task. The field of research dealing with this problem is called one-shot-learning. A promising algorithm, the memory module that could be used to solve the one-shot-learning problem is described in Ł. Kaiser et al., 2017. We have evaluated memory modules individually and in combination with the H^{2}SOM algorithm by appending memory modules to each neural unit u_j. Appending the modules is done in a similar fashion to attaching local classifiers to the H^{2}SOL. As a dataset for testing, we used the well-established mnist dataset [LeCun and Cortes, 2010]. The class containing the digit zero (6903 samples) was subsampled and the number of memory cells subsequently limited. The results are displayed in Figure 69. Since mnist is a rather easy task and we wanted to employ the Memory Module for very low abundant classes (~10-100 samples), the results are somewhat disappointing, so that we have not pursued this idea further. We either have to use a large amount of memory cells or the results decrease significantly with 0.001% (7 samples) and 0.05% (345 samples) of samples. The results for the H^{2}SOM with memory modules attached can be found in the appendix (cmp. Figure 82). However, the results are quite similar to the ones presented above.

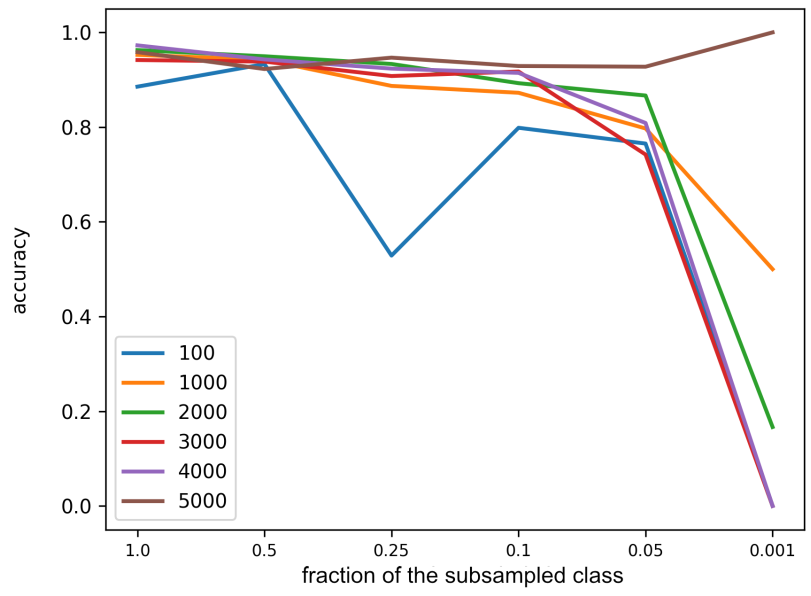


Figure 69: MemCell classification results The Accuracy of the classification of the digit zero in mnist using a memory module with decreasing capacity is shown. Each line represent one experimental run with the number of memory cells depicted in the legend. The fraction of training data for the digit zero of all data of that class in the dataset is shown on the x-axis.

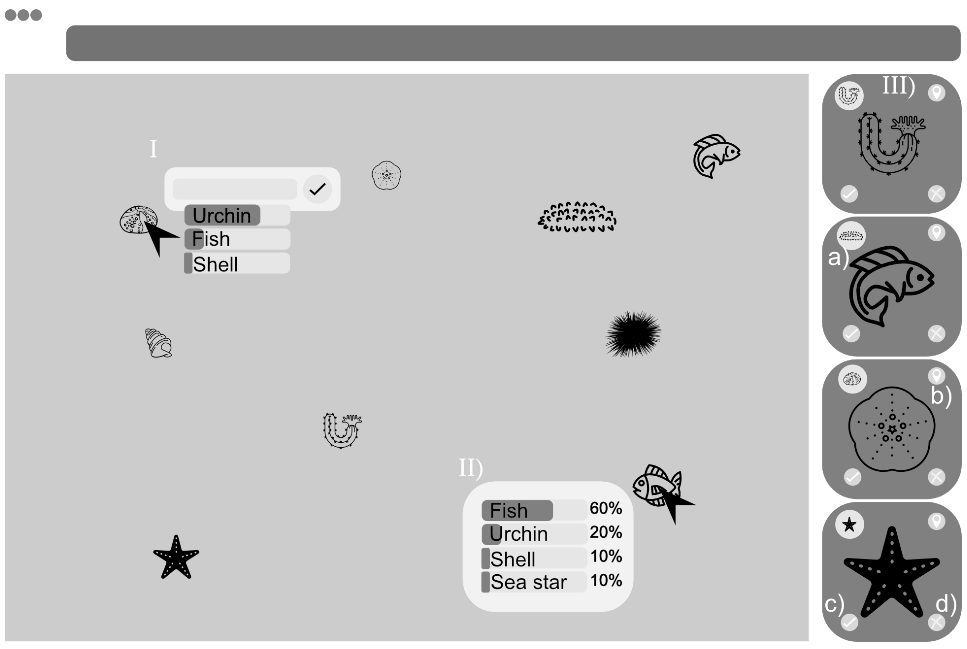


Figure 70: Concept sketch of the COATL system I) In semi-automatic annotation mode objects can be marked and a dialog box pops up proposing multiple classes along with belief values of the underlying classifier. If the correct class is not presented, one can opt to input a class in free text featuring autocompletion. II) A fully automatic mode is provided for users inexperienced in taxonomic assignment, e.g. citizen scientists. In addition, it can be used for convenience when the underlying classifier is reasonably well trained, so that results are to be trusted. The belief values of the classifier are displayed for quality control. III) Detections are shown on the right side of the screen as a list of annotation patches I_j. A proposition using automatic classification is shown in the top left corner *a)*, which can be overruled by the user by clicking on that icon. *b)* Sometimes, it is necessary to show context information to make a decision about the class, i.e. the view of the image is panned to the location of the object and zoomed in to show the object and its surroundings. By clicking on the checkmark *c)* the annotation, i.e. position and class, is accepted or by clicking on the x-mark *d)* discarded.

Assistance system In the beginning, we have to rely on crafted learners and can later switch to powerful CNNs, when enough data is available. In order to compensate for the error-prone crafted learners, we will not automatically classify objects, but rather provide assistance. A sketch of such an annotation assistance system is shown in Figure 70. The annotation aid can be divided into detection (Figure 70 III), semi-automatic (Figure 70 I) and automatic classification (Figure 70 II). This semi-automatic classification assistance (Figure 70 I) provides suggestions for possible classes for an object. These proposals are ranked by the belief value of the underlying classifier. In this way, even if the classification is flawed, the correct class will likely be proposed as one of the most likely classes. This class can now be selected by the user. The class suggestion reduces errors due to typos, or being inattentive [Metz, 1986; Culverhouse et al., 2003], thus also increasing the effectivity. In addition, selecting a class out of a few suggestions is much faster than a manual classification, which usually requires typing or browsing of a label tree. However, if the classifier is unable to predict the correct class, a user can still enter a different class as free-text, supported by autocompletion of all known classes.

The automatic classification assistance (Figure 70 II) is useful for users inexperienced in taxonomic assignment, e.g. citizen scientists. Furthermore, if the underlying classifier is reasonably well trained, the results are trustworthy and it can be used out of convenience.

For the detection task (Figure 70 III) we use the entropy-driven ROI detection presented in Section 3.3 to suggest potential objects to the user. This automatic detection enables faster annotation since the time-consuming task of scanning the image for interesting objects is at least partially eliminated. Even if not all objects are found, it helps to quickly gather annotations and get to the point where more sophisticated techniques such as CNNs can be used for classification or even object detection. At this point, deep learning techniques for full or semi-automatic classification, such as in MAIA [Zurowietz, Langenkämper, Hosking, et al., 2018], or Citizen Science-based efforts (cmp. Section 4.6) can be applied.

The combination of the staged classifier approach and the assistance systems forms a modular adaptive assistance system to significantly improve the annotation throughput and performance of marine scientists as well as citizen scientists. For better availability and easy presentation and collaboration, a web-based system was developed.

Transparent presentation of aids An essential part of the interface design is to think about how the computational assistance is presented to the annotator. The information should not be forced on the annotator, i.e. the annotator should always be able to reject the help and overrule the automatic decision. The information should be quickly accessible and easily perceptible to the user without adversely affecting the decision-making process of the user. When information

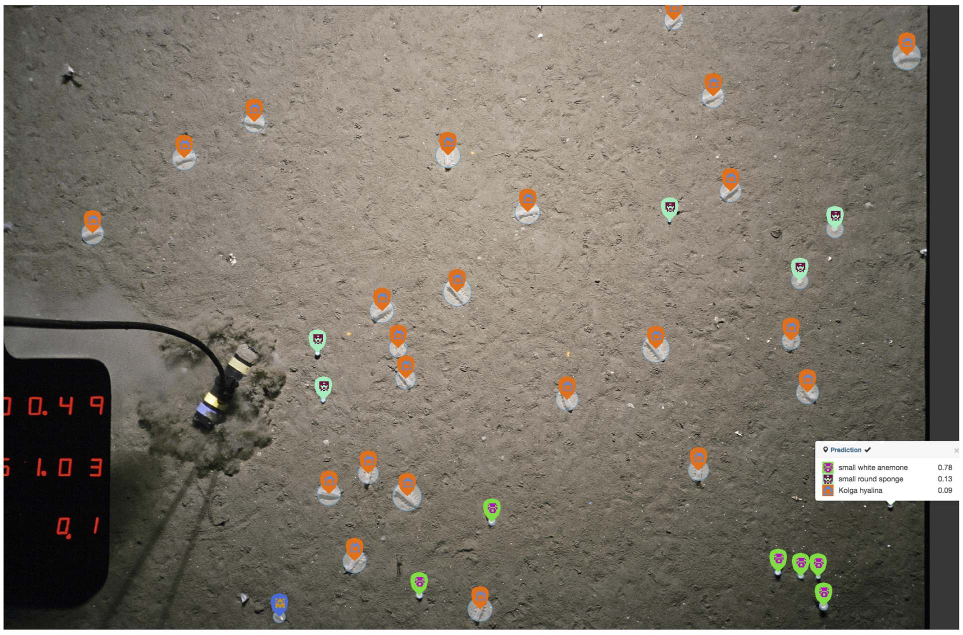
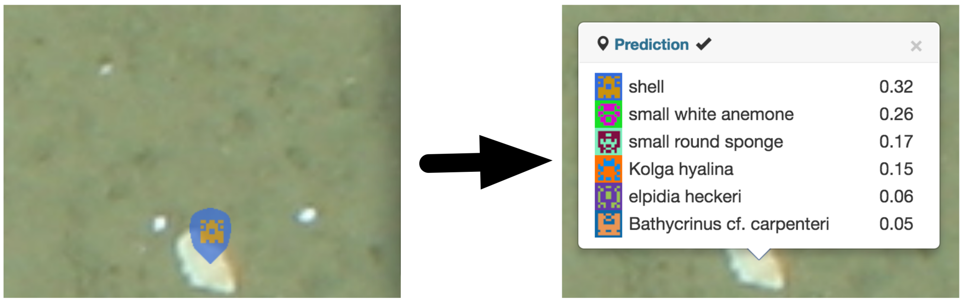


Figure 71: Annotation view of the COATL tool Here annotations can be set and an overview of all manual and (semi-)automatic annotations is provided.

is presented the annotator should, to some extent, be able to judge the confidentiality of that information, e.g. by displaying belief values provided by the underlying classifier (see Figure 72). The opacity of the annotation icon (see paragraph below) is an easily perceptible and recognizable hint of the certainty of the classifier. This helps to quickly assess the quality of the automatic annotation for a larger number of annotations. For an example of the annotation view employed in COATL, see Figure 71.

For the classification, we could automatically propose a label for the manually selected object and the annotator could either accept or reject it. Instead, we opted for assistive annotation. We present an input field featuring autocom-



Prediction ✓

shell	0.32
small white anemone	0.26
small round sponge	0.17
Kolga hyalina	0.15
elpidia heckeri	0.06
Bathycrinus cf. carpenteri	0.05

Figure 72: Prediction view of the COATL tool A click on an icon shows the prediction view with the belief values of the classifier for manual inspection.

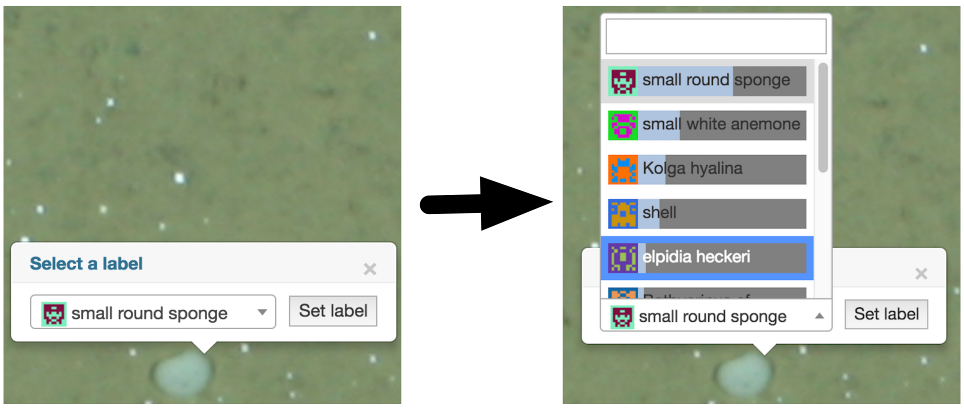


Figure 73: Selection view of COATL When clicking on the combo box all classes predicted by the classifier with their belief values are shown.

pletion of all known labels. The list of labels is sorted by the belief value of the underlying classifier. Because the most likely class is already selected, it is equally fast to the solution where the user just accept or reject a label if the user wishes to use the tool in this way. This is represented by a bar overlaid with the class label. The length of the blue bar represents the classifier's belief value (see Figure 73). This visual hint helps in choosing a class, as the certainty of the classifier is made transparent to the user.

For object detection, a list of small image excerpts with the found objects is presented to the annotator (see Figure 74 top right). It is crucial to be able to zoom in on the selected image to view the excerpts with context. This is important in order to make a more sophisticated decision, i.e. certain organisms only live on specific substrates or in the vicinity of other organisms. The pan and zoom functionality offers the possibility to display the found object in the entire image. An icon depicting the designated class provided by the automatic classification is presented for each object proposal. Clicking it accepts the position of the object, but rejects the designated class and displays the manual annotation dialog (see Figure 73). By clicking on the checkmark or the x-mark, the position and the class are accepted or rejected.

Data-driven icons If data from a previously unknown area has to be annotated, new classes that are not included in the label catalog will likely be observed. Therefore, it is not possible to provide a predefined label catalog, but it has to grow when annotating. If we want to provide icons for each class as a visual cue for easy recognition, this is a severe problem.

The solution is the automatic generation of icons based on the class label Ω. These unique icons are called Identicons (see Figure 75). They are inspired by auto-generated avatars of Gravatar or Github. For each label Ω the md5 hash [Rivest, 1992] of its class name md5(Ω) is computed. The hash length is

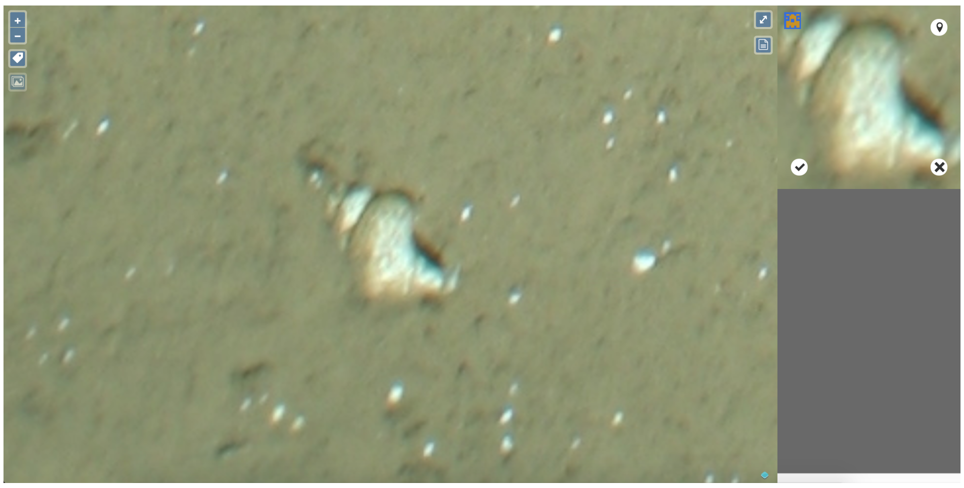


Figure 74: Proposal view of COATL On the right side object detection proposals are shown. By clicking on the pan and zoom icon (top right) the main view is zoomed and panned to the current object proposal for further manual inspection.

128 bit (16 bytes).

First we determine a foreground and a background color for the two-color icon (cmp. Figure 76). The first 3 bytes of md5(Ω) are used to determine a foreground (fg) RGB color $\mathfrak{C}_{\text{fg}} = (R_{\text{fg}}, G_{\text{fg}}, B_{\text{fg}})$ (2^8 intensity values for each $R_{\text{fg}}, G_{\text{fg}}, B_{\text{fg}}$). The background (bg) color is the inverse of the foreground color: $\mathfrak{C}_{\text{bg}} = (2^8 - R_{\text{fg}}, 2^8 - G_{\text{fg}}, 2^8 - B_{\text{fg}})$

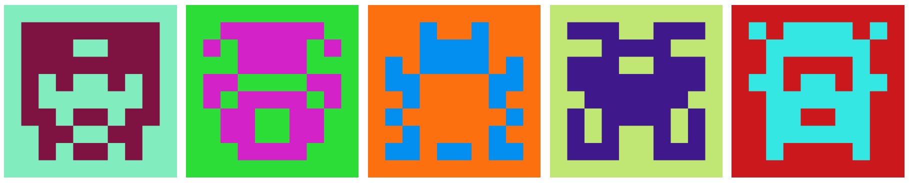
In the following, all bits (including those already used to determine the color) are used to determine whether a pixel is colored as foreground or background to yield an 8×8 pixel large Identicon. For this purpose, every 4 bits are interpreted as a hex digit. These values form a matrix $\hat{Y}$ of size 8×4. Half of the icon Y, denoted as Y' (8×4) is generated as follows:

$$Y' = \begin{cases} R_{\text{bg}}, G_{\text{bg}}, B_{\text{bg}} & \text{for } \hat{Y}_{i,j} \mod 3 > 0, \forall i, j \\ R_{\text{fg}}, G_{\text{fg}}, B_{\text{fg}} & \text{else} \end{cases}$$

The 8×8 Identicon

$$\mathcal{Y}_{i,j} = \begin{cases} Y'_{i,j} & \text{for } j \leq 4 \\ Y'_{i,4-j} & \text{else} \end{cases}$$

is generated by concatenating Y' with its flipped version to obtain a mirror symmetry. Symmetry is a strong cue for the visual system. It helps to recognize structures as one continous object apart from the background [Wertheimer, 1923].



(a) small round sponge **(b)** small white anemone **(c)** *Kolga hyalina* **(d)** *Elpidia heckeri* **(e)** Identicon

Figure 75: Identicons Identicons are a visual representation of an arbitrary word, in our case, an arbitrary label Ω. Here they are displayed as 10×10 pixel squares with a one pixel wide border painted in the background color of the Identicon. In our assistance system, we put the Identicon inside a position marker colored in the background color of the Identicon as can be seen in Figure 71.

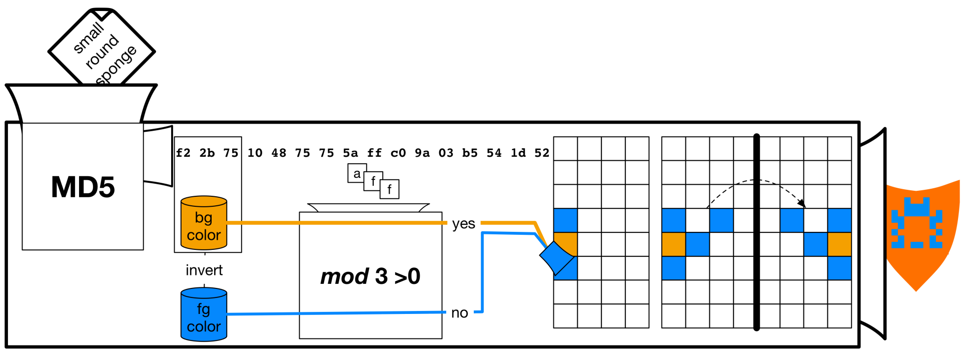


Figure 76: Identicon generation Identicons are generated out of the md5 hash of the label of a class.

5.1 Real-time systems

An essential requirement when working with user interactions is that the system works in real-time, i.e. requests are answered without noticeable delay or at least without impairment of the current task. Without a noticeable delay is a rather vague term. In the literature, we find more detailed descriptions of real-time systems. A real-time system is defined by three components. Time is the most important resource and the task must be handled within a set time frame. The correctness is time-dependent, i.e. an untimely response results in failure for the system. The whole system is to be considered when considering time-limits, e.g. including input/output latencies [Shin and Ramanathan, 1994]. In our case, the time-dependent failure of the system would be that the user will not perceive the system as being responsive enough and thus not helpful. To define an exact time limit is hard but multiple references exist, which provide task-dependent answers [Miller, 1968; Card, Robertson, and Mackinlay, 1991; Newell, 1990]. The literature offers different values for the time-limit, but a lower limit of 2 seconds is recommended to keep the user from straying with one's thoughts. Furthermore, an immediate response to the interaction should be available in no more than 0.1 seconds. In this work, we will speak of a real-time response when the response is below the threshold of $\mathcal{R} = 1$ second, which is also supported by Newell, 1990 as a threshold. We must implement a real-time system since the response time, for the user in our scenario, is rather critical. If the user waits too long for a response from the system, the computer support is perceived as useless because the user feels that his own decision-making process is faster. This limits the means which we can use to realize such a system, e.g. pre-processing of images is not possible because more time would be needed for the image processing operations. In addition, we need to consider further improvements to the runtime performance of our classifiers, as these are the most time-critical parts in our system. There are two basic ways to speed up a computation - acceleration by hardware and acceleration by using algorithmic properties. Acceleration by hardware scales, at least to some extent, with the resources one invests, i.e. the computing system one can afford. Although computer hardware may not be cheap, this is the simplest way to gain performance. In addition, the hardware is improving from year to year in terms of computing power. Acceleration by using algorithmic properties is limited by the creativity and ideas one comes up with and therefore can be harder to achieve. Nevertheless, the expected gains are generally significantly higher.

Acceleration by hardware

There are several means to accelerate a system using hardware. Almost all modern computer systems are multi-core/processor systems. In addition, these systems include vector instruction operations and caching architectures to boost the performance further. In recent years, a special focus has been placed on general-purpose graphics processing unit (gpGPU) computing, i.e. the computation on graphics cards. In machine learning, especially in deep learning, these had a major impact on performance and the possibility to implement modern deep learning algorithms. FPGAs are a special kind of reconfigurable integrated circuit. Although difficult to program and expensive, they seem to be the solution to some bottleneck problems. With the advent of big data and cloud services, computing on large-scale cluster systems became more common and enables scaling of many solutions. In Langenkämper, Jakobi, et al., 2016, we provide an overview of the acceleration using hardware means of some common bioinformatics operations. In Schoening, Langenkämper, et al., 2015 we present a way to speed up H^{2}SOM training and image pre-processing using gpGPU computing. In this work, we will focus on processing on multi-core/processor and gpGPU systems, whereas multi-core/processing is used throughout most algorithms and gpGPU computing is featured by all current deep learning frameworks.

Acceleration by using algorithmic properties

A typical step in accelerating an application is the selection of appropriate algorithms adapted to the problem, which in our case are optimized for compliance with the real-time property. As the classifiers are the most time-critical elements in our system, we present an optimized online version of the H^{2}SOL algorithm.

H^{2}SOL revisited: Online extension to the H^{2}SOL Online learning describes the ability to adapt the model using new data without retraining the whole model. The opposite is batch learning, where all training data is known in advance. Since the training of the H^{2}SOL is quite fast, training the network anew from scratch would be feasible, but then it would rather be a batch algorithm and the benefits of online learning, i.e. saved training time would be gone. Since the H^{2}SOL is trained iteratively, only small changes are necessary to enable online learning.

The most naive way to create an online classifier would be to add additional iterations of training for the new data. The problem with this is that the number of iterations would be unknown upfront and therefore, a reasonable learning rate is not possible to set. Also, the order of input into the H^{2}SOL would be of huge importance as data fed earlier into the classifier would be taken into account more than data presented later. Choosing a constant learning rate would be possible, but would probably lead to poor results. If selected too high, it would never lead to a state of equilibrium, i.e. to a state where an additional iteration would not completely change the results. If the learning rate would be chosen to low, learning would be prolonged and rarer classes would be ignored entirely. Rare classes are presented infrequently so that the adaption of a neural unit u_h towards the rare data point would be so small that it would not make any difference.

Our solution is to use the current model as a starting point and then retrain each ring. We use a decreasing learning rate as usual, fit for the number of data points, which are added in the current online learning iteration, i.e. we act as if only the new data would exist. To compensate for the model entirely adapting to the new data we present alternately new data as well as already known data, i.e. already presented data. This guarantees that the model is not completely rearranged and forgets the old data, but also fits the new data. Instead of $(\mathfrak{x} + \mathfrak{y}) \times N_{\text{epochs}}$ learning steps for training all data in batch learning mode, with $\mathfrak{x}$ being the number of already presented training samples and $\mathfrak{y}$ being the number of new training samples, we need only $2 * \mathfrak{y} \times N_{\text{epochs}}$ iterations.

5.2 Adaptive systems - static vs data-driven model

Initially, we opted for a static model of classifiers, i.e. the classifiers were switched at pre-defined points, based on the experience with these classifiers. However, it might be reasonable to consider the data to be classified and dynamically switch the classifiers based on their and the data's properties. Given a tuple of classifiers $(\mathcal{C}_q)_{q=1,\ldots,Q}$ we now propose different schemes to order these in a meaningful way to optimize the performance, i.e. accuracy $\mathfrak{P}(\mathcal{C}_q)$ or runtime $\mathfrak{R}(\mathcal{C}_q) = \mathfrak{R}^{\mathsf{t}}(\mathcal{C}_q) + \mathfrak{R}^{\mathsf{v}}(\mathcal{C}_q)$ (referred to as training runtime and validation runtime respectively) of the adaptive system. Without loss of generality, let differently parametrized classifiers $\mathcal{C}_q, \mathcal{C}_{\hat{q}}$ with $q \neq \hat{q}$ be different entities in the tuple. The switch of classifiers $\mathcal{C}_q$ to $\mathcal{C}_{\hat{q}}$ is done at a certain threshold $\mathfrak{T}_q$ of available training data Γ^t.

Static (ST) In the static case, the change of different classifiers is pre-defined based on a thorough inspection of the system behavior on out of the bag data, i.e. the classifiers and their order $(\mathcal{C}_q)_{q=1,\ldots,Q}$ as well as the thresholds $\mathfrak{T}_q$ are determined empirically. This scheme can be used effectively when the behavior (runtime and accuracy) of all the classifiers is well-known and robust, i.e. the runtime increases steadily and the performance increases or stagnates at a certain typical point.

Time limited naive (TLN) Assume, the classifiers are sorted from slowest to fastest, i.e. $(\mathcal{C}_q)_{q=1,\ldots,Q}$ with $\mathfrak{R}^{\mathsf{t}}(\mathcal{C}_q) \leq \mathfrak{R}^{\mathsf{t}}(\mathcal{C}_q + 1)$. The slowest classifier is determined as the first classifier. When the amount of annotations and thus data and runtime increase, we want to change the classifier. If $\mathfrak{R}^{\mathsf{t}}(\mathcal{C}_q) > \Theta$ (where Θ is a runtime threshold) the classifier is exchanged for the next fastest classifier $\mathcal{C}_{q+1}$. This scheme is similar to the static scheme, with the exception that the classifier switch is not defined in advance, but is dynamically determined by a time threshold. Since the switch of classifiers is only based on the runtime-limit, accuracy is not taken into account. If the first and slowest classifier never takes longer than the runtime-limit, the classifier may never be switched.

Time limited constrained (TLC) This scheme is like the time limited naive scheme combined with a static constraint of data-availability, i.e. the classifiers are changed independently of the time-limit when a certain pre-defined threshold $\mathfrak{T}_q$ of data is reached. Due to certain classifier properties, a classifier may not violate the runtime threshold Θ, but its performance may decrease, compared to another available classifier, as the amount of data increases. It is therefore necessary to introduce this data-availability constraint to change the classifier in any case.

Time limited constrained for validation and training (TLVT) This scheme is like the time limited naive scheme, but in addition to the training time it also considers the validation time, i.e. if $\mathcal{R}^{\mathsf{t}}(\mathcal{C}_q) > \Theta$ or $\mathcal{R}^{\mathsf{test}}(C_i) > \Theta$ the classifier is exchanged for the next fastest classifier $\mathcal{C}_{q+1}$. Θ can be set to the time-limit for perceiving a system to be responsive, e.g. 1 second (cmp. previous section). Some classifiers have a good training runtime, but the validation runtime does not scale that well. It is therefore important to consider both the training time and the validation time for changing the current classifier.

Background cross-validation (BG) In the background cross-validation case all classifiers perform a cross-validation on the current training set Γ^t. When the background cross-validation is finished, the best-performing classifier is selected, i.e. $\mathcal{C}_q$ with $\arg\max_q \mathfrak{P}(\mathcal{C}_q)$. This scheme is preferred if there is no prior experience of the classifiers on similar data. Furthermore, if the runtime and accuracy do not increase steadily for a classifier, a more data-driven classifier switch is required. Unfortunately, the cross-validation takes a long time. During this time, it is possible that new data is added, while the background cross-validation is still running. For this reason, the data-driven decision for a classifier is not always based on the most current training set Γ^t, especially if data is added in quick succession. Therefore, the decision is prone to be based on out-of-date data.

Training Error based (TE) This scheme works like the background cross-validation, except it uses the training error $\mathfrak{E}(\mathcal{C}_q)$ measured on the training set Γ^t instead of validation accuracy, i.e. the classifier $\mathcal{C}_q$ with $\arg\min_q \mathfrak{E}(C_q)$ is chosen. The training error is the accuracy of the classifier computed on the training data and for most classifiers is computed in any case while training. This scheme exchanges validity with speed in the decision process by not evaluating on the validation data but on the training data. This compensates the problem of being out of date. Unfortunately, the training error $\mathfrak{E}(\mathcal{C}_q)$ is not always a significant measure for determining the best classifier.

Polling (POLL) Simultaneously all classifiers train on the current training set Γ^t and the results of all the classifiers finished at this point are collected and a majority vote of all classifiers $(\mathcal{C}_q)_{q=1,\dots,Q}$ is performed. This scheme combines the predictive power of an ensemble of classifiers, but with the exception that not all classifiers might be finished with training at that point. The design guarantees that the runtime-limit is adhered to. A weak point is that it consumes more computing power because all classifiers have to be trained at the same time. Without parallel processing using multiple processor cores, this approach is not feasible. Another drawback is that with larger training sets Γ^t the fastest classifier might be ready, but slower ones not, so that the majority vote degrades to a single fast classifier that predicts the new data, which might not the best-performing one.

We conducted an experiment that compares the different methods for switching classifiers. Therefore we partitioned the Hausgarten dataset (cmp. Section 2.1) in training set Γ^t and validation set Γ^v and performed a classification experiment. The number of labeled data in the training and validation set was varied analogously to the experiments in Section 4.2, but we choose increments of one label, i.e. $J' \in \{50, 51, \dots, 1500\}$. In addition to accuracy, we also had a look at the training and validation time.

The results are presented in Table 17 and Figure 77. We can see that TLN, TLVT and TE all perform quite similarly. The reason for this is that they did not switch the classifier at all (see appendix Figure 83) and are therefore not suitable for our adaptive system. Hence we exclude them from our further analysis. The best mean and maximum accuracies are achieved by the BG, TLC and POLL schemes. The best mean training time is achieved by ST, closely followed by TLC and POLL. BG has a larger mean training time, even though it stays below 1 second. The best maximum training time is achieved by POLL only slightly above the threshold of 1 second with 1.01 seconds, closely followed by TLC (1.07s) and ST (1.08s). Again, BG requires significantly more time with 2.8 seconds, which unfortunately violates the 1-second threshold. It was expected that BG would require significantly more training time, as multiple trainings and evaluations have to be done for cross-validation instead of training a single, or in the case of POLL multiple classifiers at once. Unfortunately, the clear violation of the 1-second threshold makes the scheme unfit for use in our real-time assistance system. The validation times of ST, TLC and BG all are similar with 0.06 to 0.18 seconds as mean and 0.11 to 0.26 seconds as maximum validation time. POLL is significantly slower with 0.47 seconds as mean and 0.66 seconds as maximum validation time. This was also expected since all classifiers generate a validation at the same time and a majority vote determines the outcome. With all other methods, only a single classifier decides on the results, which is faster by design. POLL also does not exceed the 1-second threshold and can therefore still be used in our assistance system.

dyn test case	mean accuracy	max accuracy	mean training time	max training Time	mean validation time	max validation time
ST	0.71	0.78	0.49	1.08	0.06	0.11
TLC	0.74	0.80	0.55	1.07	0.10	0.23
BG	0.76	0.83	0.99	2.80	0.18	0.26
POLL	0.74	0.80	0.69	1.01	0.47	0.66
(TLN)	0.72	0.78	0.01	0.01	0.26	0.39
(TLVT)	0.72	0.78	0.01	0.01	0.26	0.39
(TE)	0.72	0.78	0.01	0.08	0.27	0.39

Table 17: Static vs. Dynamic experiment. Setups ruled out are in brackets (see text). They are also omitted for the color-coding.

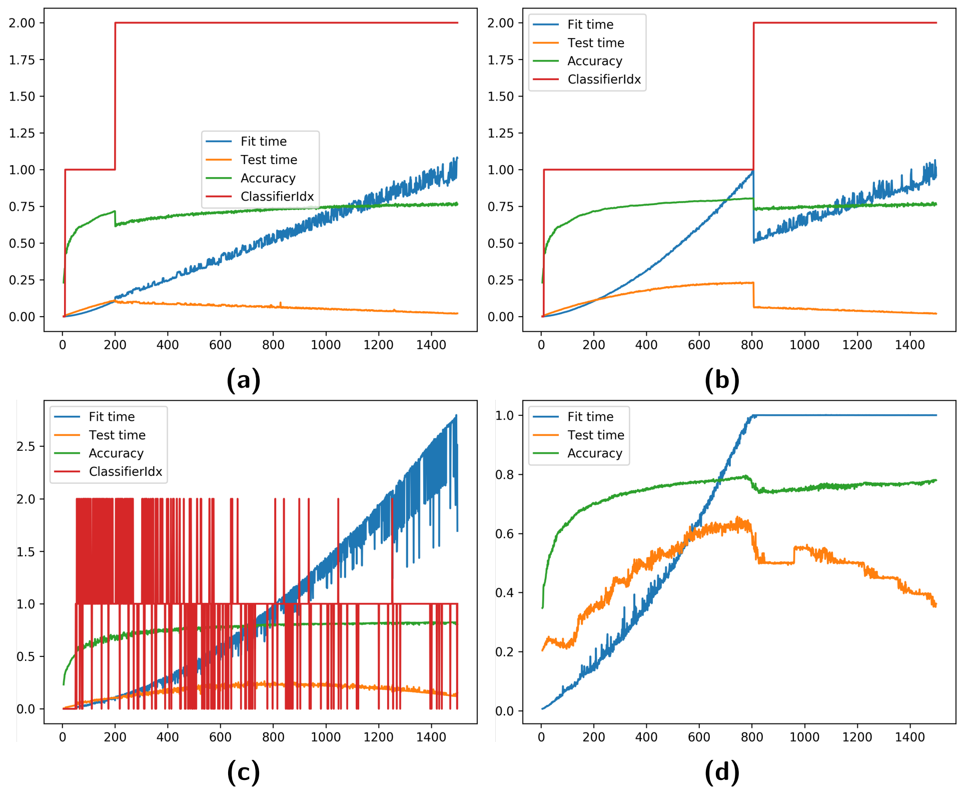


Figure 77: Comparison of different methods for switching classifiers in the staged classifier approach. *a)* Static (ST) as used in COATL *b)* Time limited constrained (TLC) *c)* Background cross-validation (BG) *d)* Polling (POLL)

5.3 Summary and list of contributions

In this section, we presented COATL, an adaptive assistance system to help when starting an annotation project without any pre-existing annotations of that area. When an annotation session is started from scratch, there are no annotations to train a classification system. Not one classifier exists handling few annotations as well as a lot of annotations. Therefore, we propose a staged approach that switches classifiers taking the current amount of annotations into account. Unfortunately, the classification accuracy with little data is not satisfactory. So we choose to provide assistance instead of providing automatic annotation. This way, the user can overrule and control the error-prone computational annotation. Our transparent presentation of aids helps to ensure the acceptance of a classification system, as the decision-making power still resides in the hands of the annotator. If enough annotations are collected, the system can switch to a fairly reliable convolutional neural network for automatic classification. An object detection system provides the annotator with possible regions of interest to quickly increase the number of annotations. Since arbitrary classes can occur during an annotation session, we needed a way to create icons unique to the class on the fly. These icons are necessary for a swift visual inspection of the annotations of one image. We have proposed a method to create Identicons computed from the label of a class.

We also stressed the need for real-time responses (in our case <1 second) to user requests. This time is the upper limit for the approval of systems that are perceived as responsive. We have introduced an online version of the H^2SOL that helps to speed up the learning process in a system with a growing number of annotations.

In addition, we have experimented with various methods to switch classifiers in a data-driven manner instead of statically determining a structure in advance. We presented and compared seven different approaches, with background cross-validation being the best, but exceeding the time-limit for a responsive classifier. The polling-based method and the time limited constrained methods performed only slightly worse but were within the time-limit.

Contributions

Staged classifier architecture

We presented an architecture for changing classifiers during the annotation process, adapting to a growing number of annotations during an annotation session.

COATL – Assistance systems in marine annotation

We proposed COATL, a marine assistance system for annotation, based on our staged classifier architecture. COATL focuses on a transparent presentation of aids and real-time (as defined above $<$1s) responses.

COALTL-CNN

For the aforementioned staged approach, we created a custom CNN model more fit for little data than larger models like ResNet.

Online H^2SOL

As a result of the real-time requirement, we proposed an online learning H^2SOL where it is not necessary to retrain the entire network when new data is presented. Rather, the network can gradually adapt to new data.

Data-driven adaptive systems

We proposed and evaluated seven different methods to change a classifier, adapting to a growing number of annotations and the need to process queries in less than one second.

Data-driven icons

Inspired by Gravatar and similar data-driven icons, we proposed a method on how to generate so-called Identicons out of the label of an arbitrary class.

6 Conclusion

In this thesis, we have addressed one of the most pressing problems of modern marine image informatics and marine science in general, i.e. annotating huge amounts of image data to gain information of this otherwise lost data.

To solve this problem, computer vision techniques can be used. Computer vision has been dominated by deep learning applications in recent years, with huge success. We have analyzed whether this success can be transferred to the domain of marine image informatics.

Automatic annotation consists of automatic detection of objects and their automatic classification. This thesis mostly researches the automatic classification of already found objects but provides some solutions and interfaces to (semi-)automatic detection.

Automatic detection of objects is possible, as we have shown in Section 4.6, Digression. Unfortunately, it is still error-prone and requires quite a bit of manual input. As an alternative, we proposed a Citizen Science-based approach (cmp. Section 4.6) to object detection. We could show that deep learning architectures can potentially handle error-prone Citizen Science detections. However, the citizen scientists have to be carefully instructed to keep the errors within a manageable margin.

Automatic benthic marine image classification can be performed with reasonable accuracy of over 95% using deep learning. The increase in accuracy compared to the shallow learners is 34 basis points for the PAP dataset and 12 basis points for the Hausgarten dataset. InceptionResNet with transfer learning and MobileNet with transfer learning achieved excellent results with all three datasets. However, MobileNet needs one order of magnitude less training time compared to InceptionResNet. A maximum accuracy of 88% could be achieved on the non-benthic UVP5 data set, which corresponds to an increase of about 50 basis points compared to the shallow learners.

In order to efficiently use these powerful deep learning networks on marine imagery, some more or less domain-specific problems have to be addressed. Marine image data is often imbalanced, i.e. the classes are not equally abundant. We addressed this with several different sampling solutions, as shown in Section 4.4. We report an increase in average classification performance of 13 basis points and a maximum of 40 basis points for a single class improvement. Furthermore, we analyzed how to deal with different sized objects in deep learning applications (cmp. Section 4.3). In Section 4.5 we analyzed concept drift introduced by data acquisition at different time points. Using fine-tuning with 20% of additional data, we could increase the accuracy of deep learning classification from 0.18%, without adaption, to 85%. However, the most prominent problem with deep learning is that enough annotated data has to be present to train the classifiers adequately. There are no hard boundaries stating on how much data is enough for training. Nevertheless, we could show that results comparable to competing approaches, i.e. shallow learners, can be achieved at relatively few data points (~500). However, the shallow learners are much more robust at this point, i.e. their accuracy does not fluctuate that much as that of the deep learners' with a comparable amount of training data does. In our experiments at about 1000 data points, deep learners are robust enough to surpass the shallow learners consistently.

If no pre-existing dataset of the area of interest is available, which could be used as a starting point using the methods shown to compensate concept drift in Section 4.5, we need another method to quickly collect enough annotations. To quickly reach the amount of annotations where deep learning can be reliably applied, we decided to help the annotator by implementing an assistance system. It uses a staged approach combining shallow learners with deep learners (cmp. Section 5). Instead of providing fully automatic annotation, we decided to provide annotation assistance at this point to overcome the error-prone annotations of shallow learners. This means that we only provide suggestions to the user and he has to decide to accept or discard them. With few data points, different shallow learners can be trained. These are switched to always train the one, fit for the currently available amount of data. The shallow learners predict the probability for a class. These probabilities are communicated to the user, who then can decide on an annotation based on the aid provided by the annotation assistance. When enough annotations are gathered, the system can switch to automatic classification in order to quickly annotate huge amounts of images. In Section 5 we discussed at which point to switch the classifiers in our staged classifier approach in order to create a real-time annotation assistance system. To increase the efficiency of the system further, we developed the H^2SOL, a supervised learning algorithm, related to the H^2SOM clustering algorithm (cmp. Section 3.4). It is competitive to the prominent SVM in accuracy, if enough training data is present, but is orders of magnitude faster than the SVM in training and classification. An online learning variant is provided as well, making it superior to the SVM for our use case.

Nomenclature

$(\mathcal{C}_q)_{q=1,\dots,Q}$	A tuple f classifiers.
(N', M')	Image size of a sub-image.
β	A threshold for the rejection classification function in the H^{2}SOM classifier.
Δ	A metric, if not stated otherwise the Euclidean metric.
$\delta_{100}, \delta_{50}, \delta_{50,75,100}$	Balancing rules used for oversampling.
ϵ^t	A time step dependent learning rate.
ϵ_h	A learning rate attached to the neural unit with index h.
Γ	The set of all features. $\Gamma = \{x_{j=0,\dots,J}\}$.
γ	SVM kernel coefficient.
Γ^t	The set of all training features.
Γ^v	The set of all validation features.
$\hat{\delta}_{75}, \hat{\delta}_{50,100}$	Balancing rules used for combined oversampling and undersampling.
$\hat{\varrho}_j$	A density distribution yielded by normalizing ϱ_j.
$\hat{n}, \hat{m}$	Position of simulated CS data, analog to n,m.
$\hat{r}$	Radius of the simulated CS data, analog to r.
Λ	The set of all annotations $\mathcal{A}_j$. $\Lambda = \{\mathcal{A}_{j=0,\dots,J}\}$.
Λ^t	The training annotation set.
Λ^v	The validation annotation set.
$\Lambda^{\mathcal{CSP}}$	The set of all citizen scientist annotations.
λ^{usm}	A parameter for unsharp masking controlling the influence of the masked image.
$\mathcal{A} = (n, m, \mathcal{I}_i, \Omega)$	An annotation which is defined as a tuple of the position n, m in an image $\mathcal{I}$ with height M and width N with a label Ω.
$\mathcal{A}'_j = (n, m, r, \mathcal{I}_i, \Omega)$	Square annotation.

$\mathcal{A}_{\hat{j}}^{C} = (x^C, y^C, r^C, I_n)$	Citizen Science annotation with symbols defined analog to $\mathcal{A}$
$\mathcal{B}$	Batch size.
$\mathcal{D}$	Dimensionality/depth/number of filters for CNNs.
$\mathcal{H}$	Height of a CNN filter.
$\mathcal{I} \in [0, 255]^{M \times N \times 3}$	An image with height M and width N and three color channels.
$\mathcal{I}_{(m,n)}$	The value of the pixel at location (m, n) in the Image $\mathcal{I}$
$\mathcal{T}$	The fraction of the most entropic pixels to retain for the entropy-driven ROI detection.
$\mathcal{V}$	A volume, i.e. a set of images. $\mathcal{V} = \{I_{i=0,\ldots,\mathsf{I}-1}\}$.
$\mathcal{W}$	Width of a CNN filter.
$\hat{\mathfrak{n}}$	A neighborhood function refined for the non-Euclidean H^{2}SOM/H^{2}SOL.
$\mathfrak{A}_p^o = \{(\hat{n}, \hat{m}, r, \mathcal{I}_i, \Omega)\}$	Simulated artificial annotations suffering from inaccurate position, with o being the minimum overlap with the original annotation $\mathcal{A}$ in percent.
$\mathfrak{A}_r^s = \{(\hat{n}, \hat{m}, r, \mathcal{I}_i, \Omega)\}$	Simulated artificial annotations suffering from inaccurate radius, with s being the over/underestimation of radius r in percent.
$\mathfrak{c}(x_j)$	A classification function to label an unknown feature vector x_j.
$\mathfrak{I}_j$	An annotation patch belonging to annotation $\mathcal{A}_j$.
$\mathfrak{k}$	Number of folds in k-fold cross-validation
$\mathfrak{l}$	A labeling function for the H^{2}SOM classifier.
$\mathfrak{n}$	A neighborhood function in the SOM
$\mathfrak{P}(\mathcal{C}_q)$	Classification performance of classifier $\mathcal{C}_q$, expressed as accuracy if not specified otherwise.
$\mathfrak{r}$	Number of rings in an H^{2}SOM/H^{2}SOL.
$\mathfrak{R}(\mathcal{C}_q)$	Runtime of classifier $\mathcal{C}_q$
$\mathfrak{R}^{\mathsf{t}}(\mathcal{C}_q)$	Training runtime of classifier $\mathcal{C}_q$
$\mathfrak{R}^{\mathsf{v}}(\mathcal{C}_q)$	Validation runtime of classifier $\mathcal{C}_q$

$\mathfrak{r}_e$	The e-th ring for square median features/super square features.
$\mathfrak{r}_e$	The e-th ring of an image for the square median features/super square features.
$\mathfrak{T}_q$	Threshold for changing from classifier q to $\hat{q}$.
$\mathfrak{w}$	Beam search width for the H^{2}SOM/H^{2}SOL.
$\mathfrak{x}$	Number of already presented training samples for batch/online learning
$\mathfrak{y}$	Number of new training samples for batch/online learning
μ	Threshold for threshold adjacency statistics.
Ω	A label.
$\Omega(x_k)$	The class of data sample x_k.
$\Omega^{(l)}$	The l-th class.
Ω_{maj}	The majority class, i.e. the most common class.
Ω_j	Label of feature x_j
$\Omega_\emptyset$	A special rejection label, meaning that the datapoint assigned with this label is left unclassified.
π_k	A permutation used for the K-Nearest Neighbor definition.
σ	The neighborhood adaption modifier used in the neighborhood function of the H^{2}SOM/H^{2}SOL.
I	The number of images.
θ_h	Component of the linear mapping in the H^{2}SOL algorithm for a neural unit with index h.
ϱ_j	Ratio of the majority data samples in the neighborhood of x_j.
$\{(u_h, z_h)_{h=1,\ldots,H}\}$	A set of neurons for an H^{2}SOM/H^{2}SOL.
a	Tuple of numbers (0,. . . , J) used for K-Nearest Neighbor definition.
A'	Set of protocol-corrected expert annotations.
A_h	Component of the linear mapping in the H^{2}SOL algorithm for a neural unit with index h.
B	The number of bins in a histogram.
C	Penalty parameter for the SVM, also known as slack.

$c_0, \ldots, c_{31}$	The quantified colors for the color structure descriptor in MPEG-7.
e	The ring index of the square median features/super square features r_e.
e	The ring index used for the square median features/super square features.
$f(\mathtt{I}_j)$	A feature extraction function applied to an annotation patch $\mathtt{I}_j$.
$g_{0,\ldots,O_g}$	Distinct gray-levels.
H	The number of neurons in an H^{2}SOM/H^{2}SOL.
h	The neuron index.
i	The image index.
J	The number of annotations.
j	The annotation index.
J'	Total number of annotations of the subset of all annotations used for evaluation the number of annotations needed for classification.
K	The number of nearest-neighbors to consider for a K-Nearest Neighbor classifier.
k	Nearest neighbor index.
l	The class index.
L'	Number of minority classes.
O_g	Number of distinct gray-levels.
P	Precision.
p	The number of sampling points for Local Binary Patterns.
$p(I_i)$	A pre-processing function applied to an image I_i.
P_{macro}	Macro-averaged precision.
Q	Number of classifiers for the adaptive systems.
q	Classifier index.
R	Recall.
r	The radius, i.e. half of the side length, of the square annotation.
r^{lbp}	Local Binary Patterns parameter determining the radius of the circle where samples are taken.

R_{macro}	Macro-averaged recall.
s	Number of neighbors of each neural unit.
$s(\Omega^{(l)})$	The number samples of class $\Omega^{(l)}$.
$s'(\Omega^{(l)})$	The number of new samples to be generated for class $\Omega^{(l)}$.
t	The time step or iteration t.
u_h	The neural unit of a neuron in an H^{2}SOM/H^{2}SOL.
V	A Voronoi cell.
x_j	A feature computed for annotation $\mathcal{A}_j$.
x'_j	A synthetic sample.
x_k	Data sample, which is a K-Nearest Neighbor to x_j.
$y(\Omega_j)$	The One-Hot-encoding of the label of the feature x_j.
z_h	The grid coordinate of a neuron in an H^{2}SOM/H^{2}SOL.
Z_n, Z_m	Random numbers generated for simulated CS data.
Z_r	Random number generated for the simulated CS data.

7 Appendix

Data: SO242 The SO242 dataset is a subset of 809 images with a size of 4096×3072 pixels extracted from the SO242/1_83-1_AUV10 survey using the AUV Abyss [Greinert et al., 2017; Greinert, 2015]. The images were taken in the Pacific near the location of APEI6 (see Figure 8). The survey images were taken at a depth between 3420m and 4140m with a target distance of 7.5m to the seabed. The camera was a Canon EOS 6D with a Canon Fisheye 8-15mm lens to capture a wide field of view. As described in Greinert et al., 2017 the images have been undistorted using the biasproject tool by Geomar. No annotations for this dataset were available.

Data: JC77 The images of JC77 were taken in the Central North Sea near the Sleipner CO_2 storage site. The dataset consists of 6321 images of sizes 2448×2048 pixels and were captured at a depth of 77m with a distance of approximately 3m to the seafloor. Only a subset of 50 images with two classes – shell and animal – were annotated, with a total of 278 annotations [Zurowietz, Langenkämper, Hosking, et al., 2018].

	KNN			H^2SOL			H^2SOMClassifier			H^2SOMClassifier(NbrS)			H^2SOMClassifier(Rej)			SVM		
Class	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score
small round sponge	0.76	0.89	0.82	0.82	0.83	0.82	0.78	0.85	0.82	0.79	0.84	0.82	0.82	0.87	0.85	0.85	0.9	0.88
Kolga hyalina	0.73	0.77	0.75	0.72	0.71	0.71	0.57	0.64	0.61	0.54	0.64	0.58	0.6	0.64	0.62	0.8	0.86	0.83
Elpidia heckeri	0.81	0.8	0.81	0.81	0.83	0.82	0.73	0.77	0.75	0.74	0.76	0.75	0.77	0.79	0.78	0.82	0.88	0.85
Bathycrinus cf. carpenteri	0.83	0.82	0.83	0.79	0.79	0.79	0.67	0.71	0.69	0.67	0.69	0.68	0.68	0.77	0.73	0.88	0.81	0.84
bathycrinus stalks	0.58	0.41	0.48	0.44	0.39	0.42	0.53	0.2	0.29	0.6	0.18	0.27	0.56	0.22	0.32	0.45	0.51	0.48
burrowing purple anemone	0.88	0.85	0.87	0.88	0.85	0.87	0.83	0.89	0.86	0.83	0.89	0.86	0.88	0.88	0.88	0.95	0.93	0.94
small white anemone	0.92	0.84	0.88	0.86	0.84	0.85	0.9	0.8	0.85	0.89	0.82	0.85	0.92	0.86	0.89	0.94	0.86	0.9
shell	0.9	0.89	0.89	0.72	0.82	0.77	0.73	0.8	0.77	0.7	0.8	0.75	0.79	0.84	0.81	0.86	0.88	0.87
shrimp	1.0	0.9	0.95	1.0	0.95	0.98	1.0	0.9	0.95	1.0	0.9	0.95	1.0	1.0	1.0	1.0	1.0	1.0
avg	0.83	0.83	0.83	0.8	0.8	0.8	0.77	0.77	0.77	0.77	0.77	0.76	0.8	0.8	0.8	0.87	0.86	0.87

Table 18: Hausgarten - Performance comparison of the classifiers presented in this section on the Hausgarten dataset.

	KNN			H^2SOL			H^2SOMClassifier			H^2SOMClassifier(NbrS)			H^2SOMClassifier(Rej)			SVM		
Class	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score
Amperima	0.56	0.77	0.65	0.67	0.81	0.73	0.5	0.71	0.59	0.47	0.7	0.56	0.51	0.73	0.6	0.83	0.62	0.71
cnidaria 10	0.39	0.53	0.45	0.62	0.43	0.51	0.27	0.02	0.04	0.39	0.02	0.04	0.5	0.03	0.05	0.4	0.64	0.49
cnidaria 16	0.59	0.81	0.68	0.71	0.67	0.69	0.57	0.63	0.6	0.57	0.63	0.6	0.59	0.71	0.65	0.63	0.8	0.7
cnidaria 2	0.53	0.63	0.57	0.66	0.76	0.71	0.44	0.56	0.49	0.42	0.55	0.47	0.44	0.57	0.5	0.75	0.49	0.59
cnidaria 8	0.59	0.35	0.44	0.74	0.42	0.53	0.39	0.37	0.38	0.43	0.28	0.34	0.4	0.4	0.4	0.48	0.59	0.53
cnidaria 9	0.23	0.18	0.2	0.47	0.09	0.15	0.29	0.0	0.0	0.0	0.0	0.0	0.33	0.0	0.01	0.13	0.68	0.22
crinoid 1	0.13	0.05	0.07	0.13	0.02	0.03	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.15	0.49	0.23
crinoid 2	0.44	0.46	0.45	0.51	0.37	0.42	0.24	0.06	0.1	0.32	0.04	0.08	0.28	0.07	0.11	0.4	0.62	0.49
Echiura	0.39	0.32	0.35	0.57	0.2	0.29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.29	0.45	0.35
Foraminifera	0.55	0.4	0.46	0.62	0.45	0.52	0.52	0.16	0.25	0.51	0.11	0.18	0.53	0.17	0.25	0.52	0.57	0.54
Oneirophanta	0.67	0.53	0.59	0.63	0.42	0.5	0.22	0.24	0.23	0.18	0.12	0.14	0.24	0.26	0.25	0.47	0.59	0.52
ophiuroidea disk	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.02	0.22	0.03
Ophiuroidea	0.68	0.64	0.66	0.69	0.82	0.75	0.58	0.66	0.62	0.57	0.65	0.61	0.59	0.66	0.62	0.82	0.35	0.49
Peniagone	0.37	0.05	0.09	0.12	0.01	0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.13	0.37	0.2
polychaete 1	0.55	0.26	0.36	0.36	0.11	0.17	0.27	0.04	0.07	0.31	0.04	0.06	0.28	0.05	0.08	0.33	0.48	0.39
Porifera	0.32	0.13	0.18	0.29	0.07	0.12	0.25	0.02	0.04	0.24	0.02	0.03	0.24	0.02	0.04	0.23	0.34	0.27
Pseudostichopus villosus	0.36	0.16	0.22	0.45	0.22	0.3	0.14	0.03	0.05	0.2	0.02	0.04	0.14	0.03	0.05	0.39	0.47	0.42
stalked tunicate	0.69	0.35	0.47	0.53	0.53	0.53	0.32	0.4	0.36	0.29	0.32	0.31	0.35	0.41	0.38	0.51	0.59	0.55
Tunicata	0.51	0.19	0.28	0.4	0.12	0.19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.26	0.45	0.33
avg	0.56	0.57	0.55	0.62	0.66	0.62	0.46	0.5	0.45	0.44	0.48	0.43	0.47	0.51	0.46	0.68	0.49	0.54

Table 19: PAP - Performance comparison of the classifiers presented in this section on the PAP dataset.

Table 20: UVP5 - Performance comparison of different classifiers on the UVP5 dataset.

	KNN			H^2SOL			H^2SOMClassifier			H^2SOMClassifier(NbrS)			H^2SOMClassifier(Rej)			SVM		
Class	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score
art bubbles	0.14	0.28	0.18	0.19	0.17	0.18	0.16	0.12	0.14	0.15	0.11	0.13	0.13	0.08	0.1	0.19	0.19	0.19
art turb	0.83	0.95	0.89	0.81	0.96	0.88	0.82	0.92	0.87	0.82	0.93	0.87	0.86	0.94	0.9	0.86	0.94	0.9
det aggr dark spherical	0.18	0.38	0.24	0.24	0.31	0.27	0.16	0.22	0.19	0.16	0.22	0.18	0.16	0.22	0.19	0.21	0.44	0.29
det aggregate dark	0.07	0.12	0.09	0.12	0.06	0.08	0.05	0.02	0.02	0.05	0.02	0.02	0.05	0.02	0.02	0.1	0.02	0.04
det aggregate light	0.09	0.11	0.1	0.09	0.05	0.07	0.1	0.05	0.06	0.1	0.04	0.06	0.09	0.05	0.06	0.2	0.01	0.02
det aggregate spherical	0.58	0.73	0.64	0.44	0.82	0.57	0.41	0.78	0.54	0.41	0.79	0.54	0.42	0.78	0.54	0.48	0.84	0.61
det feces like fish	0.13	0.2	0.16	0.17	0.13	0.15	0.13	0.14	0.13	0.13	0.14	0.13	0.14	0.15	0.14	0.22	0.1	0.14
det feces like munida	0.17	0.24	0.2	0.22	0.25	0.23	0.17	0.24	0.2	0.18	0.24	0.21	0.19	0.24	0.21	0.15	0.11	0.13
det feces stick	0.09	0.07	0.08	0.14	0.08	0.11	0.14	0.05	0.07	0.14	0.05	0.07	0.15	0.05	0.07	0.1	0.02	0.03
det fiber	0.19	0.14	0.16	0.18	0.19	0.19	0.21	0.23	0.22	0.21	0.24	0.22	0.22	0.24	0.23	0.23	0.38	0.29
phyto tricho puff	0.18	0.15	0.16	0.2	0.29	0.24	0.16	0.35	0.22	0.16	0.35	0.22	0.16	0.35	0.22	0.16	0.33	0.22
phyto tricho tuft	0.22	0.11	0.15	0.18	0.22	0.2	0.18	0.17	0.18	0.17	0.18	0.17	0.19	0.18	0.19	0.21	0.14	0.17
pro rhizaria other	0.1	0.05	0.07	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.05	0.25	0.08
pro rhizaria radiolaria collodaria solitary black	0.31	0.19	0.23	0.26	0.3	0.27	0.23	0.2	0.21	0.23	0.2	0.21	0.23	0.17	0.19	0.31	0.34	0.33
zoo crust cop	0.1	0.03	0.05	0.15	0.11	0.12	0.13	0.07	0.09	0.13	0.07	0.09	0.14	0.08	0.1	0.11	0.03	0.05
zoo crust cop like	0.12	0.04	0.06	0.09	0.06	0.07	0.08	0.04	0.05	0.08	0.04	0.05	0.08	0.04	0.06	0.09	0.02	0.03
zoo crust shrimp like	0.48	0.22	0.3	0.4	0.31	0.35	0.4	0.3	0.34	0.39	0.29	0.33	0.48	0.26	0.34	0.4	0.47	0.43
zoo gel carn medusa	0.17	0.05	0.08	0.17	0.12	0.14	0.16	0.08	0.11	0.16	0.09	0.11	0.11	0.05	0.07	0.22	0.07	0.11
zoo moll	0.31	0.15	0.2	0.27	0.41	0.32	0.21	0.38	0.27	0.21	0.38	0.27	0.21	0.4	0.28	0.23	0.45	0.31
avg	0.23	0.23	0.22	0.23	0.26	0.24	0.21	0.24	0.21	0.21	0.24	0.21	0.21	0.24	0.21	0.24	0.27	0.23

Table 21: UVP5 - Performance comparison of different classifiers on the UVP5 dataset.

Class	KNN			H^2SOL			H^2SOMClassifier			H^2SOMClassifier(NbrS)			H^2SOMClassifier(Rej)			SVM		
	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score	precision	recall	F_1-score
art bubbles	0.15	0.26	0.19	0.26	0.17	0.2	0.17	0.08	0.11	0.17	0.08	0.1	0.11	0.04	0.06	0.2	0.05	0.08
art turb	0.48	0.78	0.59	0.5	0.74	0.6	0.42	0.61	0.5	0.42	0.58	0.49	0.42	0.64	0.51	0.38	0.85	0.53
det aggr dark spherical	0.45	0.54	0.49	0.45	0.6	0.51	0.33	0.56	0.41	0.32	0.56	0.4	0.33	0.58	0.42	0.38	0.55	0.45
det aggregate dark	0.17	0.22	0.19	0.21	0.17	0.19	0.14	0.09	0.11	0.14	0.09	0.11	0.13	0.07	0.09	0.17	0.11	0.14
det aggregate light	0.35	0.45	0.39	0.39	0.38	0.39	0.29	0.33	0.31	0.29	0.33	0.31	0.27	0.3	0.28	0.43	0.36	0.39
det aggregate spherical	0.79	0.92	0.85	0.71	0.91	0.8	0.67	0.91	0.77	0.67	0.89	0.76	0.69	0.91	0.78	0.51	0.9	0.65
det feces like fish	0.3	0.36	0.33	0.33	0.34	0.33	0.23	0.28	0.25	0.24	0.28	0.26	0.23	0.29	0.25	0.23	0.24	0.23
det feces like munida	0.27	0.34	0.3	0.28	0.3	0.29	0.22	0.27	0.24	0.22	0.27	0.24	0.23	0.27	0.25	0.23	0.17	0.19
det feces stick	0.23	0.2	0.21	0.21	0.14	0.17	0.22	0.14	0.17	0.21	0.14	0.17	0.22	0.14	0.17	0.23	0.11	0.14
det fiber	0.28	0.25	0.26	0.3	0.32	0.31	0.28	0.32	0.3	0.27	0.32	0.29	0.28	0.34	0.31	0.26	0.27	0.27
phyto tricho puff	0.29	0.39	0.33	0.29	0.34	0.32	0.24	0.32	0.27	0.24	0.32	0.27	0.24	0.31	0.27	0.23	0.32	0.27
phyto tricho tuft	0.31	0.25	0.28	0.26	0.21	0.24	0.26	0.2	0.23	0.26	0.2	0.23	0.23	0.16	0.19	0.32	0.13	0.19
pro rhizaria other	0.25	0.19	0.22	0.38	0.15	0.22	0.14	0.04	0.06	0.14	0.04	0.06	0.16	0.04	0.07	0.14	0.32	0.2
pro rhizaria radiolaria collodaria solitary black	0.35	0.24	0.29	0.3	0.28	0.29	0.23	0.21	0.22	0.23	0.2	0.21	0.24	0.18	0.2	0.28	0.14	0.19
zoo crust cop	0.16	0.06	0.08	0.17	0.11	0.13	0.13	0.08	0.1	0.13	0.08	0.1	0.14	0.08	0.1	0.18	0.14	0.16
zoo crust cop like	0.19	0.07	0.1	0.12	0.09	0.1	0.15	0.07	0.09	0.15	0.07	0.09	0.14	0.07	0.09	0.16	0.05	0.08
zoo crust shrimp like	0.34	0.24	0.28	0.27	0.26	0.26	0.25	0.19	0.21	0.24	0.17	0.2	0.26	0.2	0.22	0.19	0.6	0.29
zoo gel carn medusa	0.27	0.09	0.13	0.21	0.2	0.21	0.19	0.14	0.16	0.18	0.14	0.16	0.18	0.13	0.15	0.2	0.1	0.13
zoo moll	0.54	0.48	0.51	0.39	0.55	0.46	0.3	0.42	0.35	0.29	0.41	0.34	0.3	0.44	0.36	0.33	0.45	0.38
avg	0.33	0.34	0.32	0.32	0.34	0.32	0.26	0.29	0.27	0.26	0.29	0.26	0.26	0.3	0.27	0.28	0.3	0.26

Table 22: UVP5 - Performance comparison of different classifiers on the UVP5 dataset using SHF18 features.

	Entire network	Last layer
DenseNet	0.95	0.75
Inception	0.95	0.93
InceptionResNet	0.95	0.85
MobileNet	0.95	0.34
NASNetMobile	0.94	0.01
ResNet	0.93	0.86
VGG16	0.86	0.87
Xception	0.95	0.01

Table 23: **Comparison of different transfer learning methods** Entire network refers to presetting the weights with the ones learned from ImageNet and retraining of all layers. Last layer refers to presetting the weights with the ones learned from imagenet but only retraining the last fully connected layer(s).

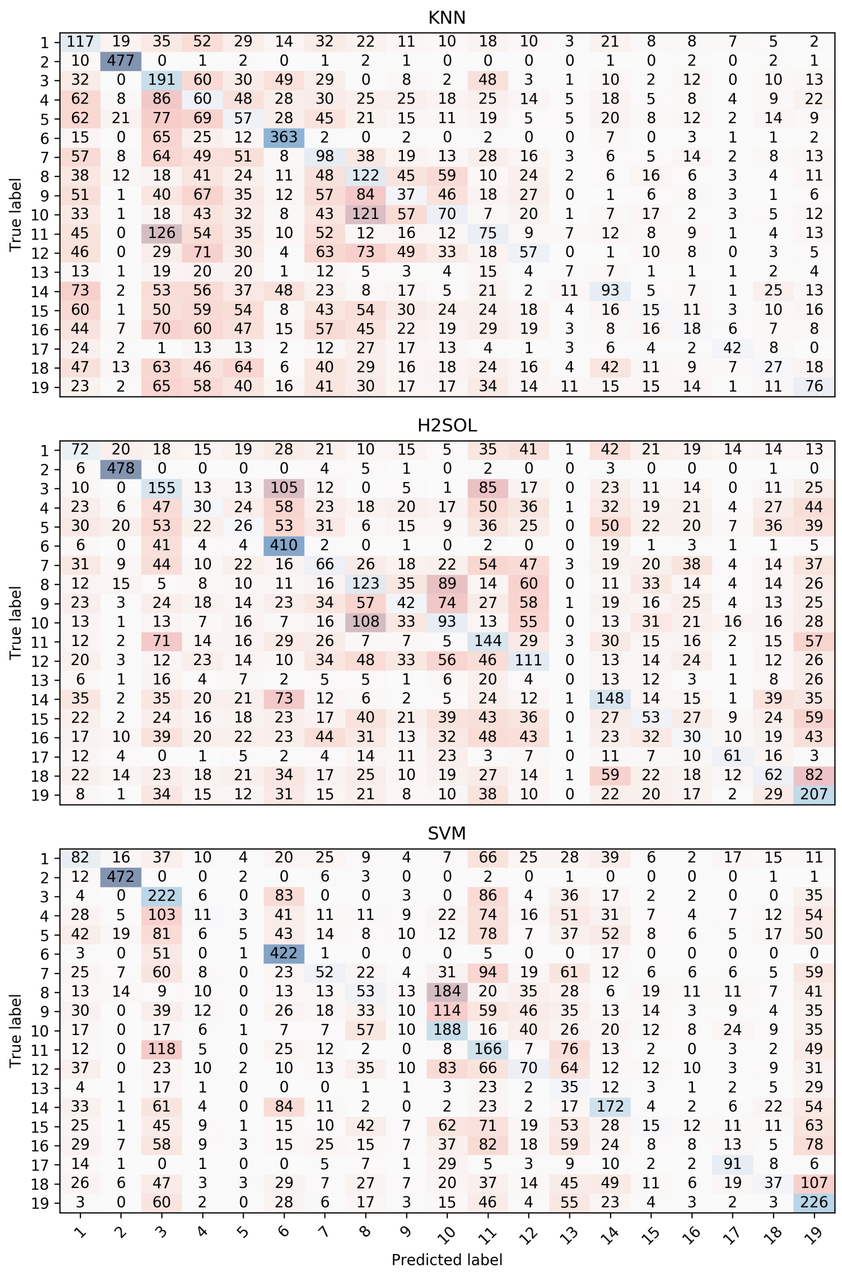


Figure 78: Confusion matrizes for the UVP5 dataset using the MPEG-7 features 1) art bubbles, 2) art turb, 3) det aggregate dark spherical, 4) det aggr dark spherical, 5) det aggregate dark, 6) det aggregate light, 7) det aggregate spherical, 8) det feces like fish, 9) det feces like munida, 10) det feces stick, 11) det fiber, 12) phyto tricho puff, 13) phyto tricho tuft, 14) pro rhizaria other, 15) pro rhizaria radiolaria 2, 16) zoo crust cop, 17) zoo crust cop like, 18) zoo gel carn medusa, 19) zoo moll

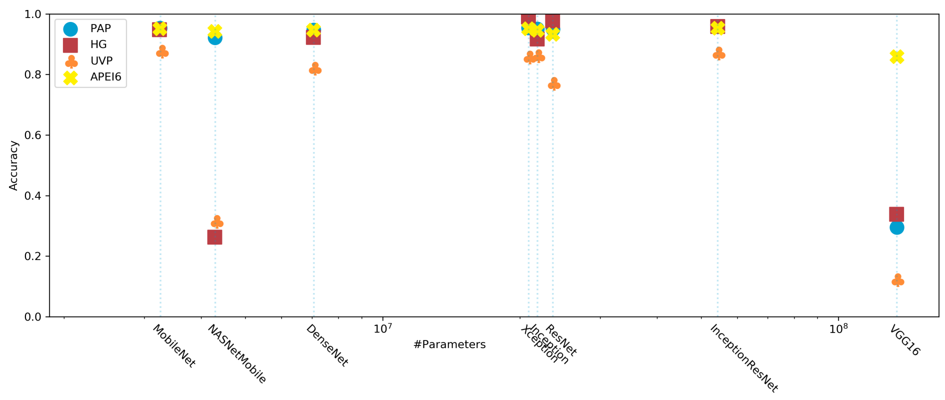


Figure 79: Validation accuracy and number of parameters of various deep learning classifiers (transfer learning) Validation accuracy and number of parameters of various deep learning classifiers with added transfer learning (pre-trained on ImageNet).

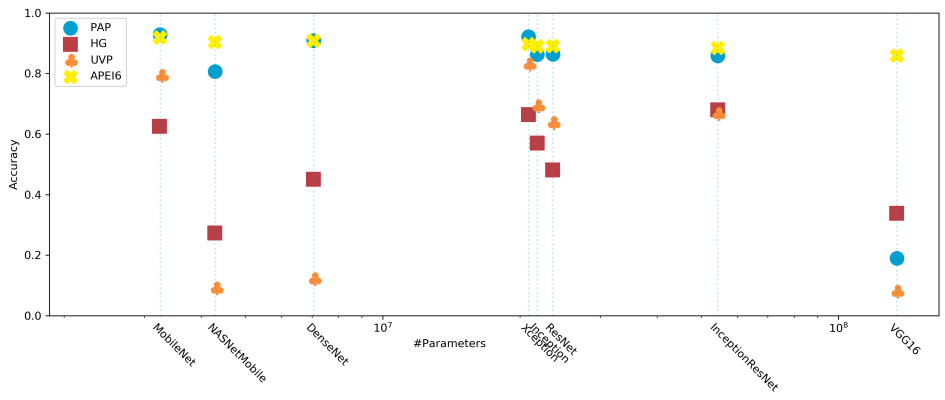


Figure 80: Validation accuracy and number of parameters of various deep learning classifiers (augmentation) Validation accuracy and number of parameters of various deep learning classifiers with added augmentation.

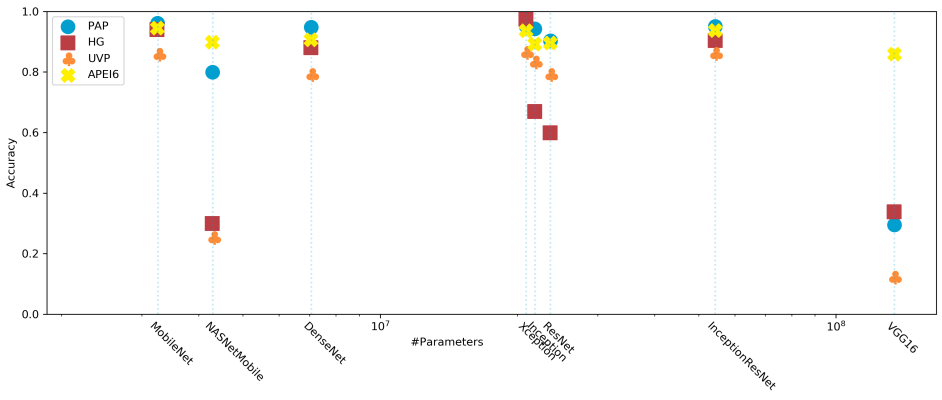


Figure 81: Validation accuracy and number of parameters of various deep learning classifiers (transfer learning and augmentation) Validation accuracy and number of parameters of various deep learning classifiers with added transfer learning (pre-trained on ImageNet) and augmentation.

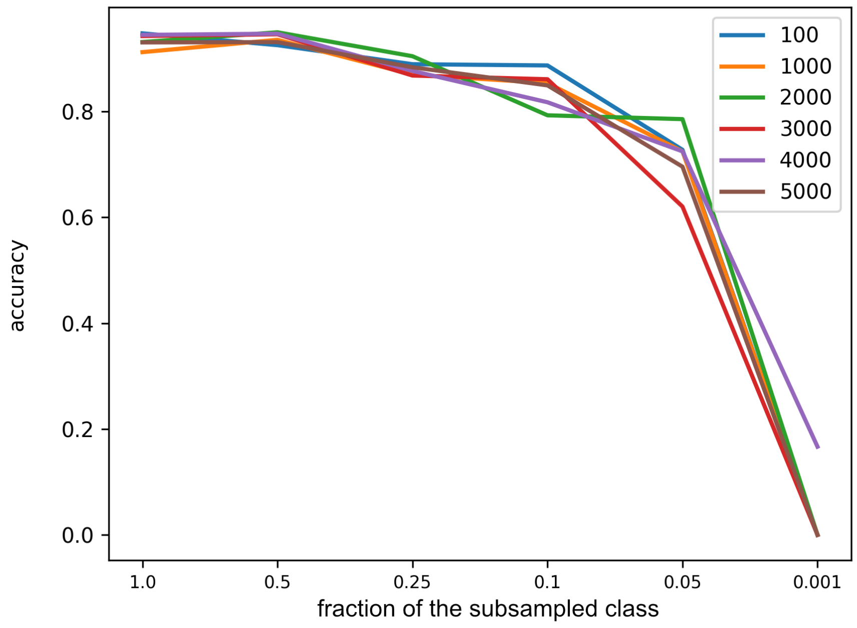
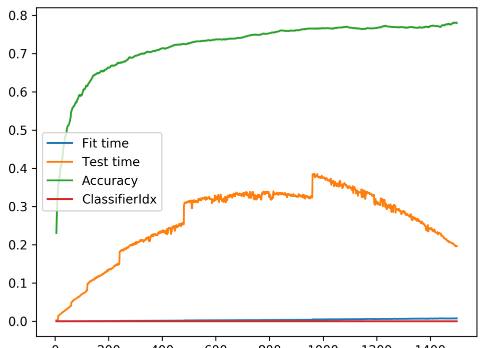
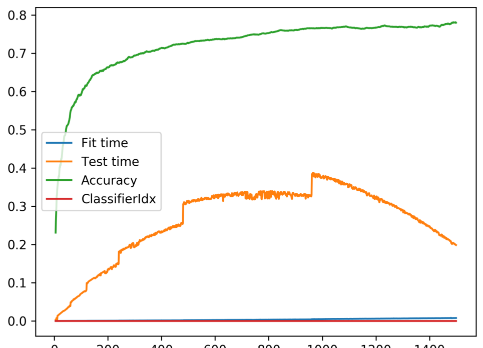


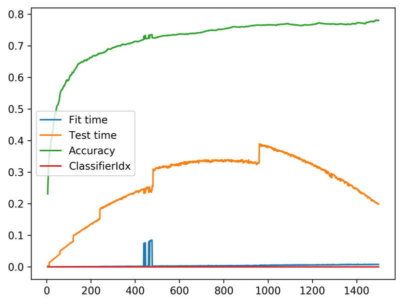
Figure 82: H^{2}SOM Memory Cell H^{2}SOM Memory Cell extension results. Each line represents one experimental run with the capacity depicted in the legend. The fraction of the digit zero of all data of that class in the dataset is shown on the x-axis.



(a)



(b)



(c)

Figure 83: Comparison of different methods for switching classifiers in the staged classifier approach *a)* Time limited naive (TLN) *b)* Time limited constrained for validation and training (TLVT) *c)* Training Error based (TE)

Bibliography

Martín Abadi et al. (2016). “Tensorflow: Large-scale machine learning on heterogeneous distributed systems”. In: *arXiv preprint arXiv:1603.04467*.

Vaneeda Allken et al. (2018). “Fish species identification using a convolutional neural network trained on synthetic data”. In: *ICES Journal of Marine Science* 76.1, pp. 342–349.

Franziska Althaus et al. (2015). “A standardised vocabulary for identifying benthic biota and substrata from underwater imagery: the CATAMI classification scheme”. In: *PloS one* 10.10, e0141039.

Codruta O Ancuti et al. (2018). “Color balance and fusion for underwater image enhancement”. In: *IEEE Transactions on Image Processing* 27.1, pp. 379–393.

Soma Banerjee et al. (2014). “Elimination of Marine Snow effect from underwater image-An adaptive probabilistic approach”. In: *Electrical, Electronics and Computer Science (SCEECS), 2014 IEEE Students' Conference on*. IEEE, pp. 1–4.

Roberto Souto Maior Barros and Silas Garrido T Carvalho Santos (2018). “A large-scale comparison of concept drift detectors”. In: *Information Sciences* 451, pp. 348–370.

Gustavo Batista, Ronaldo Prati, and Maria Monard (2004). “A study of the behavior of several methods for balancing machine learning training data”. In: *ACM SIGKDD explorations newsletter* 6.1, pp. 20–29.

O. Beijbom et al. (2015). “Towards Automated Annotation of Benthic Survey Images : Variability of Human Experts and Operational Modes of Automation”. In: *PLoS ONE* 10.7, e0130312.

Oscar Beijbom, Peter J Edmunds, David I Kline, et al. (2012). “Automated annotation of coral reef survey images”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, pp. 1170–1177.

Oscar Beijbom, Peter J Edmunds, Chris Roelfsema, et al. (2015). “Towards automated annotation of benthic survey images: Variability of human experts and operational modes of automation”. In: *PloS one* 10.7, e0130312.

Oscar Beijbom, Tali Treibitz, et al. (2016). “Improving automated annotation of benthic survey images using wide-band fluorescence”. In: *Scientific reports* 6, p. 23166.

Melanie Bergmann, Ingo Schewe, and Thomas Soltwedel (2017). *Seabed photographs taken along OFOS profile PS80/179-3 during POLARSTERN cruise ARK-XXVII/2*. data set. Alfred Wegener Institute, Helmholtz Center for Polar and Marine Research, Bremerhaven. DOI: `10.1594/PANGAEA.875082`. URL: `https://doi.pangaea.de/10.1594/PANGAEA.875082`.

Dana Berman, Tali Treibitz, and Shai Avidan (2017). "Diving into haze-lines: Color restoration of underwater images". In: *Proc. British Machine Vision Conference (BMVC)*. Vol. 1. 2.

Michael Bewley, Bertrand Douillard, et al. (2012). "Automated species detection: An experimental approach to kelp detection from sea-floor AUV images". In: *Proc Australas Conf Rob Autom*. Vol. 2012.

Michael Bewley, Navid Nourani-Vatani, et al. (2015). "Hierarchical classification in AUV imagery". In: *Field and service robotics*. Springer, pp. 3–16.

Ocean Studies Board, National Research Council, et al. (2003). *Exploration of the Seas: Voyage into the Unknown*. National Academies Press.

Anne Bowser et al. (2013). "Using gamification to inspire new citizen science volunteers". In: *Proceedings of the first international conference on gameful design, research, and applications*. ACM, pp. 18–25.

Leo Breiman et al. (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks.

Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski (2017). "A systematic study of the class imbalance problem in convolutional neural networks". In: *arXiv preprint arXiv:1710.05381*.

Alfredo Canziani, Adam Paszke, and Eugenio Culurciello (2016). "An analysis of deep neural network models for practical applications". In: *arXiv preprint arXiv:1605.07678*.

Stuart K Card, George G Robertson, and Jock D Mackinlay (1991). "The information visualizer, an information workspace." In: *CHI*, pp. 181–186.

Nitesh V Chawla et al. (2002). "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16, pp. 321–357.

China Released Clearest Full Moon Map (2018). `http://www.chinaembassy.org.ro/rom/kjwh/t904320.htm`. Accessed:19.11.2018.

François Chollet (2017). "Xception: Deep learning with depthwise separable convolutions". In: *arXiv preprint*, pp. 1610–02357.

Danelle E Cline et al. (2009). "An automated event detection and classification system for abyssal time-series images of Station M, NE Pacific". In: *OCEANS 2009*. IEEE, pp. 1–4.

MS COCO (2019). *COCO dataset explorer - Image #78565*. Website. `http://cocodataset.org/#explore?id=78565`;accessed 04/10/2019.

Luis Pedro Coelho et al. (2010). "Structured literature image finder: extracting information from text and images in biomedical literature". In: *Linking Literature, Information, and Knowledge for Biology*, pp. 23–32.

Corinna Cortes and Vladimir Vapnik (1995). "Support-vector networks". In: *Machine learning* 20.3, pp. 273–297.

Thomas Cover and Peter Hart (1967). "Nearest neighbor pattern classification". In: *IEEE transactions on information theory* 13.1, pp. 21–27.

Phil F Culverhouse et al. (2003). "Do experts make mistakes? A comparison of human and machine identification of dinoflagellates". In: *Marine Ecology Progress Series* 247.17-25, p. 5.

Bogusław Cyganek (2013). *Object detection and recognition in digital images: theory and practice*.

Bogusław Cyganek and Karol Gongola (2018). "Real-time marine snow noise removal from underwater video sequences". In: *Journal of Electronic Imaging* 27.4, p. 043002.

Jialun Dai et al. (2016). "ZooplanktoNet: Deep convolutional network for zooplankton classification". In: *OCEANS 2016-Shanghai*. IEEE, pp. 1–6.

Aymeric Denuelle and Matthew Dunbabin (2010). "Kelp detection in highly dynamic environments using texture recognition". In: *Australasian Conference on Robotics and Automation*.

Sebastian Deterding et al. (2011). "Gamification: Toward a definition". In: *CHI 2011 gamification workshop proceedings*. Vol. 12. Vancouver BC, Canada.

Jennifer Durden, Brian J Bett, et al. (2016). "Comparison of image annotation data generated by multiple investigators for benthic ecology". In: *Marine Ecology Progress Series* 552, pp. 61–70.

Jennifer Durden, Timm Schoening, et al. (2016). "Perspectives in visual imaging for marine biology and ecology: from acquisition to understanding". In: *Oceanography and Marine Biology: An Annual Review*.

Mohamed Elawady (2015). "Sparse coral classification using deep convolutional neural networks". In: *arXiv preprint arXiv:1511.09067*.

Charles Elkan (2001). "The foundations of cost-sensitive learning". In: *IJCAI*. Vol. 17. 1. Lawrence Erlbaum Associates Ltd, pp. 973–978.

Mark Everingham et al. (2015). "The pascal visual object classes challenge: A retrospective". In: *International journal of computer vision* 111.1, pp. 98–136.

Fahimeh Farhadifard, Martin Radolko, and Uwe Freiherr von Lukas (2017). "Single Image Marine Snow Removal based on a Supervised Median Filtering Scheme." In: *VISIGRAPP (4: VISAPP)*, pp. 280–287.

César Ferri, José Hernández-Orallo, and R Modroiu (2009). "An experimental comparison of performance measures for classification". In: *Pattern Recognition Letters* 30.1, pp. 27–38.

Stephanie Fitzherbert (2019). *Deepest Submarine Dive in History, Five Deeps Expedition Conquers Challenger Deep*. Tech. rep. The Five Deeps Expedition.

Tim Flannery (2007). "Where Wonders Await Us". In: *New York Review of Books. Website:* `http://www.nybooks.com/articles/archives/2007/dec/20/where-wonders-await-us/`(Accessed 10 October 2015).

Daniel J Fornari (2003). "the WHOI TowCam Group". In: *A new deep-sea towed digital camera and multi-rock coring system, Eos* 87.8, pp. 69–76.

Adrian Galdran et al. (2015). "Automatic red-channel underwater image restoration". In: *Journal of Visual Communication and Image Representation* 26, pp. 132–145.

James V Gardner et al. (2014). "So, how deep is the Mariana Trench?" In: *Marine Geodesy* 37.1, pp. 1–13.

Ross Girshick (2015). "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448.

Ross Girshick et al. (2016). "Region-based convolutional networks for accurate object detection and segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 38.1, pp. 142–158.

Olav Rune Godø, Ståle Johnsen, and Terje Torkelsen (2014). "The love ocean observatory is in operation". In: *Marine Technology Society Journal* 48.2, pp. 24–30.

Anabel Gómez-Ríos et al. (2019). "Towards Highly Accurate Coral Texture Images Classification Using Deep Convolutional Neural Networks and Data Augmentation". In: *Expert Systems with Applications* 118, pp. 315–328.

John S Gray and Michael Elliott (2009). *Ecology of marine sediments: from science to management*. Oxford University Press.

Jens Greinert (Dec. 2015). *RV SONNE Fahrtbericht / Cruise Report SO242-1 [SO242/1]*. Fahrtbericht doi:10.3289/GEOMAR_REP_NS_26_2015. Kiel, Germany: GEOMAR. DOI: `doi:10.3289/GEOMAR\_REP\_NS\_26\_2015`. URL: `http://oceanrep.geomar.de/31075/`.

Jens Greinert et al. (2017). *Seafloor images and raw context data along AUV tracks during SONNE cruises SO239 and SO242/1*. data set. Supplement to: Schoening, Timm; Köser, Kevin; Greinert, Jens (2018): An acquisition, curation and management workflow for sustainable, terabyte-scale marine image analysis. Scientific Data, 5, 180181. GEOMAR - Helmholtz Centre for Ocean Research Kiel. DOI: `10.1594/PANGAEA.882349`. URL: `https://doi.org/10.1594/PANGAEA.882349`.

Guo Haixiang et al. (2017). "Learning from class-imbalanced data: Review of methods and applications". In: *Expert Systems with Applications* 73, pp. 220–239.

Nicholas A Hamilton et al. (2007). "Fast automated cell phenotype image classification". In: *BMC bioinformatics* 8.1, p. 110.

David J Hand and Keming Yu (2001). "Idiot's Bayes—not so stupid after all?" In: *International statistical review* 69.3, pp. 385–398.

George Hanke et al. (2019). "Quantifying Macrolitter on the Seafloor: Current Practices and Outlook". in preparation.

Robert M. Haralick, Its'hak Dinstein, and K. Shanmugam (1973). "Textural features for image classification". In: *Ieee Transactions On Systems Man And Cybernetics* 3.6, pp. 610–621. URL: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4309314`.

Haibo He et al. (2008). "ADASYN - Adaptive synthetic sampling approach for imbalanced learning." In: *IJCNN*.

Kaiming He, Georgia Gkioxari, et al. (2017). "Mask r-cnn". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, pp. 2980–2988.

Kaiming He, Xiangyu Zhang, et al. (2015). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *ICCV*, 1026 –1034.

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

Joel W Hedgpeth (1957). “Classification of marine environments”. In: *Treatise on marine ecology and paleoecology* 50, pp. 17–28.

Matthias Hess et al. (Jan. 2011). “Metagenomic discovery of biomass-degrading genes and genomes from cow rumen.” English. In: *Science* 331.6016, 463 –467. DOI: `10.1126/science.1200387`.

Geoffrey E Hinton et al. (2012). “Improving neural networks by preventing co-adaptation of feature detectors”. In: *CoRR* cs.NE.

T. Horton et al. (Sept. 29, 2016). *World Register of Marine Species (WoRMS)*. http://www.marinespecies.org. Accessed: 2016-09-29. URL: `http://www.marinespecies.org`.

Brett Hosking et al. (2019). “Classification of megafauna in seabed photography using deep convolutional neural networks”. in preparation.

How much of the ocean have we explored? (2018). `https://oceanservice.noaa.gov/facts/exploration.html`. Accessed: 19.11.2018.

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

Jie Hu, Li Shen, and Gang Sun (2017). “Squeeze-and-Excitation Networks”. In: *CoRR* abs/1709.01507. arXiv: `1709.01507`. URL: `http://arxiv.org/abs/1709.01507`.

Ming-Kuei Hu (1962). “Visual pattern recognition by moment invariants”. In: *IRE transactions on information theory* 8.2, pp. 179–187.

Gao Huang et al. (2017). “Densely Connected Convolutional Networks.” In: *CVPR*. Vol. 1. 2, p. 3.

Guang-bin Huang, Qin-yu Zhu, and Chee-kheong Siew (2006). *Extreme learning machine: Theory and applications*.

Sergey Ioffe and Christian Szegedy (2015). “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167*.

Yangqing Jia et al. (2014). “Caffe: Convolutional architecture for fast feature embedding”. In: *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, pp. 675–678.

Matthew Johnson-Roberson, Oscar Pizarro, and Stefan Williams (2010). “ Saliency ranking for benthic survey using underwater images”. In: *Control Automation Robotics & Vision (ICARCV), 2010 11th Int. Conf. on*. IEEE, pp. 459–66.

Daniel OB Jones et al. (2017). “Biological responses to disturbance from simulated deep-sea polymetallic nodule mining”. In: *PLoS One* 12.2, e0171750.

Łukasz Kaiser et al. (2017). “Learning to remember rare events”. In: *arXiv preprint arXiv:1703.03129*.

Andrej Karpathy (2018). *t-SNE visualization of CNN codes*. Website. `https://cs.stanford.edu/people/karpathy/cnnembed/`;accessed 26/12/2018.

Shachar Kaufman et al. (2012). "Leakage in data mining: Formulation, detection, and avoidance". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.4, p. 15.

Salman H Khan et al. (2018). "Cost-sensitive learning of deep feature representations from imbalanced data". In: *IEEE transactions on neural networks and learning systems* 29.8, pp. 3573–3587.

Diederik P Kingma and Jimmy Ba (Dec. 2014). "Adam: A Method for Stochastic Optimization". In: *arXiv.org*. arXiv: 1412.6980v9 [cs.LG].

Teuvo Kohonen (1982). "Self-organized formation of topologically correct feature maps". English. In: *Biological Cybernetics* 43.1, pp. 59–69. DOI: 10.1007/BF00337288.

Ulrich HG Kressel (2002). "Pairwise classification and support vector machines". In: *Advances in Kernel Methods: Support Vector Learning*.

Alex Krizhevsky, Geoffrey Hinton, et al. (2009). *Learning multiple layers of features from tiny images*. Tech. rep. Citeseer.

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

Matjaz Kukar, Igor Kononenko, et al. (1998). "Cost-Sensitive Learning with Neural Networks." In: *ECAI*, pp. 445–449.

Christopher Kullenberg and Dick Kasperowski (2016). "What is citizen science?–A scientometric meta-analysis". In: *PloS one* 11.1, e0147152.

Stanford Vision Labs (2019). *IMAGENET Large Scale Visual Recognition Challenge (ILSVRC)*. Website. http://www.image-net.org/challenges/LSVRC/;accessed 05/10/2019.

Richard S Lampitt (2001). "Marine Snow". In: *Encyclopedia of Ocean Sciences (Second Edition)*. Ed. by John H. Steele. Second Edition. Oxford, pp. 686–694. ISBN: 978-0-12-374473-9. DOI: https://doi.org/10.1016/B978-012374473-9.00218-6. URL: https://www.sciencedirect.com/science/article/pii/B9780123744739002186.

Daniel Langenkämper, Alexander Goesmann, and Tim W Nattkemper (2014). "AKE-the Accelerated k-mer Exploration web-tool for rapid taxonomic classification and visualization". In: *BMC bioinformatics* 15.1, p. 384.

Daniel Langenkämper, Tobias Jakobi, et al. (2016). "Comparison of Acceleration Techniques for Selected Low-Level Bioinformatics Operations". In: *Frontiers in genetics* 7.

Daniel Langenkämper, Robin van Kevelaer, and Tim W Nattkemper (2019). "Strategies for Tackling the Class Imbalance Problem in Marine Image Classification". In: *Pattern Recognition and Information Forensics*. Ed. by Zhaoxiang Zhang et al. Cham: Springer International Publishing, pp. 26–36. ISBN: 978-3-030-05792-3.

Daniel Langenkämper, Robin van Kevelaer, Autun Purser, et al. (2019). "Gear-induced concept drift in marine images and its effect on deep learning classification". submitted.

Daniel Langenkämper and Tim W Nattkemper (2016). "COATL - a learning architecture for online real-time detection and classification assistance for environmental data". In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, pp. 597–602.

Daniel Langenkämper, Erik Simon-Lledó, et al. (June 2019). "On the impact of Citizen Science-derived data quality on deep learning based classification in marine images". In: *Plos One* 14.6, pp. 1–16. DOI: `10.1371/journal.pone.0218086`. URL: `https://doi.org/10.1371/journal.pone.0218086`.

Daniel Langenkämper, Martin Zurowietz, et al. (2017). "BIIGLE 2.0 - Browsing and Annotating Large Marine Image Collections". In: *Frontiers in Marine Science* 4, p. 83.

Steve Lawrence et al. (1998). "Neural network classification and prior class probabilities". In: *Neural networks: tricks of the trade*, pp. 299–313.

Layers of the Ocean (2018). `https://web.archive.org/web/20170207094728/http://www.srh.noaa.gov/jetstream/ocean/layers_ocean.html`. Accessed:19.11.18.

Yann LeCun, Léon Bottou, et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Yann LeCun and Corinna Cortes (2010). *MNIST handwritten digit database*. `http://yann.lecun.com/exdb/mnist/`.

Hansang Lee, Minseok Park, and Junmo Kim (2016). "Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning". In: *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, pp. 3713–3717.

Xiu Li et al. (2015). "Fast accurate fish detection and recognition of underwater images with fast R-CNN". In: *OCEANS'15 MTS/IEEE Washington*. IEEE, pp. 1–5.

Guinness World Records Limited (2016). *Guinness World Records 2017*. Guinness World Records. ISBN: 1910561339.

Tsung-Yi Lin et al. (2014). "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer, pp. 740–755.

Huimin Lu et al. (Sept. 2018). "FDCNet: filtering deep convolutional network for marine organism classification". In: *Multimedia Tools and Applications* 77.17, pp. 21847–21860. ISSN: 1573-7721. DOI: `10.1007/s11042-017-4585-1`. URL: `https://doi.org/10.1007/s11042-017-4585-1`.

Laurens van der Maaten and Geoffrey Hinton (2008). "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov, pp. 2579–2605.

Ammar Mahmood et al. (2016). "Coral classification with hybrid feature representations". In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, pp. 519–523.

Mahotas (2018). *Haralick Features documentation*. Website. `https://mahotas.readthedocs.io/en/latest/features.html#haralick-features`.

Inderjeet Mani and I Zhang (2003). "kNN approach to unbalanced data distributions: a case study involving information extraction". In: *Proceedings of workshop on learning from imbalanced datasets*. Vol. 126.

Bangalore S Manjunath et al. (2001). "Color and texture descriptors". In: *IEEE Transactions on circuits and systems for video technology* 11.6, pp. 703–715.

Aaron Marburg and Katie Bigham (2016). "Deep learning for benthic fauna identification". In: *OCEANS 2016 MTS/IEEE Monterey*. IEEE, pp. 1–5.

Ma Marcos, Laura David, et al. (2008). "Automated benthic counting of living and non-living components in Ngedarrak Reef, Palau via subsurface underwater video". In: *Environmental monitoring and assessment* 145.1-3, pp. 177–184.

Ma Marcos, Maricor Soriano, and Caesar Saloma (2005). "Classification of coral reef images from underwater video using neural networks". In: *Optics express* 13.22, pp. 8766–8771.

Stephen McPhail (2009). "Autosub6000: A deep diving long range AUV". In: *Journal of Bionic Engineering* 6.1, pp. 55–62.

Charles E Metz (1986). "ROC methodology in radiologic imaging." In: *Investigative radiology* 21.9, pp. 720–733.

Robert B Miller (1968). "Response time in man-computer conversational transactions." In: *AFIPS Fall Joint Computing Conference*, p. 267.

Torben Möller, Daniel Langenkämper, and Tim W Nattkemper (2019). "Wind turbine segmentation performing kNN-clustering on superpixel segmentations". In:

Andreas Momber et al. (2019). "Monitoring und Zustandsbewertung von Oberflächenschutzsystemen an Offshore-Windenergieanlagen". In: *Korrosionsschutz in der maritimen Technik*, pp. 59–80.

Md Moniruzzaman et al. (2017). "Deep Learning on Underwater Marine Object Detection: A Survey". In: *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer, pp. 150–160.

Jacquomo Monk et al. (2018). "An evaluation of the error and uncertainty in epibenthos cover estimates from AUV images collected with an efficient, spatially-balanced design". In: *PloS one* 13.9, e0203827.

Camilo Mora et al. (2011). "How many species are there on Earth and in the ocean?" In: *PLoS biology* 9.8, e1001127.

Kirsty J Morris et al. (2014). "A new method for ecological surveying of the abyss using autonomous underwater vehicle photography". In: *Limnology and Oceanography: Methods* 12.11, pp. 795–809.

Vinod Nair and Geoffrey E Hinton (2010). "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814.

Tim W Nattkemper, Helge Ritter, and Walter Schubert (2001). "A neural classifier enabling high-throughput topological analysis of lymphocytes in tissue sections". In: *Information Technology in Biomedicine, IEEE Transactions on* 5.2, pp. 138–149.

Diana Nelson (2019). *Save the Plankton, Breathe Freely*. Website. `https://www.nationalgeographic.org/activity/save-the-plankton-breathe-freely/`;accessed 05/10/2019.

Allen Newell (1990). *Unified theories of cognition*. Vol. 1987. The Williams James lectures ; 1987. Cambridge, Mass. [u.a.], XVII, 549 S. : graph. Darst. ISBN: 0-674-92099-6. URL: `http://digitool.hbz-nrw.de:1801/webclient/DeliveryManager?pid=3278318&custom_att_2=simple_viewer%20Interna:%20Inhaltsverzeichnis`.

Herbert Nitsch (2017). *Breathing is overrated*. Ed. by Robert Jelinek. Der Konterfei. ISBN: 978-3-903043-28-2.

Mark S Nixon and Alberto S Aguado (2012). *Feature extraction & image processing for computer vision*. Academic Press.

Timo Ojala, Matti Pietikainen, and Topi Maenpaa (2002). "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns". In: *IEEE Transactions on pattern analysis and machine intelligence* 24.7, pp. 971–987.

Jörg Ontrup, Nils Ehnert, et al. (2009). "BIIGLE-Web 2.0 enabled labelling and exploring of images from the Arctic deep-sea observatory HAUSGARTEN". In: *OCEANS 2009-EUROPE*. IEEE, pp. 1–7.

Jörg Ontrup and Helge Ritter (2005). "A hierarchically growing hyperbolic self-organizing map for rapid structuring of large data sets". In: *Proceedings of the 5th Workshop on Self-Organizing Maps, Paris (France)*. Citeseer.

Jonas Osterloff, Ingunn Nilssen, and Tim W Nattkemper (2016). "Computational coral feature monitoring for the fixed underwater observatory LoVe". In: *OCEANS 2016 MTS/IEEE Monterey*. IEEE, pp. 1–5.

Kuntal Kumar Pal and KS Sudeep (2016). "Preprocessing for image classification by convolutional neural networks". In: *Recent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE International Conference on*. IEEE, pp. 1778–1781.

Eunbyung Park (2017). *Overview of ILSVRC 2017*. Slides for Beyond ImagetNet Large Scale Visual Recognition Challenge, July 26th in conjunction with CVPR 2017. `http://image-net.org/challenges/talks_2017/ILSVRC2017_overview.pdf`;accessed 04/10/2019.

Luis Perez and Jason Wang (2017). "The Effectiveness of Data Augmentation in Image Classification using Deep Learning". In: *CoRR* abs/1712.04621. arXiv: `1712.04621`. URL: `http://arxiv.org/abs/1712.04621`.

Maria Petrou and Costas Petrou (2010). *Image processing: the fundamentals*. John Wiley & Sons.

Marc Picheral et al. (2010). "The Underwater Vision Profiler 5: An advanced instrument for high spatial resolution studies of particle size spectra and zooplankton". In: *Limnology and Oceanography: Methods* 8.9, pp. 462–473.

Michael Pidwirny (2006). *Introduction to the Oceans*. `http://www.physicalgeography.net/fundamentals/8o.html`. Accessed: 13.10.2019.

Jean Ponce et al. (2011). "Computer vision: a modern approach". In: *Computer* 16.11.

Shan E Ahmed Raza et al. (2016). "Robust normalization protocols for multiplexed fluorescence bioimage analysis". In: *BioData Mining* 9, pp. 1–13.

Joseph Redmon, Santosh Divvala, et al. (2016). "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.

Joseph Redmon and Ali Farhadi (2017). "YOLO9000: better, faster, stronger". In: *arXiv preprint*.

Shaoqing Ren et al. (2015). "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*, pp. 91–99.

Michael D Richard and Richard P Lippmann (1991). "Neural network classifiers estimate Bayesian a posteriori probabilities". In: *Neural computation* 3.4, pp. 461–483.

William Hadley Richardson (1972). "Bayesian-based iterative method of image restoration". In: *JOSA* 62.1, pp. 55–59.

Tadas Rimavicius and Adas Gelzinis (2017). "A Comparison of the Deep Learning Methods for Solving Seafloor Image Classification Task". In: *International Conference on Information and Software Technologies*. Springer, 442–453.

H Ritter (1999). "Self-organizing maps on non-euclidean spaces". In: *Kohonen maps* 73. URL: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.3936&rep=rep1&type=pdf`.

R. Rivest (1992). *The MD5 Message-Digest Algorithm*. Tech. rep. United States.

Frank Rosenblatt (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386.

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

Olga Russakovsky et al. (2015). "Imagenet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3, pp. 211–252.

Omer Sagi and Lior Rokach (2018). "Ensemble learning: A survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.4, e1249. DOI: `10.1002/widm.1249`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1249`.

Jürgen Schmidhuber (2015). "Deep learning in neural networks - An overview." In: *Neural Networks* 61, pp. 85–117.

Timm Schoening, Melanie Bergmann, et al. (2012). "Semi-automated image analysis for the assessment of megafaunal densities at the Arctic deep-sea observatory HAUSGARTEN". In: *PloS one* 7.6, e38179.

Timm Schoening, Thomas Kuhn, et al. (2015). “DELPHI—fast and adaptive computational laser point detection and visual footprint quantification for arbitrary underwater image collections”. In: *Frontiers in Marine Science* 2, p. 20.

Timm Schoening, Daniel Langenkämper, et al. (2015). “Rapid image processing and classification in underwater exploration using advanced high performance computing”. In: *OCEANS'15 MTS/IEEE Washington*. IEEE, pp. 1–5.

Timm Schoening, Jonas Osterloff, and Tim W Nattkemper (2016). “RecoMIA—Recommendations for Marine Image Annotation: Lessons Learned and Future Directions”. In: *Frontiers in Marine Science* 3, p. 59.

Timm Schoening, Autun Purser, et al. (2019). “Megafauna community assessment of polymetallic nodule fields with cameras: Platform and methodology comparison”. submitted.

Gregory S Schorr et al. (Mar. 2014). “First Long-Term Behavioral Records from Cuvier's Beaked Whales (Ziphius cavirostris) Reveal Record-Breaking Dives”. In: *PLOS ONE* 9.3, pp. 1–11. DOI: 10.1371/journal.pone.0092633. URL: https://doi.org/10.1371/journal.pone.0092633.

Burr Settles (2012). “Active learning”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.1, pp. 1–114.

Claude Elwood Shannon (1948). “A mathematical theory of communication”. In: *Bell system technical journal* 27.3, pp. 379–423.

Stuart C Shapiro (1992). *Encyclopedia of artificial intelligence second edition*. John.

ASM Shihavuddin et al. (2013). “Image-based coral reef classification and thematic mapping”. In: *Remote Sensing* 5.4, pp. 1809–1841.

Kang G Shin and Parameswaran Ramanathan (1994). “Real-time computing: A new discipline of computer science and engineering”. In: *Proceedings of the IEEE* 82.1, pp. 6–24.

Connor Shorten and Taghi M. Khoshgoftaar (July 2019). “A survey on Image Data Augmentation for Deep Learning”. In: *Journal of Big Data* 6.1, p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0.

Shoaib Ahmed Siddiqui et al. (2017). “Automatic fish species classification in underwater videos: exploiting pre-trained deep neural network models to compensate for limited labelled data”. In: *ICES Journal of Marine Science* 75.1, pp. 374–389.

Thomas Sikora (2001). “The MPEG-7 visual standard for content description-an overview”. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 11.6, pp. 696–702.

Mary Silver (2015). “Marine snow: a brief historical sketch”. In: *Limnology and Oceanography Bulletin* 24.1, pp. 5–10.

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

Marina Sokolova and Guy Lapalme (July 2009). "A Systematic Analysis of Performance Measures for Classification Tasks". In: *Inf. Process. Manage.* 45.4, pp. 427–437. ISSN: 0306-4573.

Thomas Soltwedel et al. (2016). "Natural variability or anthropogenically-induced variation? Insights from 15 years of multidisciplinary observations at the arctic marine LTER site HAUSGARTEN". In: *Ecological Indicators* 65, pp. 89–102.

Nitish Srivastava et al. (2014). "Dropout: A simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber (2015). "Training very deep networks". In: *Advances in neural information processing systems*, pp. 2377–2385.

John H Steele (2010). *Marine Policy & Economics*. Elsevier.

M Dale Stokes and Grant B Deane (2009). "Automated processing of coral reef benthic images". In: *Limnology and Oceanography: Methods* 7.2, pp. 157–168.

Devin P Sullivan et al. (2018). "Deep learning is combined with massive-scale citizen science to improve large-scale image classification". In: *Nature biotechnology*.

Christian Szegedy, Sergey Ioffe, et al. (2017). "Inception-v4, inception-resnet and the impact of residual connections on learning." In: *AAAI*. Vol. 4, p. 12.

Christian Szegedy, Wei Liu, et al. (2015). "Going deeper with convolutions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR*, pp. 1–9.

Christian Szegedy, Vincent Vanhoucke, et al. (2016). "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826.

Chuanqi Tan et al. (2018). "A survey on deep transfer learning". In: *International Conference on Artificial Neural Networks*. Springer, pp. 270–279.

Michael Reed Teague (1980). "Image analysis via the general theory of moments". In: *JOSA* 70.8, pp. 920–930.

Sik-Ho Tsang (2019). *Review: Trimps-Soushen — Winner in ILSVRC 2016 (Image Classification)*. Website. `https://towardsdatascience.com/review-trimps-soushen-winner-in-ilsvrc-2016-image-classification-dfbc423111dd`; accessed 05/10/2019.

Gene W Tyson et al. (2004). "Community structure and metabolism through reconstruction of microbial genomes from the environment". English. In: *Nature* 428.6978, pp. 37–43. DOI: `doi:10.1038/nature02340`.

Jasper RR Uijlings et al. (2013). "Selective search for object recognition". In: *International journal of computer vision* 104.2, pp. 154–171.

Sébastien Villon et al. (2016). "Coral reef fish detection and recognition in underwater videos by supervised machine learning: Comparison between Deep

Learning and HOG+ SVM methods". In: *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer, pp. 160–171.

Colin Ware (2012). *Information visualization: perception for design*. Elsevier.

LM Wedding et al. (2013). "From principles to practice: a spatial approach to systematic conservation planning in the deep sea". In: *Proceedings of the Royal Society of London B: Biological Sciences* 280.1773, p. 20131684.

Max Wertheimer (1923). "Untersuchungen zur Lehre von der Gestalt. II". In: *Psychological Research* 4.1, pp. 301–350.

Dennis L Wilson (1972). "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data." In: *IEEE Trans. Systems, Man, and Cybernetics*.

Chee Sun Won, Dong Kwon Park, and Soo-Jun Park (2002). "Efficient Use of MPEG-7 Edge Histogram Descriptor". In: *ETRI journal* 24.1, pp. 23–30.

Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng (2004). "Probability Estimates for Multi-class Classification by Pairwise Coupling." In: *Journal of Machine Learning Research*.

Russell B Wynn et al. (2014). "Autonomous Underwater Vehicles (AUVs): Their past, present and future contributions to the advancement of marine geoscience". In: *Marine Geology* 352, pp. 451–468.

Jason Yosinski et al. (2014). "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems*, pp. 3320–3328.

Matthew D Zeiler and Rob Fergus (2014). "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer, pp. 818–833.

Zhong-Qiu Zhao et al. (2019). "Object detection with deep learning: A review". In: *IEEE transactions on neural networks and learning systems*.

Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama (2016). "An overview of concept drift applications". In: *Big data analysis: new algorithms for a new society*. Springer, pp. 91–114.

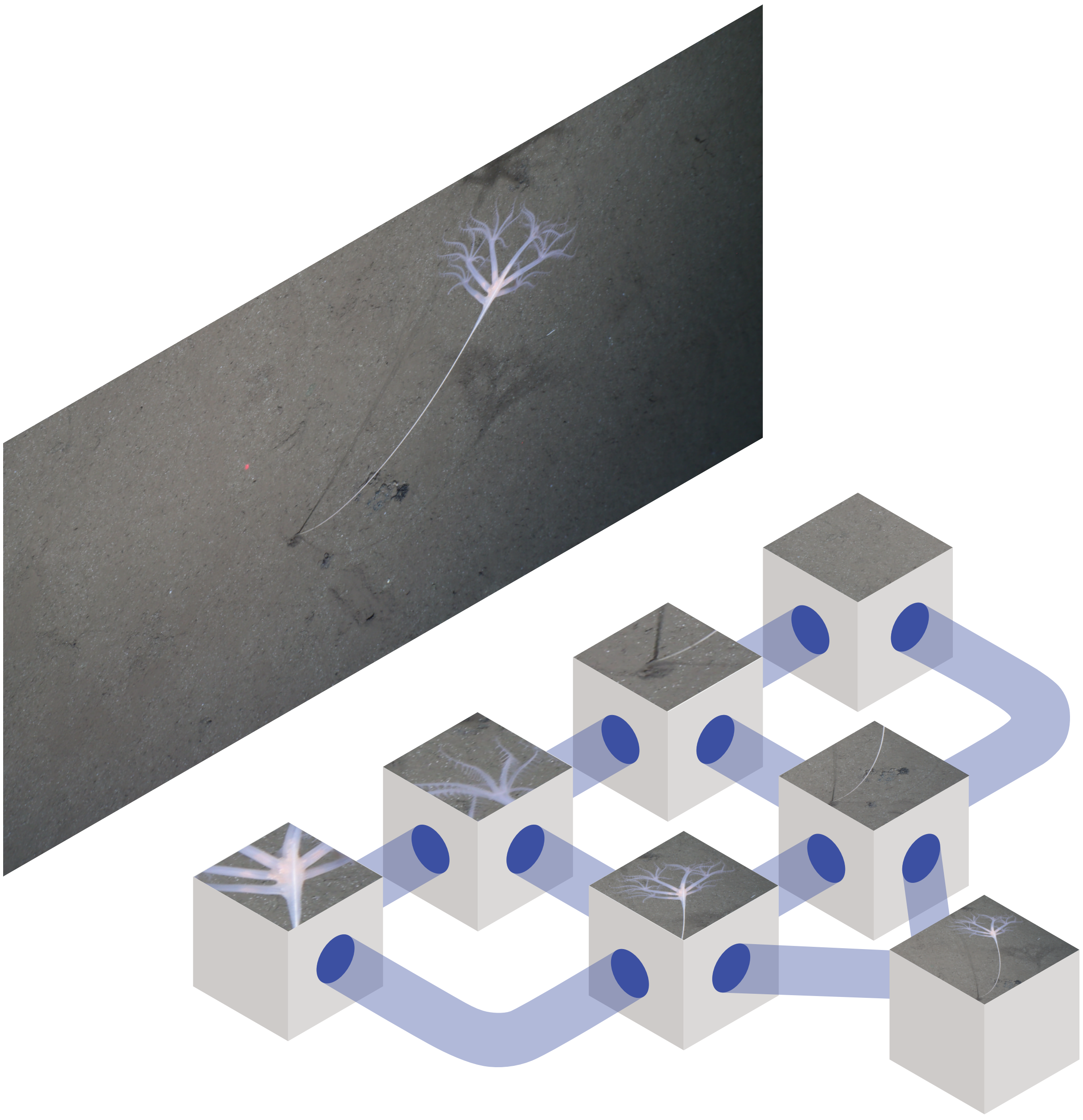
Barret Zoph et al. (2017). "Learning transferable architectures for scalable image recognition". In: *arXiv preprint arXiv:1707.07012* 2.6.

Karel Zuiderveld (1994). "Contrast limited adaptive histogram equalization". In: *Graphics gems*, pp. 474–485.

Martin Zurowietz, Daniel Langenkämper, Brett Hosking, et al. (2018). "MAIA - A machine learning assisted image annotation method for environmental monitoring and exploration". In: *PloS one* 13.11, e0207498.

Martin Zurowietz, Daniel Langenkämper, and Tim W Nattkemper (2019). "BIIGLE2Go - A scalable image annotation system for easy deployment on cruises". In: *Proceedings of IEEE OCEANS 2019*.

Martin Zurowietz, Daniel Langenkämper, Erik Simon-Lledó, et al. (2018). "A Hybrid Machine Learning Method for Object Detection in Environmental Monitoring". In: *Proceedings of the International Conference on Pattern Recognition (ICPR)*. IEEE.



TTFN
Ta Ta For Now