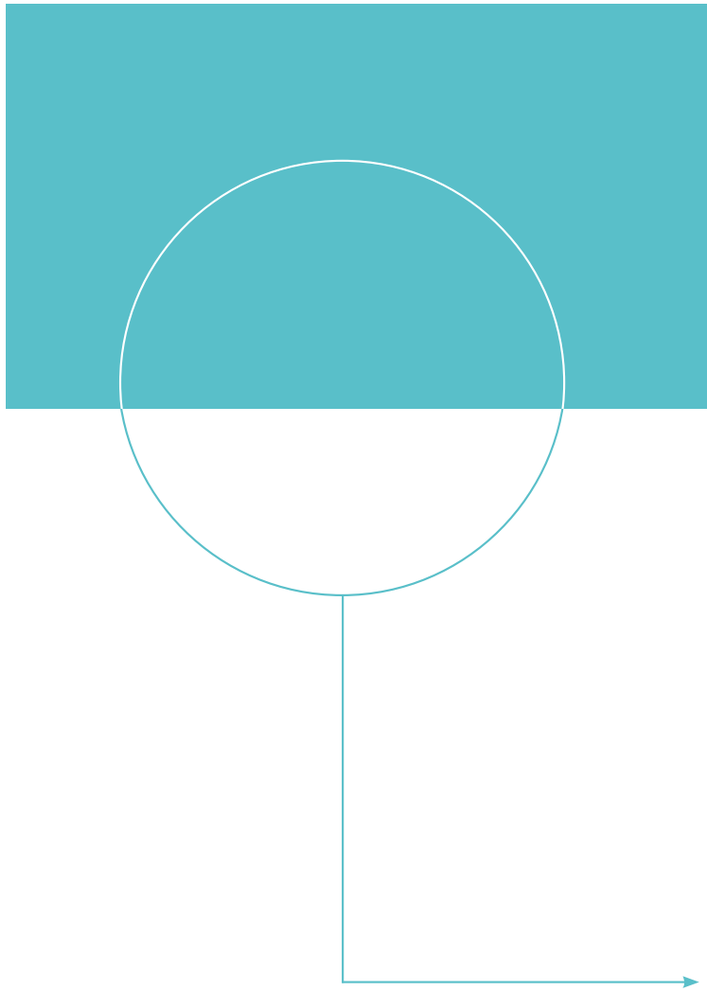


Master's thesis

2021

Eivind Salvesen

**NTNU**
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Engineering Cybernetics

Eivind Salvesen

# Unsupervised methods for in-situ classification of plankton taxa

January 2021

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

# Unsupervised methods for in-situ classification of plankton taxa

Eivind Salvesen

Master's thesis in Cybernetics and Robotics
Submission date: January 2021
Supervisor: Annette Stahl
Co-supervisor: Aya Saad

Norwegian University of Science and Technology
Department of Engineering Cybernetics

# Abstract

Planktonic species are of great importance in the marine ecosystem, standing for about half of the earth's primary production and being a fundamental part of the marine food chain. Yet, our understanding of these vital creatures and the consequences of small changes in their habitats, creating profound shifts in planktonic dispersion and abundance, is still limited. Recent development and innovation within autonomous underwater vehicles (AUV) makes it possible to utilize AUVs for in-situ identification and classification of plankton taxa resulting in faster and more reliable calculations of its distributions than any existing sampling and classification platform. For accurate image classification, the system is heavily dependent on accurate and attested machine learning techniques. Currently, deep convolutional neural networks have proven especially effective for this task. Yet, the success is fixed to supervised learning, which requires an extensive amount of labeled training data. Such methods thus require a comprehensive and time-consuming labeling effort.

The scope of this work is to make a deep learning framework for plankton classification training on images that contain no ground truth labels. This work extends the work of the specialization project by proposing new feature extraction methods using state of the art unsupervised training schemes. These models can then be used to extract features that can improve a separate clustering algorithm. For comparison to the specialization project, the models are tested over existing planktonic data sets. The most successful methods are then adapted onto the image data acquired from the AUV missions in the Trondheim fjord.

Three methods for improved feature learning, *DeepCluster* [15], a *generative adversarial network* (GAN) model and a *rotation-invariant autoencoder* were selected for this thesis. The former two were chosen as they represent some promising new directions within the unsupervised deep learning and have shown recent success at learning essential features on large image data sets. The rotation invariant autoencoder is an extension of a conventional autoencoder, which is more robust to similar class objects with different rotations. The proposed methods were then used as feature extractors to significantly improve different clustering algorithms in regards to classification performance over planktonic data. The chosen methods demonstrate some of the possibilities within the unsupervised domain, but the gap towards supervised learning is still significant.

# Sammendrag

Plankton spiller en enormt viktig rolle i marine økosystemer. De står for om lag halvparten av jordens primærproduksjon i tillegg til å være en fundamental brikke i den marine næringskjeden. Vår forståelse av disse viktige skapningene og konsekvensene av små endringer i deres habitat som forårsaker store endringer i plankton spredning og tallrikhet er likevel liten. Nyskapning innen autonome undervannsfarkoster (AUV) for in-situ identifikasjon og klassifikasjon av plankton arter kan gi raskere og mer pålitelige beregninger av deres utbredelse og fordeling enn eksisterende prøvetaknings og klassifikasjonsplattformer.

For nøyaktig klassifisering av tusenvis av bilder, kreves nøyaktige og utprøvde maskinlæringsmetoder. For øyeblikket har convolutional neural networks (CNN) demonstrert at de er spesielt effektive innenfor slike oppgaver. Suksessen er likevel kun knyttet til veiledet (supervised) læring som krever store mengder inndata-utdata par hvor merkingen av inndataen er veldig arbeids og tidskrevende.

Dette arbeidets omfang er å lage et dypt lærings rammeverk for klassifikasjon av plankton uten å benytte forhåndsmerket data. Arbeidet videreutvikler resultatene gjort fra spesialiseringsprosjektet ved å introdusere nye metoder for å lære viktige trekk ved plankton arter ved hjelp av ikke veiledet (unsupervised) læring. Disse metodene kan deretter brukes til å hente ut de viktigste trekkene ved hvert bilde for slik å forbedre treffsikkerheten til ulike grupperingsalgoritmer. For å kunne sammenligne modellene brukt i spesialiseringsprosjektet er metodene testet over eksisterende plankton datasett. De beste metodene er deretter tilpasset dataene hentet fra AUV ekspidisjonene i Trondheimsfjorden.

Tre metoder for forbedret læring over plankton data, *DeepCluster* [15], et *generative adversarial network* (GAN) nettverk og en *rotation-invariant autoencoder*, er valgt ut til denne oppgaven. De to førstnevnte ble valgt ut ettersom de representerer lovende nye retninger innen ikke veiledet læring og har vist seg kapable til å lære viktige trekk ved bilder fra en rekke store datasett. The rotation invariant autoencoder er en utvidelse av de tidligere utprøvde autoencoderne med bedre egenskaper i forhold til roterte bilder fra samme objektklassse. De foreslåtte metodene er brukt til å forbedre ulike grupperingsalgoritmer. Metodene demonstrerer noen av mulighetene innen ikke veiledet læring, men avstanden til veiledete algoritmer er fortsatt betydelig.

# Preface

This master thesis, TTK4900, is carried out in the department of Engineering cybernetics at the Norwegian University of Science and Technology. The report is a part of the multidisciplinary AILARON[1] project, and the work is conducted under the project's computer vision department.

This thesis is a continuation of the undersigned's specialization project [100], carried out during the spring of 2020. Because the basic parts are fundamental for the understanding and work in both the specialization project and the master thesis, altered and greatly improved versions of the theory chapters from the specialization-project are therefore present in this report. Chapter 2 contains background material on planktonic species. The chapter gives a brief introduction to planktonic species and presents previous work on planktonic sampling and classification. Finally, some publicly available data sets are presented. Chapter 3 contains the basic theory of artificial intelligence and deep learning being the fundamental building blocks of this thesis.

I want to thank my head supervisor, Annette Stahl, for her valuable ideas and guidance throughout the thesis. I am also truly grateful for the help and guidance from my second supervisor, Aya Saad. Her high spirits and belief in my work has been extremely motivating and helped me perform as good as possible.

*Eivind Salvesen*

*Trondheim, January 10, 2021*

[1] AILARON is a multidisciplinary project seeking to extend the knowledge of planktonic species and their real-time distributions. The project goal is an in-situ, fully automatic sampling and classification pipeline installed on autonomous underwater vehicles. This research is funded by the RCN FRINATEK IKTPLUSS program (project number 262741) and supported by NTNU AMOS.

# Table of Contents

# List of Tables

# List of Figures

# Abbreviations

| Abbreviation | Description |
|---|---|
| AE | Autoencoder |
| AI | Artificial intelligence |
| ANN | Artificial Neural Network |
| API | Application programming interface |
| AUV | Autonomous underwater vehicle |
| BCE | Binary cross entropy |
| BIRCH | Balanced Iterative Reducing and Clustering using Hierarchies |
| CAM | Class Activation Mapping |
| CNN | Convolutional Neural Network |
| DAE | Denoising autoencoder |
| DC | Deep Cluster |
| DCEC | Deep convolutional embedded clustering |
| DEC | Deep embedded clustering |
| FC | Fully connected (neural network layer) |
| GAP | Global average pooling |
| GAP-CAM | Global Average Pooling - Class Activation Mapping |
| GPU | Graphics Processing Unit |
| HAB | Harmful Algal Bloom |
| IFCB | Imaging FlowCytobot |
| ISIIS | In Situ Ichthyoplankton Imaging System |
| KL | Kullback-Leibler divergence |
| ML | Machine learning |
| MSE | Mean squared error |
| NMI | Normalized Mutual Info |
| ReLU | Rectified linear unit |
| SC | Spectral clustering |
| SIFT | Scale invariant feature transform |
| SURF | Graphics Processing Unit |
| SVM | Support Vector Machine |
| T-SNE | T-distributed stochastic neighbor embedding |
| VAE | Variational autoencoder |
| WHOI | Woods Hole Oceanographic Institution (data set) |

Chapter 1

# Introduction

Throughout history, the oceans have played an essential role in the evolution of humankind. Covering almost three-quarters of the Earth's surface, the global ocean is a vital part of the global climate system. The oceans supply important food resources and other valuables and provide a highway for trade and commerce. However, even today, our understanding of the ocean ecosystem's complexity and its impact on the global system are limited. Rapid changes in the marine environment due to increased temperatures, pollution, and overfishing can create large and unforeseen consequences, affecting the entire world.

The base of the ocean ecosystem is formed by small plants and animals called plankton. Albeit small, photosynthetic plankton species are considered to produce up to fifty percent of the Earth's primary production. Furthermore, the larger animal species, which graze on the former, are the primary food source in the aquatic food chain. As such, changes in plankton ecology may affect the surrounding marine areas and the entire global ecosystem. Consequently, increased knowledge of planktonic species and their population dynamics seems a critical starting point to understand the long and short-term consequences of the ocean climate changes.

## 1.1 Motivation

In recent years several underwater camera systems been proposed for image-based sampling of plankton and other microscopic particles [20, 26, 86]. These methods have made sampling easier, less time consuming and much more affordable than traditional sampling methods. Furthermore, the possibility of continuous real time surveillance makes it possible to develop rapid detection systems or gather long time-series of plankton data which is important for plankton research. However, classifying and assessing the increasing quantities of planktonic images is rapidly becoming impractical since traditional classification

requires extensive effort from domain level experts.

In other domains, similar problems have been solved by training a machine learning (ML) algorithm to perform the image classification. Therefore, it is of little surprise that the increase in readily available planktonic image datasets have resulted in several attempts of adapting ML to the plankton domain. In the beginning, the methods relied on careful selection of hand-designed features, which were subsequently used to train the ML classifier [41]. Even though several methods show promising results [105, 123], the challenge related to labor-intensive work partially remain. The careful selection and manual design of feature descriptors that are required to get a well functioning ML model, can unfortunately be both difficult and labor-intensive [43].

In the last decade, attention has turned to deep learning models that are capable of learning good feature descriptors on its own. Especially, Convolutional Neural Networks (CNN) have proven capable of detecting and classifying a wide set of plankton classes [71, 78, 87, 98]. The recent successes are, however, tied to the supervised domain where labeled data is an necessity. Labeling effort is thus easily getting unmanageable since deep learning models require a training sets containing thousands of images.

Observing the shortcomings of the above methods, a model of preference should be able to deduce the important aspects of the data without requiring any labor intensive supervision. Inspired by the studies of the brain, unsupervised deep learning are by many considered to be the future of deep learning [66]. Deep unsupervised models avoid not only the tiresome labeling effort, but can utilize the enormous amount of existing unlabeled data. Albeit far behind supervised models in regards to classification performance, unsupervised models are capable of learning impressive representations of complex data. The goal of this thesis is to adapt such methods into the planktonic domain and showcase the applicability of the up and coming unsupervised domain.

This thesis is a part of an overall program called the AILARON project. The project aims at streamlining plankton detection and classification by integrating camera systems and advanced machine learning algorithms onto an autonomous underwater vehicle (AUV). The system should provide in-situ observations of plankton and provide a real-time overview of the plankton distributions. One of the difficult tasks is to provide a detection and classification framework that can differentiate varying species accurately within a short amount of time. Furthermore, it is likely that new classes that the network has not seen before will appear. Thus, as time passes, the framework must be able to adapt to the new conditions. For now, much effort has been put into finding an applicable (CNN) network suitable for the task [99]. As of yet, the given training data contains seven classes and a low number of images. As the project proceeds and new species are collected, these samples must be labeled by experts in a continuous manner until all possible species are discovered. Solely relying on supervised learning seems, therefore, inadequate. Therefore, a major motivation is to improve the prediction abilities of the framework previously mentioned by adopting new techniques that are less dependent on human supervision. Furthermore, unsupervised models base their knowledge upon general traits of the underlying data. Despite strong variations between planktonic species, the existence of common characteristics between

classes might make it possible to predict unseen species.

## 1.2 Research questions and challenges

This section's main motivation is to showcase some of the challenges and important research questions that need to be solved for successfully adapting unsupervised machine learning models to the plankton domain. Hopefully, the selected topics will provide some context and insight into the work conducted in this thesis and encourage further reading.

#### Unsupervised feature learning for classification

Deep unsupervised models are typically trained on tasks that differ from the task the model is actually intended for. For instance, in the specialization-project [100], an autoencoder (AE), which is trained to reconstruct its inputs, was used as a feature extractor for subsequent classification. Therefore, the question is: Are the learned representations obtained by a deep neural network on one task transferable to a different task? In 2016, Xie et al. [117] showed a training method that adapted the learned autoencoder feature space for improved clustering. In the same manner, Hartono et al. observed that the "labels of the data influence the internal organization" [47, Hartono, p. 3] when they mixed supervised and unsupervised training. These observations clearly suggest that the novel autoencoder training scheme results in the model obtaining feature representations that are partially distinct from the optimal features for classification. Therefore, it might be necessary to find other training schemes that better facilitate the goal of making an unsupervised classification framework.

#### Rotation invariance

From the novel experiments conducted in the specialization-project [100] a consistent problem for all deep feature extraction models turned out to be the lack of rotation invariance. Visualization showed that identical objects with different orientations were split into separate groups. This is a significant problem for unsupervised clustering since separate groups indicates the presence of several distinct groups of objects. Therefore, an important requirement of the final deep feature extraction model is to obtain a feature representation that split objects belonging to different classes but avoids the separation of objects similar in all but orientation.

#### Plankton datasets, image diversity and quality

According to Goodfellow et al., the best way to improve a machine learning model's performance is to train it on more data [43]. However, obtaining large plankton datasets of high quality requires a lot of effort and time from a relatively small community. The Kaggle dataset [20] which was utilized in the specialization-project only contains thirty thousand images, which is considered a small data set for a deep learning algorithm [78]. Furthermore, plankton datasets suffers from large class imbalance and, according to [24] possibly high number of miss-classifications. How these traits affect the representations that can be learned by state of the art models adapted to the plankton domain is therefore

hard to predict. Luo et al. [78] for instance, experienced a large drop in classification accuracy when testing their supervised network on unseen data. However, when they removed all non-rare species from the test set, the model reached over 90% accuracy.

Plankton is not one class of strikingly similar images. Instead, the species are defined by their inability to move independently from ocean currents. Plankton is, therefore, a group of vast diversity in size and shape. Gelatinous species have transparent bodies, which makes them nearly invisible. Other species such as *diatoms* form colonies. These colonies are easier to detect at the cost of varying feature characteristics. Consequently, differing image quality and object features make plankton classification tough even for human experts. High performing machine learning models usually achieve results close to those achieved by humans making the domain equally difficult for computers.

#### Transferability to the AILARON project

The experiments conducted in the specialization-project [100] were performed on the Kaggle [21] dataset to find an adequate model to be used in the AILARON project. Thus, the ground assumption is that training over different data provides insight into the model capability when transferred to the AILARON task. The Kaggle dataset was chosen due to apparent similarities in the extraction method, which suggests potential similarities in the end image format. Unfortunately, the ongoing development of the AILARON image database shows that these expectations are a little optimistic. Which conclusions one can draw from the Kaggle experiments are therefore difficult to answer.

#### Speed, computational power and memory consumption

Connecting deep learning with robotics set strict limits on the use of deep learning architectures since many are very computationally expensive. For the AILARON project, the classification framework is expected to produce real-time predictions using as little memory and computational resources as possible. Concerning the prediction speed, the camera module captures between five to seven frames per second, which contain around five to six objects. To avoid a bottleneck in the classification part of the pipeline, the framework must predict around forty images per second. The use of deep learning also sets some initial hardware constraints on the robotic hardware. A graphics processing unit (GPU) is therefore inevitable to achieve reasonable timing demands. In regards to memory consumption, a standard VGG16 [104] network with image size $150 \times 150$ and batch size 32 was measured to use approximately 9 gigabytes when training. The minimum memory requirements should probably lie around this number.

## 1.3 Contribution

This thesis's main objective is to find an applicable unsupervised deep learning framework to detect and classify planktonic species. This work is a continuation of the specialization-project [100] which revealed some promising model architectures and several challenges with unsupervised training. Therefore, a secondary objective is to resolve these issues and

implement an unsupervised framework that improves the specialization-project results. Based on these goals, the objectives of this thesis can be summarized as:

I. Conduct a literature study to acquire insight from previous work and relevant machine learning applications in the plankton domain. Further assess the feasibility of adapting the current state of the art methods for plankton classification.

II. Implement and validate the most promising unsupervised deep models over plankton data sets and investigate their potential as feature extraction models.

III. Implement and validate different unsupervised clustering algorithms and estimate their ability to perform classification over the plankton data.

IV. Propose the building blocks of an unsupervised framework to detect and classify planktonic species.

V. Based on these studies, assess the possibilities for: 1. Integrating the model onto the AUV. 2. Use unsupervised learning in the labeling process to facilitate faster annotation and discovery of seen and unseen samples and soften the effort from expert biologists.

This thesis's contribution is to review and test several promising unsupervised deep learning frameworks for use in plankton classification. For this purpose, a significant literature review was conducted within the field of deep learning and computer vision. Furthermore, several state of the art unsupervised deep learning feature extraction models and a self-proposed rotation invariant autoencoder model were implemented and trained over plankton image data. A comprehensive test and validation regime was further created using real-world planktonic image data to assess and verify these models' performance and learned image representations. Finally, an unsupervised framework is introduced and evaluated on the AILARON data providing valuable insight into the applicability, the shortcomings, and possible future improvements of deep unsupervised learning within the AILARON project. The combined work of specialization-project and the master thesis led to new contributions to unsupervised learning of plankton taxa. Part of the detailed work was presented at Global OCEANS 2020: Singapore – U.S. Gulf Coast Conference[1] at the student competition track and is published in IEEE Xplore [101].

## 1.4 Outline

The rest of the thesis is organized as follows. Chapter 2 introduces the realm of planktonic species and presents the previous work in plankton sampling and classification. The last part of the chapter presents a number of relevant and readily available planktonic data sets that can be utilized in machine learning. Chapter 3 introduces the relevant theory within the field of artificial intelligence, providing the necessary background information

[1] OCEANS is a global marine conference and exhibition presented by the Marine Technology Society and the IEEE Oceanic Engineering Society. The conference gathers people from all over the world to discuss topics and recent trends in the marine sector. The official website is accessible at: https://gulfcoast20.oceansconference.org/

as well as a more comprehensive study of the deep learning domain. Chapter 4 presents the proposed unsupervised framework detailing its structure and components. Chapter 5 describes how the experiments were conducted and provides the relevant implementation details and code requirements. Chapter 6 presents the results of the experiments mentioned above, followed by a summary and reflection for each experiment. Chapter 7 provides a more general discussion answering the research questions and challenges provided in this chapter. Chapter 8 concludes this thesis by emphasizing the most important findings. Chapter 9 ends the thesis by presenting promising future directions and challenges which will be assessed in future work.

Chapter 2

# Previous work on the planktonic domain

This chapter gives an introduction to the realm of planktonic species and presents the fundamental work within plankton sampling and classification: First, a brief introduction to planktonic species, their definition and their environmental importance to nature are presented. Second, important traditional and modern sampling methods are explained, highlighting the possible advances which can by achieved by the AILARON system. Third, an evaluation of existing work for both supervised and unsupervised plankton classification is presented. This includes the previous work conducted in the specialization-project, explained in section 2.3.3, which this thesis builds upon. Lastly, a review of some of the available plankton datasets are presented.

## 2.1 Planktonic organisms and their importance to the environment

The term plankton derives from Greek meaning ”to drift” and refers to a collective group of organisms containing small plants, animals, bacteria, and viruses living in the water. Many of the species can change their depth through active swimming, but they are still drifting at the mercy of currents [13]. Planktonic species are usually very small in size ranging from microns to centimeters, but some species, including gelatinous jellyfish's, can be much larger. Plankton is generally divided into two groups. Phytoplankton mainly consists of single-celled organisms that can convert light energy into chemical energy. These organisms form the bottom of the marine food chain and are, therefore, often referred to as the ”grasses of the sea” [106, Suthers et al. p. 2]. Phytoplankton is, in turn, grazed by larger zooplankton, which are the main ocean primary consumers. Zooplankton

is a group of vast diversity, including groups such as millimeter-sized copepods and meter tall gelatinous jellyfish spending their whole life as plankton (holoplankton). Others, such as fish larvae, are zooplankton for only parts of their life (meroplankton).

Plankton is a crucial component of marine life and forms the base of the marine food chain. Phytoplankton stands for about 45% of the earth's primary production, producing oxygen and processing many of the pollutants disposed by humans [106]. Phytoplankton is such an essential part of the cycle of carbon, drawing carbon dioxide from the atmosphere. Grazing on the smaller phytoplankton, zooplankton is, in turn, "suitably sized food items for predators including commercially important fish and great whales" [13, Brierley et al. p 1].

The species have, in general, short generation time [109] and a short life span. Also, being sensitive towards minor changes in water components such as light, nutrients, pollution, and water flow can cause great and rapid changes in the diversity and population abundance of planktonic species. Rapid increase in phytoplankton population is called a bloom. Many blooms happen regularly and are vital to the survival of other species, such as fish larvae [48]. Blooms can, however, also impact the ecosystem negatively. Harmful algal blooms (HABs) can create water toxins or lead to localized depletion of oxygen, causing fish kills on wild and farmed fish [106]. The produced toxins can further accumulate in shellfish, becoming a direct threat to humans when consumed [34]. Environmental changes in the ecosystem also impact larger plankton species. Jellyfish blooms are becoming an increasing problem to the fishing industry, tourism and can even block coastal powers and seawater intakes, affecting human infrastructure [76].

Changes in plankton distributions can also provide valuable information about water quality and long term climate changes. As such, Suthers et al. [106] refers to plankton as "canaries-in-a-coal mine" [106, Suthers et al. p. 1], stating the effectiveness of using plankton as at tool for monitoring water quality. In comparison to nutrient analysis of water quality, plankton is a water quality indicator providing information about past and current water disturbances [9, 109]. Furthermore, because few planktonic species are commercially exploited and are affected by changing temperatures and ocean currents, scientists believe long term changes of plankton distributions to be an excellent indicator of the environmental impact of climate changes [48].

## 2.2 Traditional and modern sampling approaches

Depending on the precision and available budget, several sampling methods are available. Buckets or water bottles are a cheap and effective way of gathering point samples providing information of present species. The sampling area can be increased by utilizing a towed net or fine-grained mesh at the cost of missing the smallest species. Another technique is measuring watercolor using satellite images, making it possible to cover vast areas and provide a broader description of large plankton distributions [106]. However, this comes at the cost of less precise measurements. Finally, one of the most important instruments currently in use is the *continuous plankton recorder* which have been used to

monitor plankton taxa for more than seventy years. This instrument can be towed over vast distances and provides invaluable information about phyto- and zooplankton abundance [48].

In recent years underwater imaging technology has progressed, making optical in-situ sampling much more effective. The Imaging FlowCytobot [86] can work unattended for months at a stationary point providing continuous long-term image samples. Increasing the sampling area, the In Situ Ichthyoplankton Imaging System (ISIIS) [20] can be towed behind a ship capturing species in the range from 1 mm in size and larger. By adopting such methods onto existing underwater vehicles, such as the zooglider [85], a more precise estimation of plankton abundance and distribution can be achieved. Building on this work, the AILARON project strives to make an autonomous sampling and classification platform [26, 27].

## 2.3 Previous work on plankton classification

Traditional plankton classification is based on specialized human labeling using microscopy. Such work is very labor-intensive and requires highly specialized individuals [106]. The introduction of new image-based sampling methods offers an exciting solution as machine learning algorithms have shown excellent performance in image classification.

The most successful approaches are tied to the supervised classification domain, where the models rely on labeled training data. Albeit often resulting in well-performing classification algorithms, the labeling effort can be extensive and put much load on existing human resources to acquire the training set. Furthermore, the supervised model's dependency on existing labeled samples is generally limiting their classification ability on new unseen objects.

The unsupervised classification domain requires no prior knowledge of the correct class labels, which significantly reduces labeling effort. For classification, the models are generally built around feature extraction and then clustering of the extracted features. This approach might help discover new unseen classes because the new class species, having different feature characteristics, are likely to form new clusters.

Machine learning can provide a robust and efficient classification of plankton at a low cost. Still, there are some drawbacks and challenges that are important to understand. Single phytoplankton objects are usually too small for image sampling methods. However, they often form algal groups such as filaments, chains, or colonies, which are easier to spot, at the cost of varying size and shape characteristics. Sampling of gelatinous species are generally improved using image-based methods as the soft-bodied organisms often are destroyed using net sampling. Unfortunately, their close to water densities and almost transparent body types typically make image prediction difficult. Furthermore, machine learning is currently not ideal for differentiating the same class species into more fine-grained categories. For instance, different types of copepod species are typically determined based on the design of the fifth leg [106]. Such small distinctions are complicated to differentiate

given the current image quality and capabilities of machine learning models.

### 2.3.1 Supervised plankton classification

The novel approaches of supervised learning are based upon handcrafted feature extraction and then training a separate machine learning algorithm. In 2007 Sosik et al. provided a support vector machine (SVM) classifier achieving up to 88% on selected data [105]. Gorsky et al. [45] trained a random forest algorithm over a larger number of classes achieving comparable classification results. Other studies related to these approaches are well summarized by Gonzales et al. [40].

Recently, approaches using supervised convolutional neural network (CNN) architectures have overcome the traditional machine learning approaches except in some special cases. Zheng et al. [123] used handcrafted features to train a multiple kernel learning algorithm. Their approach achieves worse than deep learning models on large data sets, but can also be applied successfully on much smaller data sets. Solving a similar problem, Rodrigues et al. [95] trains a CNN network on a large plankton data set. The CNN is then used to extract features from a smaller data set to train an SVM algorithm.

However, the most effective classifiers on plankton imagery given enough labeled data is conventional CNN networks. In 2015 Orenstein et al. set the baseline for their WHOI plankton data set [87] using a VGG16 [104] as a backbone network. They achieved an accuracy of 93.8 % over 70 classes of plankton. However, since the data set have high class imbalance, the unweighted F1 score was only 0.42. Lee et al. [68] later improved the model results on small sized classes, while Lumini et al. [77] achieved better performance by combining deep learning with traditional feature extraction.

In 2015, the Data Science Bowl[1] competition challenged people to come up with a plankton classification framework. Using the Kaggle data set [21], the winners achieved 81.5% accuracy using an ensemble of over 40 deep CNN networks. The performance indicates some of the challenges of plankton imagery compared to other computer vision tasks as the accuracy score is often much higher. Furthermore, for real-world classification, ensembles of networks might not be a feasible solution. Constraints such as low memory consumption and fast prediction rates might require smaller networks. In such a manner, Py et al. [90] proposes an inception [107] inspired network and Li et al. [71] a ResNet [49] model which can achieve close to state of the art on the Kaggle data set.

### 2.3.2 Unsupervised plankton classification

Plankton classification using unsupervised algorithms is still a relatively unexplored field of research. In 2020, Pastore et al. [88] proposed the *plankton classifier* being a set of models for unsupervised classification and detection of plankton. Their proposed data pipeline

[1]The Data Science Bowl, `https://datasciencebowl.com/`, is a yearly data science competition presented by Booz Allen Hamilton and Kaggle. The competition brings together participants from all over the world and challenges them to tackle real-world problems using data and technology. Examples of previous events include plankton classification, diagnostics of heart disease, and diagnostics of lung cancer.

starts by reducing large images into cropped versions containing only a single planktonic organism. From these images, handcrafted features are extracted and fed into a fuzzy-K-means clustering algorithm. This pipeline achieved well over an image data set containing 640 images, scoring an accuracy of 89%. Furthermore, an extension to this framework is proposed training supervised models using the k-means clustering labels. This approach is tested on a selected uniform set of the WHOI data set, achieving a classification accuracy of 63% using a *random forest classifier*.

Comparing to the *plankton classifier* framework, a sub-goal of this thesis is the extraction of features provided by a self-learned deep learning network. The unsupervised feature learning is in this regard more related to the deep learning algorithms proposed by Kyzminykh et al. [65] and Wang et al. [114], that are presented next.

Kuzminykh et al. [65] proposed a variation of an autoencoder architecture to improve image classification using their self-proposed feature extraction technique. Their framework was, among others, tested on the Kaggle data set. Importantly, their objective was not to find an appropriate framework for plankton classification but to prove the efficiency of their self-proposed gram pooling technique. In this regard, the plankton data, containing objects positioned with different translations and rotations proved valuable to demonstrate their framework's capability. Since their focus was not to achieve state of the art results in image classification, they restricted the autoencoder into a shallow architecture. The model was first trained to learn an adequate data representation. Assuming that a better classification score is due to a better latent representation of the data, a supervised classifier was then trained on top of the feature extraction network. The overall framework showed promising results, achieving an accuracy of 62.2%, which was nearly a 10% increase to a traditional autoencoder. Furthermore, their framework proved the value of rotation invariant networks in regards to plankton classification.

The *CGAN-plankton* proposed by Wang et al. [114] is made for solving class imbalance problems by creating new synthetic images that look at least superficially realistic. A generator network creating fake data competes against a discriminator network deciding whether the image is real or fake. The ability to create authentic images is thus connected to the networks ability to learn the important structures and features of the training data. Whereas their main contribution is the creation of realistic images, they also show their models ability to improve classification. Using the weights obtained by distinguishing real from fake data in the discriminator, the model is fine-tuned using supervised learning to improve the classification accuracy over CNN networks trained from scratch or using transfer learning from other non-related data sets.

### 2.3.3 Specialization-project

The specialization-project [100] was performed during the spring of 2020 by the undersigned. The project's main aim was to gain deeper insight into the domain of unsupervised deep learning through a literature study and the exploration and implementation of two basic frameworks over existing plankton data sets. The proposed frameworks were an adaption of the *autoencoder* model of [92] and the *deep convolutional embedded cluster-*

*ing* of Guo et al [46].

In the specialization-project work, the autoencoder (AE) was built symmetrically using an encoder function, converting the input data to a lower dimension and then converting it back to its original shape using the opposing decoder function. The structure is highly flexible, making it possible to test several regularization schemes and switch backbone network testing, respectively: 1. a fully connected neural network (FC), 2. VGG16 [104] and 3. A 5-layer CNN (5-Conv) [98]. After training, the encoder part was used for feature extraction, and classification was performed using a separate clustering algorithm. A deep convolutional feature extractor proved capable of improving both a k-means clustering algorithm from 44% accuracy up to 65% on five selected classes from the data set. Due to the encoder's non-linear output, a spectral clustering (SC) algorithm outperformed the k-means scoring up to 76% on the same selection of data. Albeit outperforming the k-means in terms of accuracy, the nearest neighbor selection performed by SC has high time complexity and requires much support data. Such requirements might be infeasible for the in-situ classification.

Noteworthy, an autoencoder algorithm is constructed with the purpose of image reconstruction and might not provide optimal traits for classification. Guo et al. [46] proposed an autoencoder with an incorporated k-means based clustering layer to improve the feature extractions in regards to classification. The network can then be jointly trained using reconstruction- and classification loss. 71% accuracy was achieved over the five selected classes using the same regularization and backbone networks. Compared to the autoencoder + k-means approach, the feature space became more optimal for clustering models using the euclidean distance metric. Furthermore, the DCEC has only a slightly higher time complexity making it the ideal choice for in-situ classification.

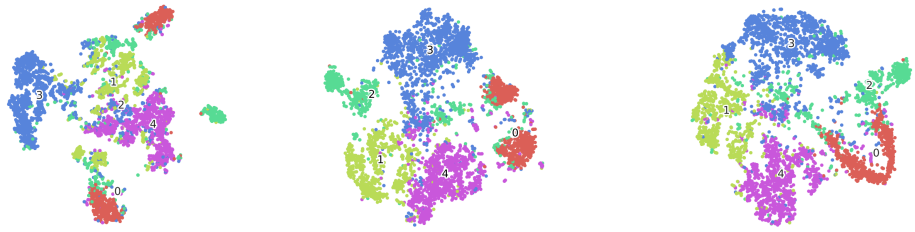


**Figure 2.1:** The figure depict the t-SNE visualization of the different autoencoder feature representations. **Left image:** Features extracted from a fully connected (FC) network. **Middle image:** Features extracted from a convolutional autoencoder (AE). **Right image:** Features extracted from a DCEC model. Illustrated in [100, Salvesen p. 35-38 ]

Figure 2.1 shows the quality of the learned feature representations extracted by a fully connected, an autoencoder, and a DCEC model, respectively. The fully connected model is clearly worst, proving the power of using an architecture with convolutional layers. For the deep convolutional autoencoders, the space only differed slightly depending on the backbone network. This leads to the belief that it is the latent layer choice, such as a GAP or

sparse layer, which is more critical for the performance. Another interesting discovery was the separation of red class points into two classes, depending on their orientation. Class separation is problematic because the unsupervised clustering algorithm can be tricked into believing several more classes exist.

## 2.4 Plankton datasets

A key factor for the performance of deep learning algorithms is the data on which they are trained. In this regard, training on lousy input data, even the most superior deep learning networks can achieve next to nothing. For image recognition, aspects such as image quality, data set size, and uniform class distribution are fundamental parts of a good performing neural network. For supervised learning methods, an additional requirement is also that the labeling accuracy for the priorly defined correct classes is high. The following sections give a description of some of the existing and readily available plankton data sets. Albeit images of plankton data, the data sets contain very different images. Therefore, it is not certain that a model producing good results on one data set can be transferred to another data set. Lastly, it is worth mentioning that the labeling quality of human domain level experts is much debated in the plankton community[2].

### 2.4.1 AILARON

The specialization-project provided concept models tested over publicly available data sets. Albeit applicable over these data sets, it is not given that the models will perform well when applied to the autonomous in-situ sampling and classification pipeline. Therefore, it is important to test the model on data with similar characteristics as the data that appears in the real task environment. A data set captured by the autonomous under water vehicle (AUV) is currently under construction and will provide valuable insight into the actual model performance. For now, the test data is captured using the same camera system, but the system is fixed to a stationary profiler. Still, it is believed that the captured images are very similar to the images captured by the moving underwater vessel.

**Figure 2.2:** Examples extracted from random classes of the AILARON data set. Observe that the different samples vary greatly in size and same class images are taken using different image resolutions.

[2]The quality of the labeling regarding human experts ability to identify planktonic species correctly is of much debate. Culverhouse et al. [24] set a much-referenced benchmark reporting 67–83% self-consistency over planktonic data. Luo et al. [78] argue that this benchmark is too low, given that Culverhouse et al. experienced a complicated identification task trying to differentiate within class species from each other. According to Luo et al. [78], to distinguish broad plankton classes is much easier, making a potential human labeling benchmark closer to 90%.

The data was captured in the Trondheim fjord between 2015 and 2017, using the SINTEF developed SilCam [26] camera. Every second, the camera system takes up to 7 overview photos, see figure 2.3, containing particles and organisms ranging from 50$\mu$m to several cm in size. Note that due to the telecentric receiving optics, the sample size is the same regardless of the distance to the camera. The overview photos are then cropped into regions of interest seen in figure 2.2, providing smaller images that can be labeled by expert biologists.

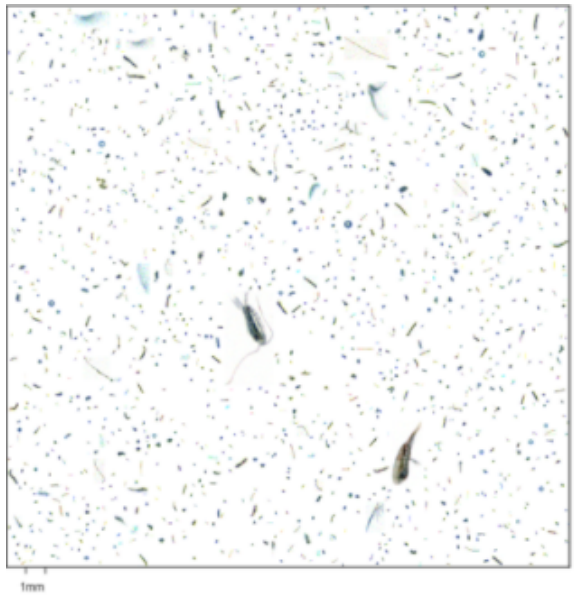
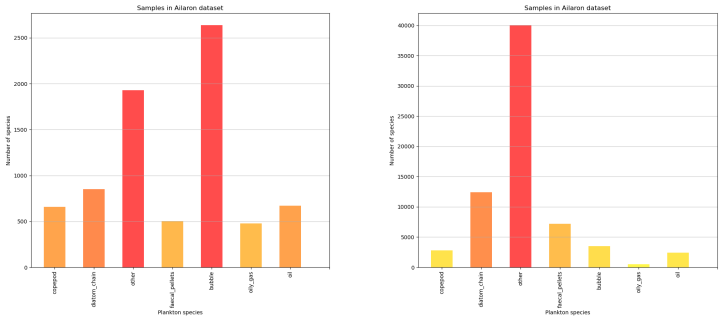



**Figure 2.3:** Raw overview image taken with the SilCam. From these images, areas of interest are cropped into regions of interest. Illustrated in [25, Davies]

The data is split into two data sets, denoted by the prefixes 'DB1' and 'DB2' in table 2.1. DB2 is a labeled data set containing all data samples. Unfortunately, the set suffers from an unknown number of misclassifications, making it unreliable for supervised training and validation. DB1 is a smaller subset of DB2 that is considered to have a much lower labeling error. The data set consists of 7 classes, namely *bubble*, *copepods*, *diatom chains*, *faecal pellets*, *oily gas*, *oil* and *other*. As can be seen in fig 2.4, the distribution of samples is uneven. Nearly 60% of the set consists of the 'others' class, whereas only 0.67% of the data is from the 'oily gas' class. Each image is in RGB format with pixel values ranging from 0-255. The image dimensions are inconsistent, some having only a width or height of 2 pixels, whereas the largest images have dimensions of more than 1300 pixels.

**Table 2.1:** Number of samples and classes in the AILARON data sets

| Data set | Number of species | Number of classes |
|---|---|---|
| DB1 | 7728 | 7 |
| DB2 | 68792 | 7 |




**(a)** DB1 **(b)** DB2

**Figure 2.4:** Histogram of AILARON-DB1 (a) and AILARON-DB2 (b) showing the class distribution of the seven different AILARON data classes. Observe that the *Other* class contains nearly two-thirds of the data set.

An apparent drawback of the current data set version is the small number of classes and, of these, an even smaller number of relevant plankton classes. Furthermore, many of the appropriate images containing plankton are taken using different image resolutions. This results in same class images having very different feature characteristics. This can be a real challenge in unsupervised representational learning because classes can be split based on the different traits. Other problems include the large class imbalance and a seemingly high number of misclassifications, making it more challenging to validate the models succeeding the training. This leads to the conclusion that other plankton data sets might also provide important information towards choosing an applicable model.

### 2.4.2 Luo plankton set

The *Luo plankton set* is a dataset used in the Luo et al. [78] paper to build their plankton segmentation and classification pipeline. The dataset strongly resembles the *Kaggle* dataset using the same ISIIS system and similar preprocessing steps. However, the data collection was done in 2011 between July and August in the northern part of the Gulf of Mexico. The dataset contains over 40 000 images sorted into 108 classes. Similarly to the *Kaggle* dataset, some classes contain under 20 species, while others contain more than 2000. The images' size is equivalently inconsistent, varying from 20x20 pixels to over

400x400 pixel-wise size.

### 2.4.3 Pastore plankton set

The *Pastore plankton set* is a dataset used in the Pastore et al. paper [88] for training their unsupervised plankton classifier. The data was collected using lensless microscopy and was originally a set of RGB colored videos containing several freshwater planktonic species. From these video frames, images are cropped out to include only single species. The dataset contains 5000 images sorted evenly into ten classes. The images are of varying sizes. The smallest are under $40 \times 40$ pixels, while the largest are $128 \times 128$ pixels.

### 2.4.4 WHOI

The *Woods Hole Oceanographic Institution (WHOI)* [87] dataset was collected using the Imaging FlowCytobot (IFCB) [86] in the Woods Hole Harbor water. The sampling began in 2006 and resulted in a sample set of more than 700 million samples in 2014. A set containing over 3.5 million images of marine plankton was extracted from random time frames from two weeks period and labeled into 103 categories from this data. The data is thus a representation of "the aggregated natural variability of the class distributions over time" [87, Orenstein et al. p. 1]. As might be expected, the set thus contains classes of varying sizes. The *mix class*, containing small undifferentiated particles, covers approximately 60% of the set, whereas, for instance, the *akashiwo* class only contains about 0.01% of the data. Examples of species are depicted in figure 2.5 Providing a starting point for plankton classification, Orenstein et al. [87] also set a baseline using different machine learning classifiers, achieving an accuracy of 93.8% using a CNN network.

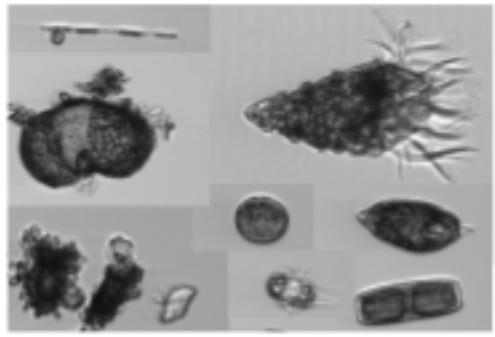


**Figure 2.5:** Examples extracted from random classes of the WHOI [87] data set. Observe that the different samples vary greatly in size.

### 2.4.5 Kaggle

The *Kaggle - national science bowl* [21] dataset was collected using the In Situ Ichthyoplankton Imaging System (ISIIS) [20] from the Straits of Florida. Compared to the static IFCB system, the ISIIS system is towed by a boat and can collect samples from much larger areas. For this dataset, nearly 50 million samples were collected between May and June in 2014. From these, a set of 30 000 images were labeled by Hatfield marine scientists and published as a training set for the *Kaggle - data science bowl* 2015 competition. An additional unlabeled test set containing over 160 000 images was also provided to validate the competing team's performance. The images are categorized into 121 classes, some of which have fewer than ten species. Because the images are segmented from much larger frames, they vary a lot in pixel-wise size. The smallest ones are only about $30 \times 30$ pixels, whereas the largest are over $400 \times 400$ pixels wide. As the dataset was used for a competition, it has provided great knowledge of supervised machine learning capabilities. The winners, *Deep Blue*, achieved an accuracy of over 80% using an ensemble of CNN networks. Given that human labeling accuracy is following the findings of Culverhouse et al. [24], the *Deep Blue* performance is close to the labeling accuracy of domain level experts.

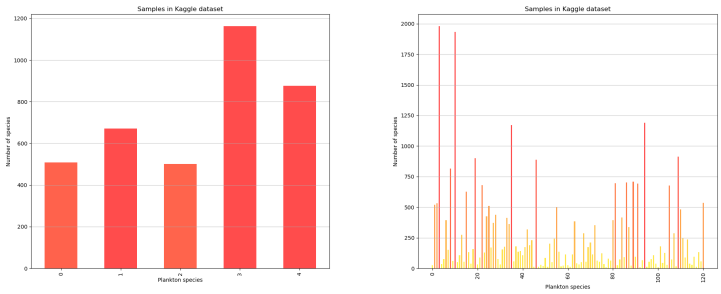
**Figure 2.6:** Examples extracted from random classes of the Kaggle [21] data set.

The excellent image quality, wide range of classes, and relatively large dataset size made it an ideal choice as an exploration set in the specialization-project [100]. Therefore, the dataset is also used in this work to make a fair comparison to the specialization-project algorithms. Following the specialization-project implementation, the data is split into two data sets, denoted by the prefixes 'DB1' and 'DB2' in table 2.2. DB2 refers to the full

Kaggle dataset containing all the labeled data samples. The histogram in fig. 2.7 shows the DB2 data distribution. Evidently, the set is relatively imbalanced, with the smallest classes containing close to zero samples, while the largest classes contain close to two thousand species.

**Table 2.2:** Number of samples and classes in the Kaggle data sets

| Data set | Number of species | Number of classes |
|---|---|---|
| DB1 | 3720 | 5 |
| DB2 | 30336 | 121 |




**(a)** DB1 **(b)** DB2

**Figure 2.7:** a): Showing the class distribution of the five classes in the Kaggle DB1 data set. (b): Showing the class distribution of the 121 different classes in the Kaggle DB2 data set.

DB1 is a subset of DB2 containing only five classes. This was created to make it easier to visualize and compare the quality of different feature extractions. The five classes are depicted in fig. 2.8 being *"acantharia protist"*, *"copepod calanoid"*, *"diatom chain string"*, *"faecal pellet"* and *"protist other"*. The classes were selected with the goal of finding species with a diverse set of features. Furthermore, the set has is more even in regards to the number of species per class.

**(a)** Achantaria **(b)** Copepod **(c)** Diatom chain **(d)** Feacal pellet **(e)** Protist

**Figure 2.8:** Examples of species from Kaggle DB1. From left to right: *"acantharia protist"*, *"copepod calanoid"*, *"diatom chain string"*, *"faecal pellet"* and *"protist other"*.

Chapter 3

# Artificial intelligence

Artificial intelligence (AI) is a field within computer science aiming to achieve machines that can mimic human intelligence through rational thinking or humanlike performance [97]. A more formal definition is the ”system’s ability to correctly interpret external data, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation” [55, Kaplan et al. p. 1]. From its early beginnings and great enthusiasm in the nineteen fifties, the field has gone through periods of significant advancements and renewed optimism into periods of recession and subsequent disappointment. In 1997 a breakthrough seemed close when *Deep Blue*, an IBM developed AI, defeated chess world champion Garry Kasparov. Albeit having slower progression and being less impactful than expected, AI experienced a boom at the beginning of the new century. In 2008 Google launched a successful speech recognition algorithm available for all smartphone users. In 2011*Watson* proved capable of understanding complex questions beating humans in a game of Jeopardy. As of today, with the availability of more extensive data set sizes and increased computational power AI is now an integrated part of fields such as computer vision, speech recognition, machine translation, and autonomous vehicles [97].

This chapter gives an introduction to the theoretical background within the field of AI and computer vision. The goal of this chapter is not to dwell on theoretical details but rather to provide an overview of the key principles of the field. First, section 3.1 - 3.3 provides the relevant background information about the field of AI and machine learning, then section 3.4-3.7 dives into the deep learning domain.

## 3.1 Disciplines of artificial intelligence

Because of the broad definition of AI, there exists a wide variety of disciplines and attempts to achieve machine intelligence. These domains are broadly depicted in figure 3.1,

showing relations between important AI disciplines. Many projects have focused on making a machine learn to perform in a limited domain known as a ”microworld” where the environment's formal rules are hard-coded a priory. However, such approaches are very limiting because the legal restrictions can be impossible to define appropriately. **Machine learning** approaches avoid such explicit domain definitions and make their own logic by finding patterns in the input data [43]. Given sufficient data, such models can predict, for instance, future wine prices or risk of diabetes with better accuracy than those achieved by humans. However, the machine learning model's capability to find relationships and patterns in the data, is highly dependent on the data having a good set of feature representations. In many cases, it is difficult to find such relationships from raw data and good performance is thus reliant on the programmer's ability to design a good set of feature representations. Models that can learn to extract such features themselves are known as **representational learning** models. In domains such as computer vision and speech recognition, where it can be challenging to provide hand-crafted features of the image concepts, multilayered neural networks or **deep learning** models have proven incredibly successful.

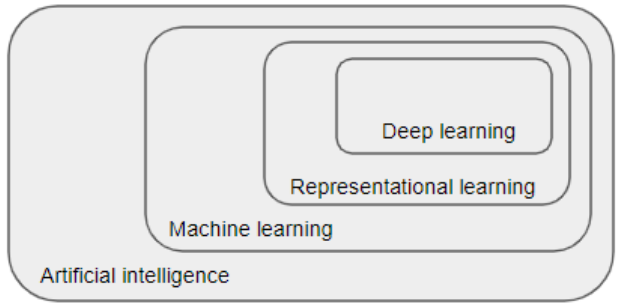



**Figure 3.1:** Venn diagram depicting the relationship between various domains within artificial intelligence.

## 3.2 Machine learning basics

According to Mitchell et al., machine learning deals with computer programs that can improve their performance through experience [83]. The machine's capability of “learning” refers to the algorithms ability to find statistical patterns in the provided training data. The self-acquired knowledge can then be used to predict outcomes on data the model has not seen before. Machine learning has proven especially adequate in domains where the information load is huge or precise commands are hard for the programmer to define. This section gives a brief introduction to the field of machine learning, starting with the basic terminology and concepts. Then, the most important machine learning algorithms utilized in this thesis are presented.

### 3.2.1 Machine learning approaches

Depending on the available data and the kind of problem, machine learning is traditionally divided into three main categories, namely: supervised, unsupervised, and reinforcement learning. Some models are also overlapping, making semi-supervised learning a potential fourth category.

#### Supervised learning

The objective of **supervised learning** is to find a function which approximates an unknown function that maps input data into a corresponding output. To learn this function, the model is trained over samples consisting of an input vector and a corresponding output label [97]. For instance, in computer vision, the input data set typically consists of input images along with their corresponding ground truth labels. Thus, the training strategy is to search for image patterns or commonalities between images, so that the same pictures end in the same class. If the model predictions are far from the correct class labels, the search continues until the model cannot improve its prediction ability. Its most significant strength is also the biggest weakness as gathering and labeling large loads of data can be very costly and time-consuming. Today, supervised learning is successfully applied to domains such as computer vision, speech recognition, spam detection, banking, and market analysis [72].

#### Unsupervised learning

The task of learning the properties of the data without any knowledge of the ground truth labels is called **unsupervised learning**. The goal is typically to observe structures or useful patterns in the data for which no prior knowledge of the data is required. Machine learning models, generally improve their performance when trained over large amounts of data [43]. Thus, unsupervised models can in theory surpass the performance of strictly supervised models by utilizing the vast amount of existing unlabeled data. Still, models trained in a supervised fashion are, in general, superior except in very special scenarios. Today, as stated by Russel et al., clustering is one of the main unsupervised tasks [97] aiming to find useful commonalities within groups of data. Another typical use case is dimensional reduction, where high dimensional data is mapped down to a few dimensions for visualization or storage purposes [10].

#### Reinforcement learning

The goal of **reinforcement learning** is to find appropriate actions for a given set of circumstances that maximizes the reward. The method is strongly influenced by real-life scenarios where players learn to behave in their surrounding environment through experience. Similarly to a real-world environment, there are usually no labels describing an optimal set of actions for a given scenario. Instead, the model has to discover a set of actions that lead to an ultimate goal through trial and error [10]. The model must thus choose its decisions based on previous experience or try new actions to acquire new knowledge. Typically the final goal will require many different actions over a long time period. The model must thus choose actions that might not give an immediate reward but which are profitable in

the long run. Reinforcement learning methods are typically applied in automation robotics and computer games.

#### Semi-supervised learning

In practice, the distinction between machine learning approaches are not so transparent [97]. In many situations, it is possible to acquire at least a small amount of ground truth labels. However, these labels might be of varying quality, and one might have a corresponding big unlabeled data set. **Semi-supervised learning** lives at this cross edge between the supervised and the unsupervised domain. Thus, the goal is to utilize both domains' advantages, for instance, by learning the patterns and data structures using unsupervised learning and then fine-tune over the labeled data. Semi-supervised learning methods are typically applied in real-life applications to speed up the image annotation task or situations where it is difficult to obtain a large labeled data set.

### 3.2.2 Generalization, over- and underfitting

The goal of a machine learning algorithm is to find a function that best fits the patterns and structures of the data. However, a function that perfectly matches the training data might turn up misleading because the machine learning algorithm has put too much emphasis on irrelevant or random features. This problem is referred to as **overfitting** and often occurs if the number of training samples is small or the model is too complex [43]. An overfitted model will therefore make erroneous predictions on unseen test data. Therefore, the requirement for a well-performing algorithm is to achieve low loss on the training data and achieve a similar performance on unseen test data. This property is called model **generalization**. Opposite to overfitting, **underfitting** happens if the model is incapable of finding an appropriate function to describe the data. It is thus not obtaining an adequate performance over the training data. The problem of underfitting typically arise from a model that is too simple to capture the complex structures within the data.

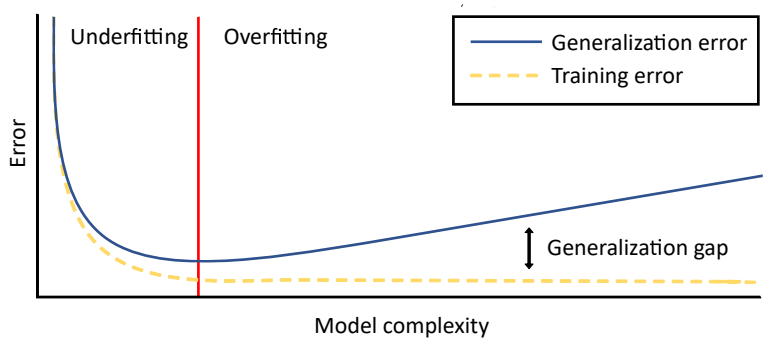



**Figure 3.2:** The graph shows the general relationship between error and model complexity. Underfitting can happen if the model is too simple and incapable of learning essential data features. Underfitting models can often significantly improve their performance by increasing model complexity. However, an overly complex model easily overfit learning irrelevant variations in the training data. This leads to poor performance over the test data or equivalently increased generalization error. The figure is inspired by the illustration in [43, Goodfellow et al. p. 115].

Figure 3.2 describes the trade-off between performance loss and model complexity showing the relation between generalization, over-, and underfitting. A simple model, for instance, only capable of fitting linear functions would suffer significantly when it is given data with polynomial features. The solution to reduce the error is to increase the model complexity allowing more complex function representations. However, increasing the complexity too much might result in a model capable of learning the noise or variance of the data [83], subsequently leading to overfitting and increased generalization error. The simplest but possibly most resource-intensive solution to avoid overfitting is to provide more data samples. Another common strategy, called **early stopping**, is to stop the training when the generalization gap increases over a predefined threshold.

### 3.2.3 Classification metrics

When deciding which machine learning algorithm is better, the best strategy is to compare their performance using predefined performance metrics. The choice of the metric must correspond well to the system application, and one must be able to distinguish between important and less important aspects of the performance. For instance, it might be of higher importance for an autonomous car to avoid collision than always being on schedule. For classification, the goal is often to find the model which can map most input data samples together with its corresponding ground-truth label. The most straightforward metric is the accuracy, given in equation 3.1, which measures the ratio of correctly classified samples divided by the total number of samples.

$$Acc = \frac{True\ positive + True\ negative}{Total} \tag{3.1}$$

Accuracy is easy to understand and is often a suitable performance indicator. However, if the goal is only to predict samples belonging to a specific class, accuracy is less indicative. Another issue appears if the data set have large class imbalance. For instance, in the WHOI dataset one class contains over 60% of the samples. Any model can thus achieve a decent score predicting all samples into the same category.

To avoid such issues, one can utilize metrics based on **precision**, equation 3.2a, and **recall**, equation 3.2b. Precision measures each class's quality, meaning that a predicted class with high precision has more correct than incorrect elements. Recall is, in contrast, a measure of quantity, calculating the number of same category elements that were correctly classified.

$$Precision = \frac{TP}{TP + FP} \tag{3.2a}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.2b}$$

Except in the case of a perfect prediction, there exists a trade-off between the two. High precision usually means a high *False Negative* rate, and thus a lower recall. Or equivalently high recall results in higher *False Positive* rate, thus leading to low precision measure. F1-score, given in equation 3.3, is a metric that balances precision and recall by calculating an average of the two. Compared to accuracy, the F1-score depends more on the number of misclassifications. This makes it more ideal for situations where the consequences of wrong predictions are greater or in cases with significant class imbalance.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.3}$$

For unsupervised models determining the prediction accuracy is not as simple as comparing the predicted labels against the ground truth labels. Obviously, an unsupervised prediction task for which there exists no ground truth labels make any comparison attempt futile. However, having a set of human annotated labels, a direct comparison might still be impossible because the unsupervised algorithm might have produced a different labeling order. As an example, the class "cat" in a human annotated set might be labeled as class "1" but in the clustering algorithm having no information about the relative labeling order decides that images with cat features belong to class "2". A clustering algorithm is thus likely to produce group labels that mismatch with the ground truth labels even if all samples were grouped correctly. In the case of semi supervised prediction with existing ground truth labels a fair comparison can be achieved by using one of the following algorithms:

- The **Hungarian method** was developed by Kuhn [64] in 1955, seeking to find the minimum cost of assigning a set of workers to a set of jobs. In the case of a multiclassification problem with labels 1,2,3 and predictions 1*,2*,3* the problem can be represented by the $3 \times 3$ adjacency matrix depicted in figure 3.3. In this case, the

goal is to find the mapping which assigns the predicted samples to the ground truth class that maximises the number of samples in each class. In other words, the algorithm finds the mapping that makes the sum of diagonal matrix values maximum. After the matching is done, the mapped predictions can be compared to the ground truth labels using the accuracy, precision and recall metrics described above.

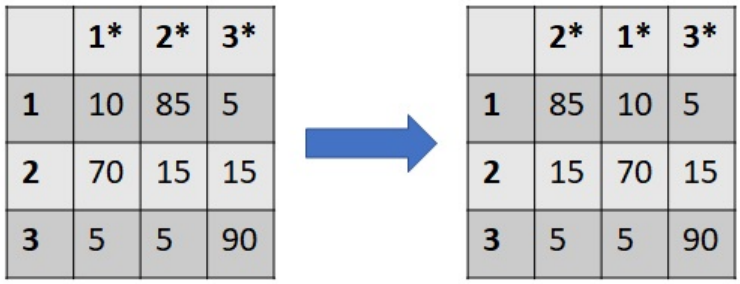



**Figure 3.3:** The figure depicts a typical unsupervised multiclassification problem where predictions 1*,2*,3* does not yet correspond to the labels 1,2,3. The Hungarian method [64] solves this by finding the mapping that best matches number of sample predictions and ground truth predictions.

- **Mutual info** is a measure of the shared information shared between the ground truth labels and the predictions. In other words, how much information does the data predictions $X$ tell us about the true labels $Y$. More mathematically mutual information, given in equation 3.4 calculates the reduction in entropy of the true labels $H(Y)$ given the entropy of $H(Y|X)$. Taking the normalized mutual information (NMI) of the variables, the assignments $X$ and $Y$ are totally independent if the NMI is zero whereas a NMI of one means they are perfectly correlated [15].

$$I(X;Y) = H(Y) - H(Y|X) \tag{3.4}$$

## 3.3 Computer vision and image classification

The field of **computer vision** involves machines capable of solving complicated tasks using extracted information from digital images, and videos [121]. Inspired by the human visual system, computer vision tasks typically seek to replicate human visual abilities [43] for the purpose of automation of repetitive and labor-intensive tasks. Industrial applications include inspection and quality control of manufactured goods, adaptive cruise control, fingerprint analysis, and medical computer-assisted diagnosis [5, 28]. The benefits of automation are the low operating costs of having machines that can work continuously

and provide consistent and improved results. Achieving such systems is, however, not trivial. Computer vision tasks that are seemingly easy for the human visual system have turned out to be very difficult to replicate in computer systems [28]. Additionally, many of the tasks have inherent real-time requirements, stressing the need for fast and expensive hardware.

The application of visual sensing machines are most commonly used within the field of object recognition. This domain can be divided into three categories: **Object classification** is used to predict the one class or the list of classes to which the image is likely to belong. Object classification is thus limited to predict only the ”most striking” object category in the image. To find more than one object and possibly mark the different objects with a bounding box, **object localization** is utilized. Finally, **object detection** is a combination of the former two approaches, first locating the presence of various objects and then categorizing them.

This section presents the necessary foundation for object classification within computer vision. The necessity and importance of feature extraction is presented first. This subsection includes an introduction to feature extraction approaches from the machine learning and deep learning domain. The last subsection is then dedicated to classification models that make predictions using the extracted features. A more comprehensive review of the deep learning domain is dedicated to section 3.4.

### 3.3.1 Feature extraction

For the task of image recognition, the easiest solution is to treat the input image as a vector and feed it into a machine learning algorithm. This, however, is not very effective because of the high dimensionality of real-world images. The problem of high dimensional data is referred to as *the curse of dimensionality*. First, machines typically suffer from high computational load and memory issues when trained on high dimensional data. However, *the curse of dimensionality* is more severe because machine learning algorithms easily overfit if the number of observations is much lower than the number of input dimensions [28]. Yet, real images may contain similarities and structures such as shapes and colors, making it possible to capture the essence of the image with much fewer parameters [5].

Thus, the task of object recognition relies heavily on the machine learning algorithms ability to extract the meaningful information from the images and discard the rest. This ability is referred to as **feature extraction**. In general, models produce either features describing the full image (global features), or features describing smaller regions or parts of the image (local features). Such choices depend on the classification, but normally global features are useful for rough segmentation and large data sets, whereas local features are used for more fine-grained classification [5].

#### Machine learning feature extraction

Early on, the conventional approach to image feature extraction was based upon the selection of hand-crafted image attributes such as object size, shape, and color. Albeit providing

adequate features for the specified task, such an approach require carefully designed filters and is dependant on domain knowledge [119]. Furthermore, since the extraction method is specialized to a specific domain, it cannot be adapted to other tasks. Therefore, a requirement for more effective implementation and utilization of feature extraction algorithms are their ability to adapt to a wide variety of image domains [5]. The *scale-invariant feature transform* (SIFT) [74] and the *Speeded Up Robust Features* (SURF) [6] models presented next are among the more versatile models. The SIFT is a method that detects key points from various image locations and then extracts distinctive features based on the gradient around these points. Lowe in [74, Lowe p. 1], claims the method to be "invariant to image scale and rotation" and to provide robust matching under different distortions such as lightning, noise, and affine transformations. The SURF model shares many commonalities with the SIFT algorithm detecting and extracting information from relevant key points within the image. However, it is much faster, and according to Bay et al. [6], the algorithm is more robust to different image transformations compared to SIFT. Yet, the best performing model can vary significantly from scenario to scenario. A thorough comparison of the two models over various image transformations and deformations was conducted by Karami et al., finding that the SIFT algorithm outperformed the SURF in most scenarios [56].

### Deep learning feature extraction

In recent years, deep artificial neural networks have achieved great results on many difficult image classification tasks[1]. Deep neural networks typically consists of several stacked layers extracting relevant information from the input before a final layer outputs a prediction. One of the biggest strengths of ANNs is the capability of the first layers to learn the important structures and patterns in the input data and gradually evolve these structures into finer concepts [43]. This learning process is called **representation learning** and similarly to a conventional feature extractor, the learned concepts can be extracted and used as features in an independent classification algorithm. For instance, for supervised deep learning models, the feature extraction is done by the first layers, whereas the last fully connected layer can be seen as a linear classifier. However, deep learning feature extraction as a single step is typically associated with the semi- or unsupervised domain where a deep learning model is trained separately from the classifier.

The typical example of an unsupervised representational learning model is the autoencoder. An autoencoder can learn important data representations by reducing the image to a few dimensions in its *encoder* part and then reconstruct the image using a mirrored architecture of the encoder, the *decoder* part. After training, the encoder part can be utilized as a feature extractor [4]. A remaining question, however, is whether such representational learning provides optimal features for classification. In his paper, Hartono claimed that the "labels of the data influence the internal organization" [47, Hartono, p. 3]. His research suggests that models trained for classification might emphasize features that better separates groups of data. With such notions in mind, Xie et al. [117], and Guo et al. [46] developed models which refines the autoencoder representational space by incorporating

[1] For more details about ANN properties see section 3.4.1.

a clustering layer into the model. Following this success, the most recent approaches to deep representational learning for classification utilizes conventional neural networks trained solely on ”fake” labels produced by unsupervised clustering algorithms.

### 3.3.2 Unsupervised classification based on feature extraction

Several methods are applicable to make predictions after utilizing the feature extraction technique. The most effective models are based on the supervised domain requiring a given amount of ground truth labels. However, this amount can, in general, be relatively low due to the reduced dimensionality of the data. A few of the more influential models include linear classifiers, support vector machines, naive Bayes, and decision tree algorithms.

In the unsupervised domain, having no access to ground truth labels, the classification technique is based on grouping examples based on their feature similarities [43]. These methods are referred to as **clustering** methods containing among others; connectivity-, partitional- and density-based models. Which algorithm to choose can vary greatly depending on the data because all the models, to some extent, make assumptions on the data structure. **Connectivity-based** clustering is built around the idea that neighboring points are likely belonging to the same class. Typically the algorithm starts with all objects belonging to singleton clusters. Then, the ”closest”[2] items or set of objects are merged into a new cluster in a repeated loop until all items belong to the same class. The result can then be depicted in a dendrogram, where distinct classes can be found by cutting the dendrogram at specified levels. An example is shown in figure 3.4 where 269 elements are gradually clustered together. Observe that the black line cuts the graph so that the elements are grouped into seven classes. However, a general problem with connectivity-based models is the time, and memory complexity limiting the application of those algorithms on large scale data sets [118].

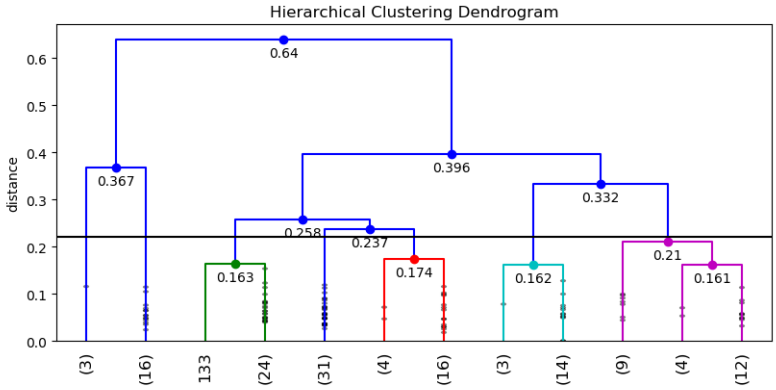



**Figure 3.4:** Showing an example of connectivity based clustering visualized in a dendrogram. Observe that the two closest groups/points are merged until finally all points belong in the same group.

[2]The term ”closest” is a relative measure depending on the choice of distance metrics. Usually, connectivity-based models use some variant of euclidean based measure, but other options are also applicable.

In contrast to the iterative grouping of similar points, **partitional-based** clustering methods seek to divide the data into $K$ distinct clusters. A classical algorithm is the K-means in [80], which aims to minimize the sum of squared errors between each cluster center and its assigned items. A great advantage in the K-means is the approximately linear time complexity, making K-means a good candidate for large scale datasets [118]. However, the conventional k-means algorithm easily fall short when the data is spherical or contains an uneven number of points per class. Moreover, K-means requires an initial number of clusters. In a purely unsupervised task, such knowledge can be challenging to obtain. To achieve better performance on non-linear data, k-means can be extended by transforming the data into a more k-means friendly space before the clustering. One such approach is the spectral clustering method (SC) [103]. SC starts by sorting the data points into a similarity matrix, for instance, by applying the k-nearest neighbor algorithm. Based on this matrix, the graph Laplacian and its corresponding eigenvalues and eigenvectors are calculated. Finally, the k-means algorithm is run using the $K$ first eigenvectors. According to Luxburg in [113], SC often yields much better results than models using euclidean based metrics. However, the time and memory complexity needed when constructing the similarity matrix makes the method less applicable to large data sets.

## 3.4 Deep Learning

The field of deep learning is an advanced part of the machine learning domain where the models are capable of solving difficult tasks by finding relationships and complicated patterns in vast amounts of data [108]. From its infancy, many have regarded deep learning models as the future of machine learning because their structure and processing of input signals are very similar to how processing is done in the biological brain [30]. The earliest attempts can be dated back to the late nineteen sixties, but especially due to the non-convex properties of the networks and respective difficulties of finding an adequate solution made the field less impact-full. Instead, research progressed on simpler models with convex optimization functions such as support vector machines [38]. Today, especially due to the increase in computational power and the increasing availability of large training data sets, the field is experiencing a new wave. In contrast to other machine learning approaches, deep learners have proven extremely useful in domains such as natural language processing and image recognition solving tasks that are ”easy for people to perform but hard for people to describe” [43, Goodfellow p. 1].

The term *deep* refers to the models consisting of many layers connected after each other forming a larger structure. The results provided by the network are thus coming from multiple concurrent interactions between connected neurons. Thus, deep learning models are often treated as ”black boxes” because it is very difficult to interpret how the results arrived based on the provided input. According to Goodfellow et al. [43], there exists two views on why deep learners are so successful. First, the networks are in contrast to other machine learning models capable of learning complicated concepts without explicit information or knowledge by gathering information from the simpler concepts and discoveries earlier in the network. Secondly, the networks can store state information such as preceding events, which possibly helps organize the data and make future predictions easier.

As of today, deep learning models have become the workhorse in data applications such as big data, speech recognition, and in computer vision [30]. In some of these domains, such as image recognition, models have already surpassed human capabilities for given data sets. With the growing computational power and continuous increase in the availability of large quantities of training data, the possibilities are still huge. How far the domain can reach is yet to be seen.

This section covers the necessary background information to understand the deep learning algorithms utilized in this thesis. Subsection 3.4.1-3.4.6 covers the fundamental principles and important building blocks of deep learning. Subsection 3.4.7-3.4.8 then presents some practical concerns including image pre-processing and basic training of neural networks. Finally, subsection 3.4.9 gives a brief presentation of the most used research data sets within deep learning. Note that the theory presented in this section is not restricted to the supervised or unsupervised domain. Distinctions between these domains are elaborated in the next sections 3.5 and 3.6 respectively.

### 3.4.1 Artificial neural network

An artificial neural network (ANN) are one of the most common network structures of deep learning. As the name suggest, artificial neural networks have drawn much inspiration from the biological brain consisting of a web of interconnected ”neurons” that obtains ”intelligent” behavior by propagating an input signal through the network [83]. These webs and neurons are, however, very simple compared to the biological brain. According to Mitchell et al., ”biological neurons output a complex time series of spikes” [83, Mitchell et al. p. 82] whereas a typical artificial ”neuron” only outputs a real-valued number based on a single **weight** and **bias** parameter. During training, these parameters can be adjusted to find a function approximation that explains the input data.

Figure 3.5 illustrates an artificial neuron which is connected to a number of previous nodes with output values $x_1 - x_n$. The neuron calculates a sum based on the input multiplied with a corresponding weight $w_1 - w_n$ and the bias parameter. Lastly, the value is sent through an activation function, which essentially decides if the node should activate or not. Observe that a neural network is essentially only a linear function of sums and multiplications. Thus, different activation functions are added to make the network capable of learning non-linear traits of the input data.

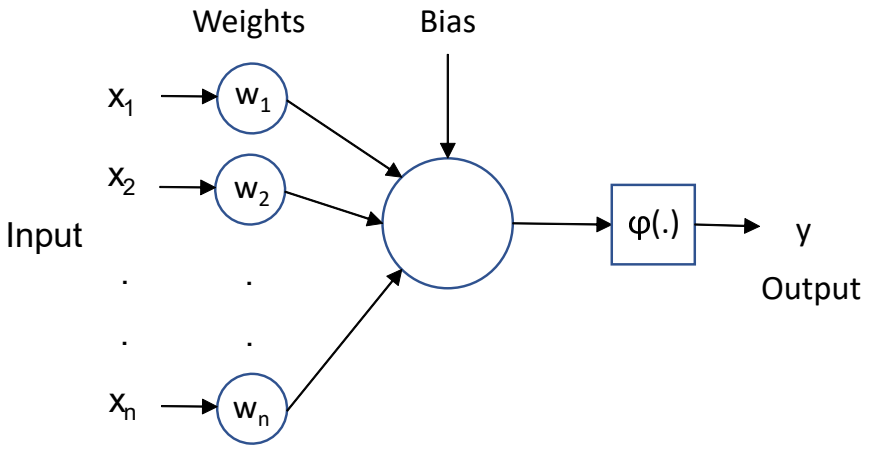



**Figure 3.5:** Model of a artificial neuron with non linear activation function. The figure is inspired by the illustration in [23, Csáji, p. 6].

Figure 3.6 shows a simple yet effective network structure called a **feed forward neural network** where information flows from input to output without any feedback. Such networks consist of an input layer receiving the input data signal. The input is then connected to many hidden layers that perform intermediate computations before the output layer computes the network prediction. To learn patterns in complex input data, the network capacity can be increased by increasing the number of nodes in the existing network or by making the model deeper by adding more hidden layers. In theory, a single-layered network with an unknown but finite number of parameters is sufficient for approximating a wide number of functions [23]. In practice, however, deeper models tend to be more easily optimized and hence obtaining better performance.

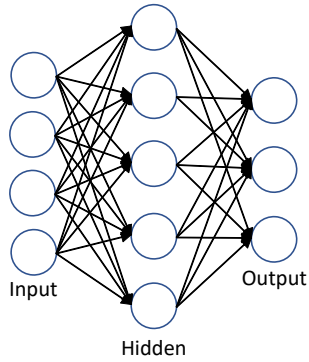



**Figure 3.6:** Feedforward neural network architecture showing one input, one output, and one hidden layer.

To obtain a well performing network over the given input data, requires careful tuning of

various parameters and selection of building blocks. These configurations are generally referred to as **hyperparameters** describing static values which are not adjusted by the learning algorithm [43]. Such options includes the number of hidden layers, types of layers, activation functions and learning rate. Important and commonly used hyperparameters and building blocks are discussed in the next sections.

### 3.4.2 Activation functions

As depicted in 3.5 the activation function succeeds the calculation between the input and learned parameters determining the neuron's final output response. In contrast to the node's weight and bias parameter, the activation function is predefined and normally a fixed function. Its purpose is to transform the node's linear summation into a non-linear output response that makes the network capable of approximating non-linear data.

There exists a wide variety of such functions with different strengths and weaknesses, making a choice heavily dependent on the application. The most commonly used are the *sigmoid* and the related *tanh* [30] which squeezes the output into the [0,-1] or [-1,-1] domain respectively. The downsides are, among other things, an added expense when computing gradients, and especially the problem referred to as *vanishing gradient problem*[3]. According to Goodfellow et al. [43], modern neural network architectures are therefore defaulting to the *rectified linear unit* (ReLU) activation. The ReLU is a piece-wise linear function, passing the identity value for all positive values and zero for all negative values. The gradient computation and subsequent convergence is thus much faster for ReLU activated networks. Furthermore, because of the allowance of large positive values, it generally suffers less from vanishing gradient descent. A well-known problem occurs if a neuron starts outputting zero regardless of input. This is referred to as *the dying ReLU problem* [75] potentially making extensions such as the leaky ReLU and parametric ReLU more tractable.

### 3.4.3 Neural network layers

An artificial neural network usually consists of a hierarchy of different types of layers stacked on top of each other. Fundamental blocks include the *fully connected* and the *convolutional* layers consisting of a set of connected nodes that are optimized during the training. Other layers such as the *max-pooling* layer does not contain trainable weights but outputs a deterministic output based on the input. This section introduces the most fundamental layers of artificial neural networks and their applications.

[3]The *vanishing gradient problem* is an issue in artificial neural networks which makes it difficult to update the network weights. The problem is especially encountered when using activation functions such as the *sigmoid* which "squash" the input signal into a small output. A node output is thus only slightly varying regardless of input values. For a deep model with many layers, a "large" response in the first layer will thus only result in a tiny response in the final output. As the function loss gets smaller, the gradients of the first layers can become exceedingly small or "vanish" thus making it impossible to refine these weights.

### Fully connected

A fully connected layer (FC), depicted in figure 3.7, consists of a set of neurons where each neuron are connected to all the outputs of the previous layer. Historically, the first neural networks where effectively utilizing architectures consisting of only such layers. Albeit, used effectively in many domains, fully connected layers require a large amount of parameters making them dependent on excessive memory and expensive computations.

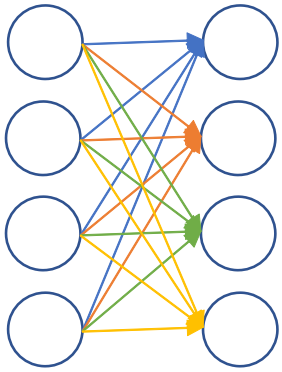


**Figure 3.7:** Figure showing all 16 connections (colored arrows) between a 4 valued input and a fully connected layer with 4 neurons.

### Convolutional

In real-world applications, data such as, for instance, time series or images most often contain structures where nearby points are more closely related than more distant ones. Fully connected layers ignore such local interrelations making the network suffer from the aforementioned ”curse of dimensionality” and excessive memory consumption for large input data. In contrast, **convolutional layers** are using multiple **kernels** which basically is a grid of connected weights. The kernels can thus learn the relationships between close data points, making them much more capable of exploiting local structures in the data. The output of the convolutional layer is calculated by ”sliding” the kernel over the input. For each ”slide,” a value measuring the correlation between the kernel and the current input image slice is computed, as depicted in figure 3.8, and stored in an **output feature map**. A feature map is thus a measure of the presence of a particular feature in the input. Architectures that utilize several convolutions layers have proven excellent at feature extraction. The first layers can learn how to extract low-level features such as corners, edges, and blobs of pixels, whereas later layers can merge such features into higher level concepts [83].

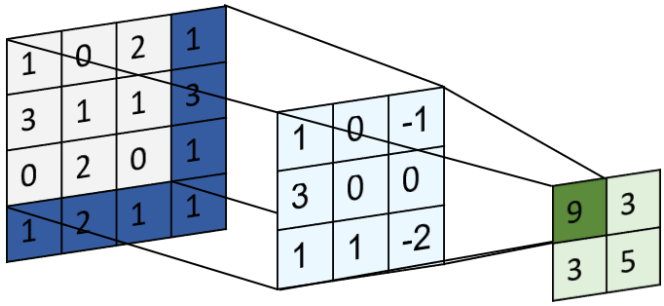



**Figure 3.8:** The figure shows a convolution operation using a $3 \times 3$ kernel (light blue) over a $4 \times 4$ input (dark blue) resulting in an output feature map (light green).

One of the main advantages of the convolutional layer is the capability of detecting important features using a minimal number of weights. *Sparse connectivity* and *parameter sharing* is obtained by keeping the kernel much smaller than the input, and the weights fixed across all image slices [43]. For instance, given an input of size $500 \times 500 \times 1$, a typical kernel of size $5 \times 5$ only contains 25 parameters, whereas a fully connected layer with one node would require $250000$ parameters. Furthermore, the convolution operation is approximately invariant to translation since a shifted input only results in the output feature map's values being equally shifted [83].

For the last decade, convolutional layers have become a fundamental building block of most neural networks being used on applications such as visual recognition and time series where nearby data is related. For instance, the layer is an important part of the tremendously successful *convolutional neural networks* (CNN), which are the current state of the art models for a wide variety of tasks.

### Pooling

Pooling layers are most commonly used together with a convolution layer to reduce the size of the output feature maps. Thus, the network needs fewer parameters for the following layers, meaning increased computational efficiency and reduced memory consumption [43]. Furthermore, the reduced input dimension forces the network to remove fine details and preserve only the most apparent features. Pooling is thus encouraging network generalizability as well as adding robustness to local translations [30]. The pooling operation is performed as depicted in 3.9 by concatenating nearby points using a predefined function. Basic pooling layers include the *max-pooling* which chooses the highest value in a region and drops the rest, and *average-pooling* which computes the average over that region. Additionally, a more extensive approach is the *global-pooling* operations, which reduces the output feature map into one single value.

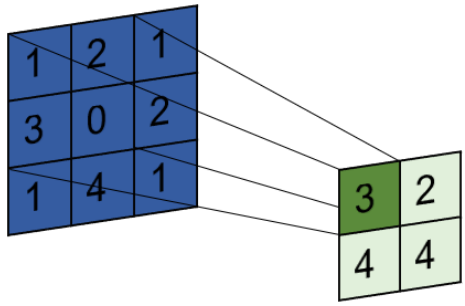



**Figure 3.9:** The figure shows a *max-pooling* operation on a $3 \times 3$ input image (dark blue) resulting in a $2 \times 2$ output (light green).

### Transposed convolutional layer

The transposed convolution layer is essentially the opposite of a convolution and pooling operation, resulting in an up-scaling of the input dimensions [32]. The operation, depicted in figure 3.10, is performed by first increasing the input using zero padding before a convolution operation is performed on the resulting map. In computer vision, the layers are an important part of deep models such as auto-encoders and generative models, which produces ”real” high dimensional images from low dimensional input vectors [36]. Equivalently to convolution layers, transposed convolutions generally provide better results at lower computational cost than up-scaling using fully connected layers.

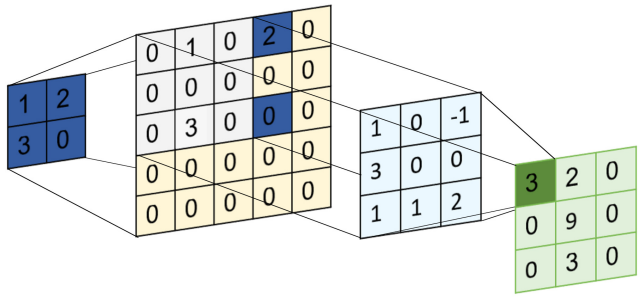



**Figure 3.10:** The figure shows a transposed convolution operation using a $2 \times 2$ input (dark blue). The input is first padded, then a convolution operation with a $3 \times 3$ kernel (light blue) is performed resulting in an $3 \times 3$ output feature map (light green).

### Group convolution and group pool

Observe, from section 3.4.3, that convolutional layers are approximately invariant to image translations. However, there is no inherent equivariance to other image symmetries, such as rotations and reflections. Instead, deep learners tend to learn rotated copies of the same filter to achieve partial invariance to translation [69]. Inspired by such observations, Cohen et al. proposed a new layer that can ”increase the expressive capacity of the net-

work without increasing the number of parameters" [19, Cohen et al. p. 1]. Their new convolution operation is depicted in figure 3.11 where parameters are shared over several kernels that are flipped ninety degrees in relation to each other. The four output feature maps resulting from the input convolution with the four kernels are thus in combination equivariant to ninety degrees rotations.

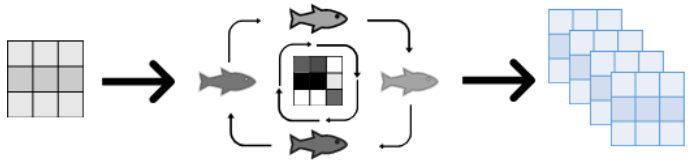


**Figure 3.11:** Example of an input being transformed into four feature maps by performing four convolutional operations with a ninety degree rotated kernel.

To get a more intuitive understanding, the normal convolution operation used in most deep learning frameworks, is defined in equation 3.5 based on the Cohen et al. notation [19, Cohen et al. p. 4]. Here, the kernel $K$ is shifted over an input $X$ resulting in an output feature map.

$$(X * K)[n] = \sum_{y \in \mathbb{Z}^2} X(y)K(n - y) \tag{3.5}$$

The group convolution operation, defined in equation 3.7, is quite similar to normal convolution. However, instead of performing the calculation once, the kernel $K$ is rotated ninety degrees using the parametrization defined in equation 3.6 where the integer $r$ sets the degrees of rotation, and $u$ and $v$ represents a point coordinate in the kernel matrix.

$$g = \begin{bmatrix} \cos r\pi/2 & -\sin r\pi/2 & u \\ \sin r\pi/2 & \cos r\pi/2 & v \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$(X * K)[g] = \sum_{y \in \mathbb{Z}^2} \sum_{m=1}^{4} X(y)K_m(g^{-1}y) \tag{3.7}$$

Observe that given a constant number of unique kernels, the number of convolution operations and resulting feature maps are quadrupled. This results in increased computational cost and memory consumption. Compared to a similar size convolutional network, group convolutional networks must therefore have fewer learnable kernels. The price is, thus, a reduction in the network capability of learning many unique input patterns. However, based on the assumption that rotated kernels are very useful in convolutional networks,

the group convolutions clearly increase network capacity without increasing the number of parameters.

*Group pooling* is a simple operation which can be utilized together with group convolutions to extract rotation-invariant features from the network. Observe that the output feature maps are only rotation invariant in groups of four. To extract rotation-invariant features, Baasveeling [112] therefore made an operation that combines rotated features maps into one by averaging their values.

### 3.4.4 Loss functions

The training of a deep learning model is generally a search for the weights that provides the ”best” output result. This measure is formalized using a **loss function** which ”reduces all the various good and bad aspects of a possibly complex system down to a single number” [93, Reed et al. p. 155]. The model results can be improved by searching for weights that minimize this value. The problem, of course, is to find a performance measure that fully represents the model goals. For supervised models, the measure is typically a direct comparison between the prediction and the ground truth labels. This can be captured by loss functions like *Cross-Entropy* for classification and *Mean squared error* (MSE) for regression tasks [93]. In the unsupervised domain, where ground truths are unspecified, the loss functions instead measures model aspects that are believed to be important for a well-performing model. For instance, MSE is used to compute the pixel by pixel reconstruction loss in autoencoders and *Kulback-Leibler divergence* to measure the divergence between the model distribution and, for example, a gaussian function.

### 3.4.5 Optimizers

Deep learning optimization refers to the search for model parameters that minimize the error defined by the loss function. The most used optimization strategies for deep learning models are built on *gradient descent*, where weight updates are performed using the gradient of the loss function with respect to model parameters [96]. However, the optimization poses multiple challenges, especially due to the highly non-convex error functions that are common for deep learning networks. Training can, therefore, quickly halt at *saddle points* or in multiple local minima [96]. *Stochastic gradient descent*, which is currently the most used optimization strategy [43], have proven especially susceptible to such challenges. Training is thus heavily dependent on fine-tuning parameters like learning rate, batch size, and momentum to achieve acceptable results. Seeking to make training more straightforward and more efficient optimization strategies such as *RMSProp*, *AdaDelta* and *Adam* have been proposed. According to Goodfellow et al. [43] there are, however, no consensus about the optimal choice.

### 3.4.6 Regularization

*Regularization* refers to a collection of methods that are designed to increase model generalization and avoid over-fitting [83]. For simple machine learning models, the classic example of such a method is an added constraint in the loss function that penalizes overly

complex models. The regularization thus forces the model to find a function that achieves a sufficiently small training loss while still keeping the model complexity low. However, deep learning models are almost always applied in complicated domains where the data approximation cannot be found by tuning the number of parameters. Instead, regularization in the deep learning domain focuses on methods which encourages generalization of large models with fixed parameters [43]. The most basic regularization methods are briefly mentioned in table 3.1 while data augmentation is detailed in the next sub-section.

**Table 3.1:** Commonly used deep learning regularization methods

| | Description |
|---|---|
| Weight decay | Weight decay is used to penalize large parameter values. The model complexity is thus constrained by forcing most weights close to zero which essentially reduces overfitting. |
| Batch normalization | Batch normalization applies rescaling and centering to the input of all layers. The rationale behind this is that small shifts in input can cause large changes in the input at deeper layers. By applying normalization, large shifts are avoided. Empirically batch normalization can provide faster learning rates and much better performance [53]. |
| Dropout | Dropout is a method that during training randomly selects a number of neurons that are temporarily removed from the network. Dropout can such force the network to rely on information from several parts of the network instead of only specific parts [43]. |
| Early stopping | Model performance over the training and validation data is generally improving up to a point where the model is improving on the training set at the expense of the validation. Stopping at this point is called early-stopping and is believed to help avoid overfitting. |

### Data augmentation

One of the most effective ways to improve model performance is to increase the number of training samples. Goodfellow et al. states that ”a rough rule of thumb is that a supervised deep-learning algorithm will generally achieve acceptable performance with around 5,000 labeled examples per category” and can surpass human capability having over 10 million samples [43, Goodfellow et al. p. 36]. In practice, however, data sets this large are hard to obtain and potentially require much labeling effort. Most deep learning algorithms are therefore trained on limited data sets.

To still achieve a sufficient amount of training examples, a common strategy is to augment the existing data [8]. ”New” training samples can thus be achieved by adding small changes to the original images such as rotations, translations, scaling, flips, and color trans-

formations. An example can be seen in figure 3.12. The goal is such to avoid over-fitting by adding new unseen samples that are not too different from the original class samples. It is worth noting that data augmentation only works if applied with caution. First of all, data augmentation is essentially a regularizer that hinders model over-fitting by avoiding the same image to appear multiple times. To adapt to new data, the model is still dependent on enough real samples to capture the data's essential features. Gathering enough data is, therefore, still an important requirement. Secondly, data augmentation must not alter the data in a way that does not represent the true environment. For instance, by flipping a car, the model would learn that cars can be upside down, which is not true in the real world.

**Figure 3.12:** Augmenting an image into nine different samples by adding random rotations, horizontal flips and translations.

### 3.4.7 Data quality and pre-processing

The goal of a deep learning model is to learn an approximation of the training data to perform well on unseen data. Needless to say, the complexity and nice traits of any model is therefore heavily dependent on the quality of the underlying data. Data sampling is thus an essential part of any successful deep learning framework.

Having obtained good quality data, pre-processing might be necessary to convert raw data into an acceptable format [2]. A typical data set might contain features with missing values, text strings, number values in different scopes, redundancies, and other problems, which makes it unfeasible to input raw data into the machine learning model. The first steps of pre-processing is thus, data cleaning, normalization and standardization of fea-

ture values. Afterward, pre-processing might contain work such as feature engineering to create valuable features based on domain knowledge.

In computer vision, data samples are usually not expected to have missing values or different types of features. However, images can still be of different shapes, contain noise and other intractable characteristics. Image pre-processing is therefore focusing on simply standardizing images to the same format and potentially incorporate operations like *gaussian smoothing* which reduces noise [37] and specific filters that enhances edges [126]. Other important pre-processing steps, for instance, in real time classification are dimensionality reduction to reduce computation speed drastically [83].

For deep learning models applied in computer vision, most of the more sophisticated pre-processing steps are usually skipped [43]. However, most models require that the input images have the same image dimensions making image resizing a common pre-processing step. Other operations as done in the *ResNet* [49] framework include pixel-wise rescaling to a small value domain such as [-1,1] and subtracting the the per-pixel mean. Generally speaking, image pre-processing is less valuable for deep learning models because the network itself can learn many of the human hand-designed pre-processing steps.

### 3.4.8 Training and tuning a neural network

In practical applications, the most successful approaches to deep learning are training a mainstream model over the data and subsequently tuning its hyper-parameters [43]. The first baseline is set by choosing a common network architecture, an appropriate loss function, and an optimization algorithm. The network is then trained to produce better result. Training is performed in iterations starting with a forward pass for which input data is flown through the network resulting in an output prediction. The loss between the output prediction and the optimal target is then calculated by the loss function. Finally, a backward pass is performed where the network weights are updated by gradient calculation to produce a smaller loss.

After setting the initial baseline, improving the model follows the graph in figure 3.2. If the initial model does not achieve a satisfying result, the model is likely underfitting due to bad data quality or because the model is too simple. After assessing that the image quality and the pre-processing steps are appropriate, the solution is to increase model complexity. When good training results are achieved, the model's ability to generalize is tested by applying the model to a separate test set. Note that good training results usually[4] implies a similar potential on the test set. However, deep models easily overfits to the training data leading to bad generalization traits and bad test results. Regularization constraints are therefore applied to increase generalization with the potential cost of increasing training error.

[4] A partial truth with many pitfalls. Because deep learning models are quite complex and difficult to interpret, they can achieve "good" results with unwanted behavior. For instance, learning "that all women wear red t-shirts" is an example of training over too few samples, thus adapting to useless features. Forgetting to shuffle the data is another mistake leading to perfect results by simply remembering the sample order.

### 3.4.9 Research datasets

Research and innovations within deep learning are mostly performed on a number of well known and readily available data sets. This section provides a brief introduction to the most important research sets and how model performance on these sets should be valued.

#### MNIST

The *MNIST* [67] data set, depicted in figure 3.13, is a collection of handwritten digits. The database contains a total of 70 000 images where all are black and white and $28 \times 28$ pixels large. For classification purposes *MNIST* is today considered a relatively simple set that gives good results for both traditional machine learning and neural network models. This year, the current state of the art performance for a single model was $99.79\%$ [14], and an off the shelf model can easily achieve over $99.0\%$. Despite this, the data set is still considered an important benchmark set used in many publications.

**Figure 3.13:** Example images taken at random from the research data sets. **Left image:** Images from the MNIST [67] data base. **Right image:** Images taken from the CIFAR10 [61] data base.

#### CIFAR 10

The *CIFAR 10* [61] data set, depicted in fig. 3.13, consists of 60000 images distributed into ten evenly sized classes. The images are RGB colored and $32 \times 32$ pixels large. The data set is generally considered more difficult than the likes of *MNIST* with state of the art neural networks only reaching 79% accuracy in 2010 [60]. Today, however, the state of the art neural network achieved 99.5% [31].

#### ImageNet

The *ImageNet* [29] data set is one of the biggest image databases in the world for object classification containing more than 15 million labeled samples belonging to over 22 thou-

sand categories [62]. The images are RGB colored and of varying size. From 2010 until 2017, a subset of this set containing 1.2 million images and 1000 categories has been used every year in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). It is on this subset many of the famous neural network benchmarks are set. In 2012, AlexNet [62] achieved a top-5 accuracy of 84.7%. In 2015 ResNet [49] achieved 96.43% exceeding human labeling performance before the seNet [51] set the final score to 97.75% in 2017.

## 3.5 Supervised deep learning

As explained in section 3.4.8, current deep learning models are trained by predicting some outcome and then update its weights based on the resulting loss. Naturally, a training method that compares the model prediction against the ground truth seems an ideal way to optimize the model. Of course, for many domains, there might not exist one absolute answer. Still, supervised deep learning models have proven tremendously successful and have become the current state of the art in computer vision.

Even though this thesis's focus is unsupervised deep learning, the recent success of supervised deep learning is interesting in many ways. First, in many unsupervised research projects, the goal is to reach similar results as supervised models without using the labeled data. Furthermore, the lines between unsupervised and supervised learning are often blurred [43]. Many neural network structures can, therefore, be utilized in both domains. Thus, a deep neural network that performs well in a supervised setting might produce useful results when adapted to the unsupervised domain.

In image classification, the neural network architectures are typically very similar regardless of the training domain. In this domain, state-of-the-art supervised models are convolutional feedforward networks, consisting of a number of convolutional layers followed by a small number of fully connected layers. The first part of the network can be viewed as a feature extractor that gradually learns the feature representations. This information is then passed to the last layer, which acts as the network classifier. Albeit trained end to end, supervised models thus have very similar structures as their unsupervised counterparts. Therefore, a goal of unsupervised classification models should be to learn similar feature representations as the seemingly superior supervised networks.

There are several existing supervised networks which have provided excellent results in image recognition and object recogntition Because of their good performance it is generally believed that the architectures are especially good for such purposes. Therefore these architectures or parts of them are often adapted to the unsupervised domain and used as backbones for the unsupervised model. The important architectures utilized in this thesis is:

#### 5-CONV

*5-CONV* is a deep neural network consisting of five convolutional layers which extracts information to five succeeding fully connected layers. For plankton classification the network have provided excellent results at the *AILARON* data set with over 95% accu-

racy [99]. The network architecture is depicted in figure 3.14 showing a structure with strong resemblance to the *VGG16*, but being much more shallow. As suggested in the specialization-project [100], a shallow structure might prove beneficial when the data set size is small. The network were thus less susceptible to over-fit providing as good results as much deeper networks.

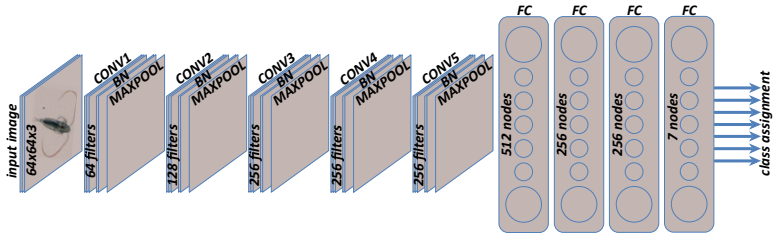



**Figure 3.14:** The 5-CONV architecture. Illustrated in [99, Saad et al. p. 45]

### VGG16

The *VGG16* are a deep network proposed by Simonian and Zisserman in 2015 [104]. As the name suggest, the *VGG16*, seen in figure 3.15, consists of sixteen sequential layers of which thirteen are convolutional and three are fully connected. Compared to previous state of the art models, such as *AlexNet* [63], the network have more layers but a much smaller receptive field with kernels of size $3 \times 3$. The architecture formed the backbone in the authors ILSVRC-14 submission ending second in classification with a top-5 accuracy of 93.2%. As suggested by the authors, the increased network depth compared to previous ILSVRC submissions clearly proved beneficial. The cost of having a larger network with more parameters is the requirement of bigger computational memory. Furthermore, the complex structure can be notoriously hard to train and can easily over-fit on small data sets. Overfitting can, however, be overcome by proper regularization and added batch-normalization [73].

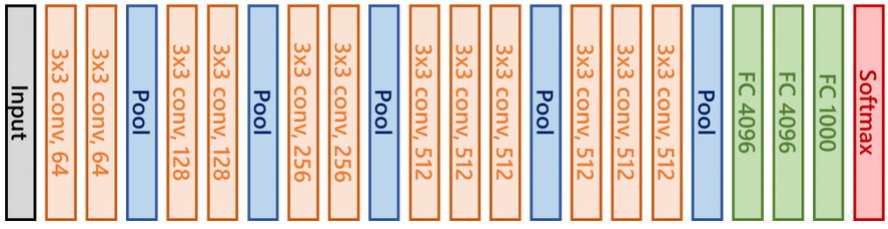
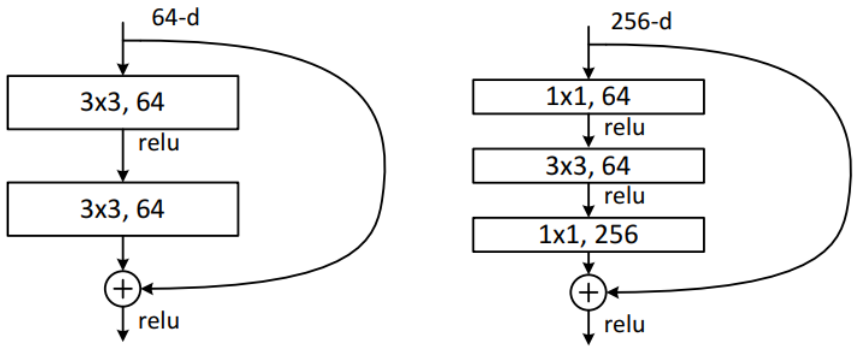



**Figure 3.15:** The VGG16 architecture. Illustrated in [39, Gong, p. 1].

### ResNet

*Residual neural network* (ResNet) [49] is a class of very deep neural network architectures that delivered astounding results in the ILSVRC-15 challenge. The networks consist of several skip connection blocks, known as *residual blocks*, which make it possible for the output signal to shortcut several layers. According to He et al. [49], the ResNet architecture makes it easier to optimize networks of large depth, making it feasible to train networks with up to 152 layers. The *ResNet-34* design draws much inspiration from the VGG network using $3 \times 3$ kernels and keeping the number of filters constant between each downsampling operation. Of course, the biggest difference is that the layers are grouped into residual blocks, depicted in figure 3.16 (a), where the signal can skip over two consecutive convolutional layers. Afterward, this design was improved in the *ResNet-50* design, which replaced the initial residual blocks with a block, depicted in figure 3.16 (b), consisting of three convolutional layers. This model was then expanded into the ResNet-152 architecture, which won the ILSVRC-15 challenge with a top-five accuracy of 96.43%.




**(a)** Residual block of ResNet-34 **(b)** Residual block of ResNet-101

**Figure 3.16:** The figure depicts the two variations of residual blocks used in the ResNet design. **Left image:** Depicting the ResNet-34 design consisting of two convolutional layers and one skip connection. **Right image:** Depicting the Resnet-50 design consisting of three convolutional layers and one skip connection. Illustrated in [49, He et al. p. 6]

## 3.6 Unsupervised deep learning

Unsupervised deep learning is essentially a sub-field of the broader unsupervised domain described in section 3.2.1. In contrast to other unsupervised algorithms, however, deep models are typically more complex and better capable of acquiring knowledge of abstract concepts which is more informative than the sum of its parts [70]. Note that the relatively clear distinctions between learning approaches in machine learning are less distinctive for deep learning models. Goodfellow et al. informally states that; ”unsupervised learning refers to most attempts to extract information from a distribution that do not require human labor to annotate examples” [43, Goodfellow et al. p 146]. Unsupervised models have, therefore, no access to annotated labels but can use other constructed target functions to

optimize the weights similarly as in supervised training. A simple example is using the inputs as targets, which is done by *autoencoders*. Such models can learn important data features by making identical copies of the input.

A primary motivation behind unsupervised training is that learning without labels seems much closer to how knowledge is acquired in real world biological systems. Many thus believe that the next break-through of intelligent computer systems will emerge from this domain [66]. While it is difficult to predict future advancements, it is clear that deep learning models greatly benefit from large amounts of training data [8]. Therefore, an apparent advantage of unsupervised training is the capability to utilize the enormous amount of available unlabeled data [70]. Still, the domain of computer vision is dominated by supervised models except in special cases where the number of labeled samples are minimal [43].

Throughout the last decade, unsupervised deep learning has been used in a wide variety of applications. First, unsupervised learning played a major part in the resurrection of deep learning in early 2000 when pre-training made it possible to train deeper and more complex models [11]. Today, deep supervised models are less dependent on pre-training [111], so the research has shifted more towards applications such as feature extraction, dimensionality reduction, and denoising. This section takes a deep dive into the domain of unsupervised feature extraction, providing necessary background material and presenting the important models utilized in this thesis.

### 3.6.1 Untrained neural network

One of the easiest strategies to obtain an unsupervised feature extraction network is to simply utilize a untrained network. In image feature extraction, CNN networks with randomly initialized filters often works quite good [43]. Untrained CNN feature extraction was introduced by Ulyanov et al. [110] who demonstrated that their network was capable of extracting low-level statistics. The authors attest this capability to the invariant properties of convolutions and when stacking convolutional layers, their ability to capture greater pixel relationships. These findings coincide with the experiments of Chang et al. [16] who claim that randomly initialized filters are excellent edge detectors. Importantly, the choice of network architecture affects the initial performance which ultimately affects how good results an equivalent trained model can achieve. To choose an appropriate network architecture an effective strategy according to Goodfellow et al. [43] is therefore to train only the last layer of several different architectures. The best performing network can then be trained using a more expensive approach.

### 3.6.2 Autoencoder

An *autoencoder* is a neural network architecture that learns to create an identical copy of the input. In theory the network can learn a perfect mapping by simply remembering a one dimensional code for each input image or by remembering all the pixels. The networks are therefore constrained with various architectural constraints and regularizers making the networks unable to achieve perfect reconstructions [38]. An autoencoder can for in-

stance have a bottleneck structure that forces the input to be represented using only few parameters. The network is thus encouraged to describe only the most important aspects of the data.

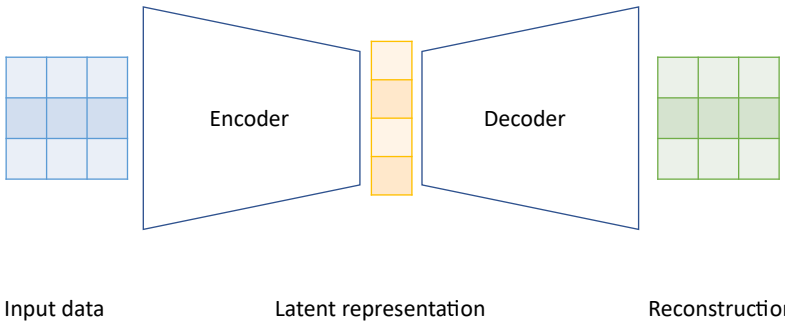



**Figure 3.17:** The figure describes the autoencoder architecture. First, input data is transformed to a latent representation by the encoder part before the input is reconstructed by the decoder part. Illustration inspired from [94, Rocca].

The most common type of autoencoder architecture is depicted in figure 3.17. It can be seen as an extension of the feed-forward neural networks sharing the same architectural building blocks and training strategies [43]. Albeit not necessary, the architecture is usually symmetric [3] consisting of an *encoder* and its inverse *decoder* part. The encoder first maps the input data into a smaller data dimension before the decoder tries to reconstruct the input based on the low dimensional representation. The network can such be optimized by finding the encoder ($e_\theta$) and decoder ($d_\theta$) weights that reduces the synthetic loss function defined in equation 3.8 with respect to the input data (**x**).

$$L(\mathbf{x}, \mathbf{x'})) = \sum_{i=i}^{n} (x_i - d_\theta(e_\theta(x_i))) \tag{3.8}$$

The ultimate objective is, however, not to achieve a perfect reconstruction. Instead the hope is that the network have learned informative representations that is useful for other tasks. Autoencoder pre-training have provided a starting point for training deep supervised models [50]. However, it remains an open question if such implicit training schemes can provide optimal solutions [7]. For instance in classification, Hartono et al. pointed out that labeled training seemed to "influence the internal organization" [47, Hartono, p. 3]. Task specific training might therefore learn a different set of features which is more optimal than the set of features obtained by autoencoder training.

Since the start of the new deep learning wave, autoencoders have been extensively studied and applied to a wide variety of tasks. Such tasks include dimensionality reduction and feature extraction [38]. More recently, the decoder part of a special type of autoencoders has started getting renewed attention as they can learn to make interesting artistic

illustrations [3].

### 3.6.3 Generative adversarial networks

The *generative adversarial network* (GAN) is a relatively new idea which were proposed by Goodfellow et al. [44] in 2014. The proposed idea is essentially based on a zero-sum game between to opposing players where one player tries to trick the other player into believing that a fake data sample is actually real. The model, seen in figure 3.18, consists of two parts: The *generator* network first creates a number of fake images. Together with a selection of real samples, the fake images are then fed into the *discriminator* network which tries to distinguish the real data from the fake. The competition can thus be defined by the minimax loss defined in equation 3.9 given in [44, Goodfellow et al. p. 3]. Here, G is the generator, D the discriminator, x true data and G(z) synthetic data.

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z}(z)[log(1-D(G(z)))] \quad (3.9)$$

Upon reaching equilibrium, the generator should be able to make synthetic data which is so similar to the real data that the discriminator is forced to guess [38]. Reaching the equilibrium can, however, be extremely difficult. First of all the GAN can be quite unstable since the loss is oscillating up and down depending on which network part that is currently performing better. Another problem, called *mode collaps*, can happen if the generator finds it easier to fake a particular class. The model can then be good at faking and recognising a specific class, but forget everything about the other classes [38]. This year, Yazici et al. [120] proved against earlier intuition that GAN networks can overfit by memorizing the data set. The generator can thus win the game by making a copy of existing data which complicates learning further. Training of GAN networks is thus usually a tedious task which requires careful selection and fine tuning of the model architecture and hyper-parameters [43].

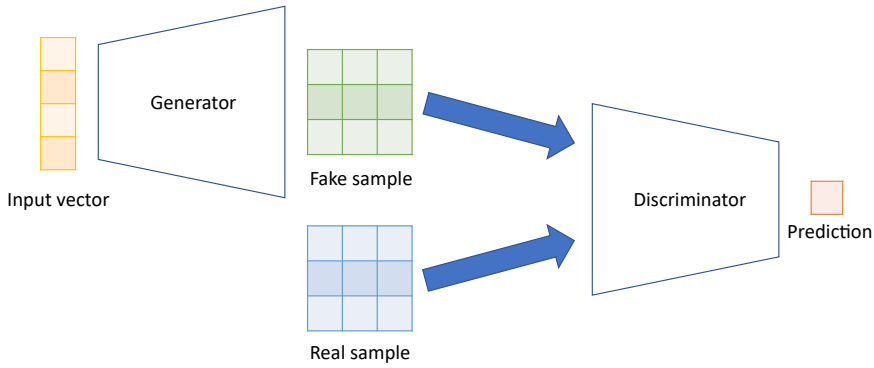



**Figure 3.18:** GAN architecture. The generator network takes a randomized input vector and produces fake images. These images are then tried distinguished from the real samples by the discriminator part.

After training the model successfully, the general use case is to create synthetic samples which can look amazingly realistic. As an example, Karras et al. [57] created human faces of amazing quality while Elgammal et al. [33] showed impressive art design. Figure 3.19 shows synthetic MNIST numbers created by a small GAN network. More important than artistic capabilities, new and unseen data samples can be used to increase a data set or to reduce class imbalance [81]. GANs can therefore be an important tool for increased deep learning performance.

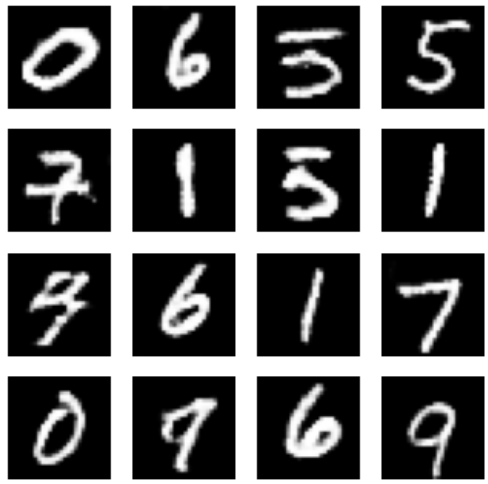


**Figure 3.19:** Synthetic data which are very similar to the original MNIST [67] data, described in section 3.4.9. The image was produced by a deep convolutional GAN using a Wasserstein loss.

### 3.6.4 Unsupervised models for better feature extraction

In the domain of image classification, supervised results are today performing much better, compared to unsupervised or even semi-supervised algorithms. This makes it natural to question whether models such as autoencoders or GANs, which are trained to learn different tasks, actually obtain feature representations that are also optimal for classification purposes. Instead, the supervised end to end training using labeled data might be a more effective method for learning good classification features. Inspired by such observations, Xie et al. [117] proposed in 2016, the *deep embedded clustering* (DEC) algorithm which creates a feature space that is better for linear clustering. First, an autoencoder is trained end to end. Afterwards, the encoder weights are refined based on the high confidence assignments from an added clustering layer. The DEC method and later improvements [46, 59] have sucessfully shown their capability of manipulating the feature space of autoencoders for improved clustering. However, they are only tested using simple CNN

networks on relatively small data sets with few classes.

With a similar purpose, Yang et al. [119] tried to achieve a better representation space by training a CNN network on labels provided by a clustering algorithm. Their *Joint Unsupervised LEarning* (JULE) model is trained step wise. First, the labels are provided from an early stage of an agglomerative clustering model. The network is then trained on these labels for around twenty epochs before repeating the steps. The algorithm proved capable of obtaining a useful feature space on the MNIST dataset. However, it might be challenging to scale the framework to bigger datasets [15]. In 2019, Caron et al. [15] therefore proposed the *Deep Cluster* (DC) which simply train a deep CNN network using labels obtained from a K-means clustering algorithm. The model is trained very similar as the JULE framework by iteratively update the network weights and reassign the clustering labels. Even so, the framework are successfully applied to the ImageNet data. Interestingly the measured NMI between the true labels and the pseudo labels reveals a high amount of misclassifications. Nonetheless, the model is capable of gradually improving its initial feature representation to one which greatly improves the clustering accuracy. Finally, the model is used as feature representation is used to achieve state of the art can be used for feature extraction

**Deep cluster model**

In 2018, Caron et al. [15] at Facebook AI Research proposed a deep learning algorithm that could learn useful feature representations by training on ”pseudo-labels”. As mentioned in the previous section 3.6.4, their work belongs to a relatively new class of unsupervised deep learning models which tries to augment the feature representation by incorporating some form of labeled training. The *Deep Cluster* method depicted in figure 3.20 consists of a conventional CNN network in combination with a K-means clustering algorithm. The model starts by extracting features using the untrained CNN network and then labeling these features using the K-means algorithm. The network is then trained in a supervised fashion for one epoch before the first step is repeated.

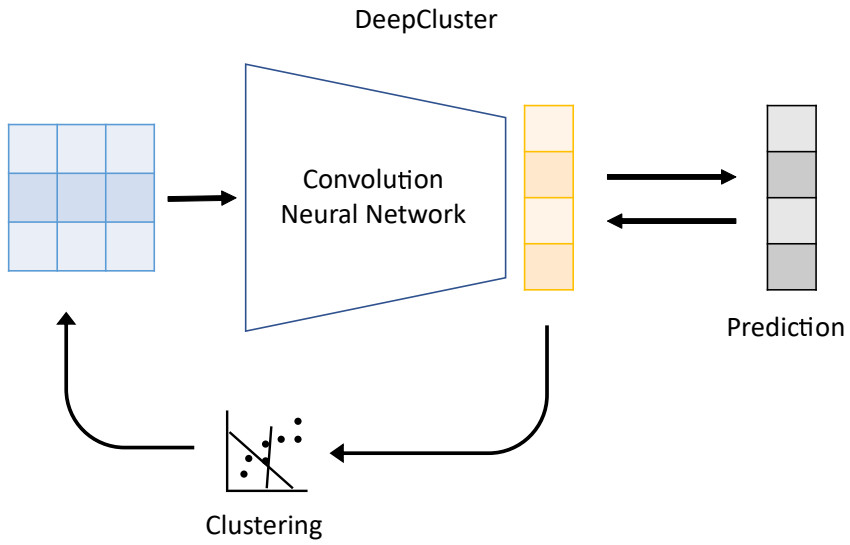



**Figure 3.20:** The *Deep Cluster* is trained using supervised learning. However, prior to each epoch, the training labels are created by a clustering algorithm.

Note that the clustering usually results in a rather bad labeling accuracy meaning the network trains on data containing many labeling errors. Prior to the first training epoch, Caron et al. [15] reported a clustering accuracy of around 12% on the ImageNet data set meaning nearly 90% of the set is misclassified. However, as training progresses the network is gradually improving its feature extractions and thus increasing the clustering accuracy. Albeit seemingly bad clustering results at the start of the training, the clustering accuracy is much better than what a k-means algorithm can achieve on its own. A key factor for the methods success is therefore the CNN networks capability of extracting relevant features, as discussed in section 3.6.1, without any training. This is important since the *DC* method achieves an adequate starting point which make it possible to improve the discriminative power using clustering labels.

### 3.6.5 Unsupervised deep learning classification

The typical framework structure in unsupervised classification consists of a deep feature extraction part and a separate classification layer. The big drawback of this separation is that the classification part can't influence how the feature extraction layer represents the data. In section 3.6.4 several methods were described that partially solves this drawback by introducing techniques that augments the feature representation to be more classification friendly. However, the approaches are still focused on improving the feature extraction part and then add a separate clustering model. Clearly, a preferable solution would be to embed the classifier into the network structure and then train the parts together. Albeit a seemingly reasonable solution, defining a loss function without knowledge of the ground truth labels is, of course, the major challenge of unsupervised learning. Still, there exist

some promising attempts of embedding clustering into the neural network structure. The following sections attempt to give a more profound description of some of these methods.

### DEC

The *Deep Embedded Clustering* (DEC) [117] is a method that facilitates for embedding the clustering part into the deep learning structure. First, an autoencoder is trained to obtain a good initial data representation. Then the decoder part is dropped, and the encoder is merged with the clustering layer. The network is then fine-tuned using Kullback-Leibler divergence $L = KL(Q||P)$ between the clustering assignments $Q$ and an auxiliary target distribution $P$. The cluster assignments are calculated based on the students-t probability distribution in equation 3.10 measuring the similarity between the output feature vector $z$ and the clustering center $\mu$.

$$q_{ij} = \frac{(1 + ||z_i - \mu_j||^2)^{-1}}{\sum_j (1 + ||z_i - \mu_j||^2)^{-1}}, \tag{3.10}$$

Here $q_{ij}$ represents the soft cluster assignment which is the likelihood for the $ith$ output feature vector $z_i$ to belong to the $jth$ cluster center.

The auxiliary target distribution $p$, given in equation 3.11 is then calculated by squaring the soft assignments $q_{ij}$ and then normalizing over frequency per cluster $\sum_i q_{ij}$.

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j (q_{ij}^2 / \sum_i q_{ij})}, \tag{3.11}$$

Thus, the training can be seen as a form of self-training where the network tries to adapt its predictions based on the initial high confidence assignments. In [117], Xie et al. observed that the method gradually improves performance, which shows that the model can manipulate the feature representation to improve classification. However, the model is only tested using relatively simple network architectures and relatively simple data sets such as the MNIST [67]. The method's capacity, when applied on more sophisticated tasks, is therefore uncertain.

### DAC

The *Deep Adaptive Clustering* (DAC) [16] is another technique to embed the clustering process into the deep learning architecture. However, different to the above method, DAC is trained in one single stage, skipping any pretraining stage. The algorithm starts by selecting pair-wise similar images by computing the dot product between the soft label assignments $I_i$ and $I_j$ given in equation 3.12.

$$r_{ij} := \begin{cases} 1, & \text{if } I_i \cdot I_j > u(\lambda) \\ 0, & \text{if } I_i \cdot I_j < l(\lambda) \\ \text{None}, & \text{otherwise}, \end{cases} \tag{3.12}$$

Where $u$ and $l$ represent the upper and lower thresholds for selecting the labeled pair as similar or dissimilar, and $\lambda$ is an adaptive parameter. Furthermore, the assignments $I$ are constrained by the $L_2$ norm so that $||I_i||_2 = 1$. In this way, the vectors tend to be one-hot vectors. Based on these obtained labels, the network is then trained in a supervised fashion. Note that the network is trained in iterations by alternating the labeling and the training process. The DAC algorithm is tested on several research datasets achieving great results on simple datasets and an accuracy of convincingly 52% on the ImageNet [29] dataset.

## 3.7 Visualization tools

As deep neural networks' network structures get more profound and complex, it gets ever more difficult to interpret how the parameter values and their interconnection affect the output prediction. Thus, deep learning neural networks are often referred to as "black boxes," where the network internals are unknown. Deep neural networks' performance are, therefore, often based on metrics such as accuracy in which model predictions are compared against human-annotated labels. However, entirely relying on such metrics is a pitfall since the network may have learned to partition the data in a way that was not intended. For instance, a much-used example in deep learning is the story of the very expensive development of a neural network that was capable of separating American and Russian tanks with 100% certainty [35]. However, when further examinations were made, it turned out that the images were taken in different weather conditions meaning the network had only learned to distinguish sunny and cloudy pictures. Albeit most likely an urban legend, the story gives an important lesson not to rely solely on single metrics. This section presents techniques that offer other ways to learn about the internals of the neural network. These include methods that visualize the network's low dimensional feature representations and practices that reveal the image regions the network finds more interesting.

### 3.7.1 PCA

*Principal component analysis* [115] (PCA) is an effective and well proven method for dimensionality reduction and feature visualization [10]. The goal of PCA is to represent the input data into a low dimensional space using only a selected number of *principal components* while retaining as much information as possible. Figure 3.21 shows an example of a PCA reduction showing that the important information can be kept by a much lower number of dimensions. Given an input $x$ the method calclulates the covariance matrix given in eq. 3.13 and then performs a linear eigendecomposition given in eq. 3.14.

$$S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T \tag{3.13}$$

$$u_1^T S u_1 = \lambda_1 \tag{3.14}$$

Where $\lambda_1$ is an eigenvalue and $u_1$ an eigenvector of the covariance matrix $S$ given in eq. 3.13. The eigenvector $u_1$ with the largest eigenvalue $\lambda_1$ is the first *principal component*. More informally, the first component is the vector that explain the most variance in the data and thus retain most information. Often, most information can be kept by only a handful of such components. Dimension reduction can thus make it much easier to explore the data and speed up machine learning training without reducing the accuracy. However, PCA are limited by its linear nature. The method therefore fails when the data points correlate in non-linear ways such as spherical shapes.

**Figure 3.21:** The figure shows PCA dimensionality reduction on the MNIST [67] data set. **Left image:** Original image having $28 \times 28 = 784$ dimensions. **Middle image:** Same image reduced to 236 dimensions retaining 90% of explained variance. **Right image:** Same image reduced to 141 dimensions retaining 80% of explained variance.

### 3.7.2 T-SNE

T-Distributed Stochastic Neighbor Embedding [79] (t-SNE) is a method for dimensionality reduction and visualization of high dimensional data. As mentioned in the section above, PCA and other linear visualization techniques perform badly on non-linear data. According to Maaten et al. [79], linear techniques focus on separating dissimilar points apart while keeping the low-dimensional representations of similar points close. The t-SNE minimizes the mismatch between the conditional probabilities $p_{ij}$ and $q_{ij}$ defined in equation 3.15 and 3.16 reproduced from [79, Maaten p. 2584].

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq i} exp(-||x_k - x_l||^2/2\sigma^2)} \tag{3.15}$$

$$q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq l} exp(-||y_k - y_l k||^2)} \tag{3.16}$$

Where $p_{i|j}$ converts the high dimensional data points **x** into probabilities using the Euclidean distance between points and their probability mean. Equivalently, $q_{i|j}$ is calculated using the low dimensional points **y**. The t-SNE algorithm now searches for a representation of the points $x$ and $y$ so that the distribution $p$ equals $q$. An example of a two dimensional PCA and t-SNE plot is seen in figure 3.22. The t-SNE model are evidently a very strong tool for visualization of non-linear data and clearly manage to preserve some of the local structure and data groupings. An important note is that the t-SNE algorithm does not retain the distances between points. One must therefore tread carefully when measuring class similarities based on point-wise distance.


**Figure 3.22:** The figure show a two dimensional representation of the MNIST [67] data using the PCA (**left image**) and t-SNE (**right image**) algorithms. Similar colored points belong to the same class as the ground truth labels. Observe that the two dimensional space produced by the t-SNE algorithm seems much better at keeping similar points close and separating dissimilar groups.

### 3.7.3 CAM

*Class activation mapping* CAM is a method that visualizes the image regions that were more important in the CNN classification decision. The method builds on the idea that the last convolutional output feature maps displays high-level concepts and details of certain regions easily visualized in a heatmap. Furthermore, by tying these maps to a specific class, the network reveals information about which image aspects the predictions were based upon. The method was proposed by Zhou et al. [124] and used to describe neural network behavior on the ImageNet [29] data set. The required network architecture follows conventional CNN networks, but the classification layer is constrained to a GAP layer followed by a fully connected softmax layer. As seen in figure 3.23, a heatmap for a specific class can then be calculated by summing the class weights and the output feature

maps from the last convolutional layer.

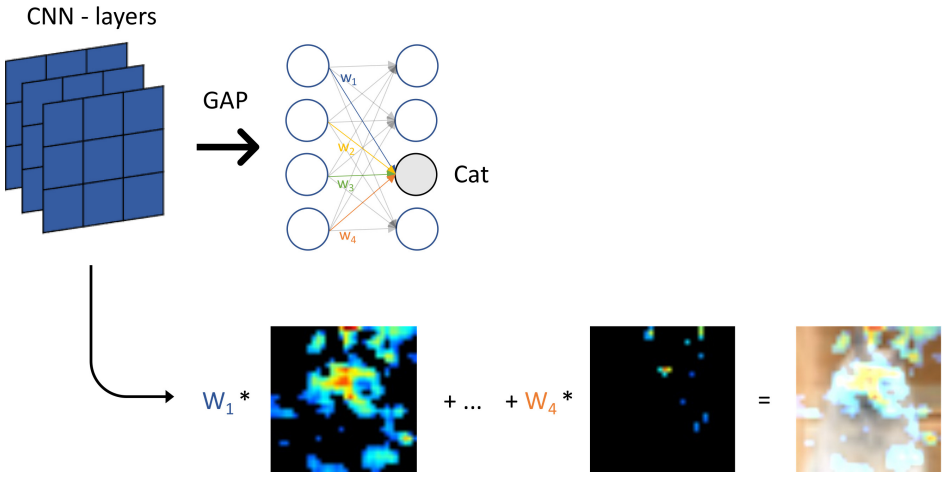



**Figure 3.23:** The figure show the construction of a CAM. The map is constructed by summing the class weights and the output feature maps from the last convolutional layer in the neural network. Note that the method requires a particular network structure to work.

### 3.7.4 Grad-CAM

A drawback with the CAM algorithm, described in the above section 3.7.3, is the requirement of a particular network structure. Many networks does not have this structure meaning the method cannot be used. The recently proposed *Grad-CAM* method [102] is a generalization of the CAM method that works on several more CNN model types. As in the original CAM method the algorithm first calculates the *forward pass* and stores the output feature maps and class predictions. However, in the Grad-CAM method, the weights are calculated using the gradient of class score with respect to the feature map activation's of a convolutional layer. The method thus circumvent the necessity of a strict network structure making it much more accessible.

Chapter 4

# Methodology

This chapter presents the proposed unsupervised classifier framework in this thesis. A detail on the model implementation and experiments is further elaborated hereunder. First, section 4.1 presents the proposed unsupervised classification framework's general setup which consists of three main components. These components are depicted in figure 4.1 and each component is detailed more closely in its own section. First, the data and preprocessing pipeline is detailed in section 4.2. Then, section 4.3 describes the deep learning feature extraction part. Finally, section 4.4 presents the classification and subsequent image labeling.

## 4.1 Unsupervised classification framework

The basic classification framework follows the traditional object recognition approach described in section 3.3. The framework is visualized in figure 4.1 and consists of three main parts:

1. The images are transformed into a consistent format in the pre-processing part.
2. The feature extraction network transforms a high dimensional input image into a low dimensional feature representation.
3. The produced low dimensional representation is fed to a classification model that maps the input to a category.

Observe that the different components in this framework can be adopted in most object recognition algorithms. For instance, the convolutional layers of a supervised CNN network are used to extract useful features, which are then fed to the classification part represented by the last fully connected layer. However, the main purpose of this framework

is the adoption of the integrated approaches in the unsupervised learning domains. Feature extraction is done using deep learning while the classification is performed using an additional layer or a separate machine learning clustering algorithm. Section 4.3 presents the different feature extraction alternatives and section 4.4 describes the clustering models used for classification.

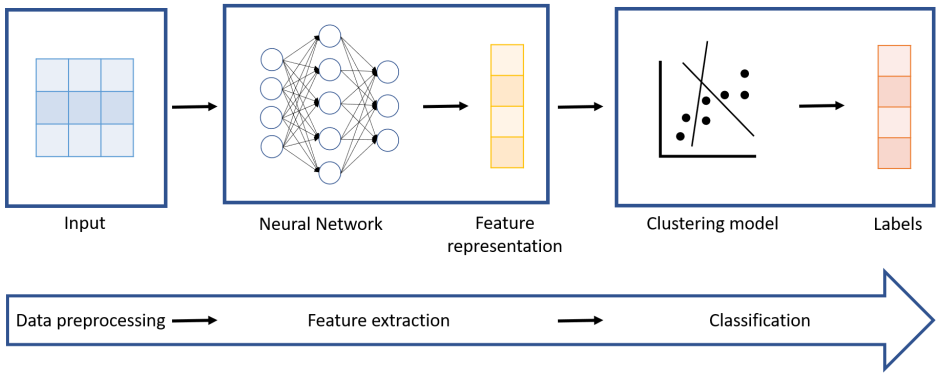



**Figure 4.1:** The proposed unsupervised classification framework. Input images are mapped to a low-dimensional space by the feature extraction network and subsequently classified by the clustering model.

## 4.2 Data pre-processing

This section explains the pre-processing and image augmentation steps over the Kaggle and AILARON data set, respectively. Based on the theory in section 3.4.7, the image pre-processing was kept simple so that the deep learning models can learn the best way to pre-process the data on their own. The same conclusion was drawn in the specialization-project were some novel attempts of enhancing image quality using hand-designed filters were tested without noticeable improvements. Some pre-processing steps are, however, performed in this thesis. Due to architectural constraints, the images need to have similar dimensions. Image resizing is therefore applied. Furthermore, deep neural networks usually work better on small input data. It was thus decided to standardize the pixel values into a small domain. Data augmentation is applied at training time to avoid overfitting and increase model generalizability. Since the plankton objects are depicted in all possible orientations, this allowed for a quite aggressive augmentation in regards to rotation and translation.

The data pre-processing over the Kaggle and AILARON data mainly follows the specialization-project steps to ensure a valid comparison. Furthermore, the data pre-processing is kept simple in accordance to the general requirement discussed above and current best practise in the deep learning domain. The steps are as follows: 1. The images are resized to $64 \times 64$ pixels which was the image format used by Saad et al. [99] in the supervised

classification domain. 2. The data is centralized around zero by subtracting the mean. 3. The data is normalized to the [-1,1] domain. Lastly, the models are constructed for the purpose of classifying RGB images (which are the format of the AILARON data set). The one-dimensional images are therefore copied three times to fit the three-channel input while the AILARON dimensions are kept.

Unless stated otherwise, the algorithm training is conducted using data augmentation to increase generalization and avoid model overfitting. An example of these random augmentations applied on a copepod image can be seen in figure 4.2. During training, the following functions were applied at random to each batch:

- Rotations: Random rotations between zero and ninety degrees.
- Shifts: Random horizontal and vertical shifts which moves the object at most 10% in either direction.
- Zoom: Increasing or decreasing the object size by adding or interpolating pixels respectively. The range is set to 20% in either direction.
- Flip: Random horizontal or vertical flips of the image.

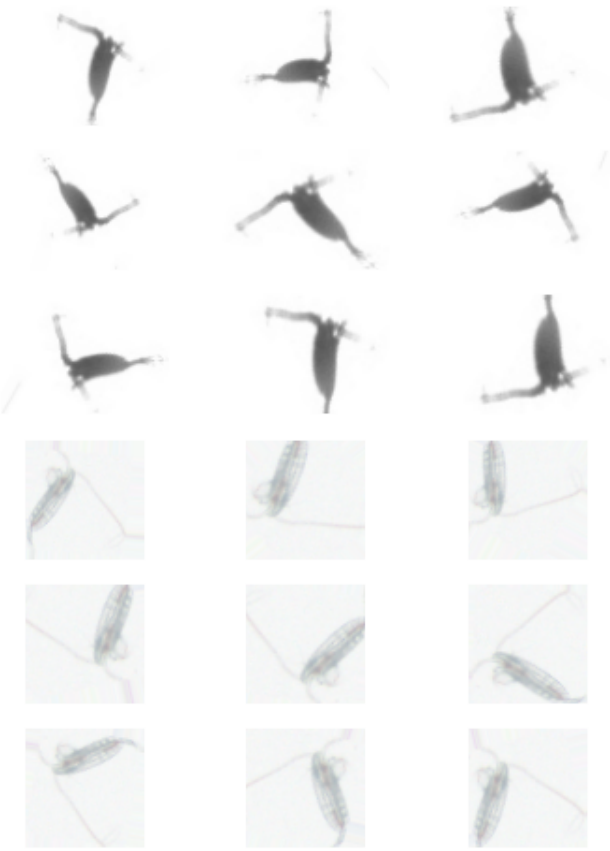


**Figure 4.2:** Example of data augmentations applied to a copepod image in the Kaggle and the AILARON data set.

## 4.3 Feature extraction network

This section explains the second step of the framework explained in section 4.1. Conventional machine-learning clustering methods usually have poor performance on high dimensional data [82]. Therefore, deep learning feature extraction methods are used to map the data into a feature representation space where existing clustering methods can more easily classify the data. Today, a wide variety of methods seek to learn a clustering-friendly data representation using neural networks. First, the traditional machine learning approaches for feature extraction are presented in section 4.3.1. Then, three different unsupervised deep learning methods are presented next for improved feature extraction. Section 4.3.2 presents an autoencoder network. Then, in section 4.3.3 a GAN network is described. Finally, in section 4.3.4 the DeepCluster algorithm is presented.

### 4.3.1 Traditional unsupervised feature extraction models

Prior to deep learning-based feature extraction, image feature extraction was typically performed by less complex machine learning algorithms. Such models might prove beneficial for the AILARON task as they are much faster and easier to train and require less memory compared to the more complex neural network models. In this work, the SIFT [74] and the SURF [6] algorithms, described in section 3.3.1, are utilized. These models are among the most successful machine learning classifiers, albeit usually surpassed by deep learning networks in recent applications.

### 4.3.2 Autoencoder model

Autoencoders have, for a long time, been considered one of the most promising unsupervised deep learning models. Related to this work, there have been several attempts at utilizing them for feature extraction, and subsequent classification [18, 92]. However, few great successes have been achieved, and the focus has gradually shifted to purely supervised models. As discussed in section 2.3.3, autoencoders were revisited in the specialization-project, and the feature extraction capabilities of several architectures was notoriously tested. The deep convolutional models proved capable at learning data representations where similar samples were grouped and separated from dissimilar ones. However, a consistent problem was the separation of similar species of different rotation.

This thesis copies the same general autoencoder approach as done in the specialization-project. However, the proposed architecture is changed to create a more robust model to similar species with different rotations. Such an autoencoder idea was explored in the work of Kuzminykh et al. [65] who created an autoencoder that is invariant to thirty degrees rotation. However, their work is not publicly available, which makes it challenging to adapt their model. Furthermore, their work focuses on extracting invariant features using a new extraction technique. Their network is, therefore rather shallow and with a very different structure. Other relevant works include the supervised *Group-equivariant CNN* network created by Veeling et al. [112]. This network becomes rotation invariant to ninety degrees rotations by utilizing group convolution layers. The proposed method is depicted in figure 4.3 where the model is trained end to end by learning to reconstruct planktonic images. At test time, the decoder part is removed, and the latent representation learned by the encoder part is fed to a separate clustering algorithm.

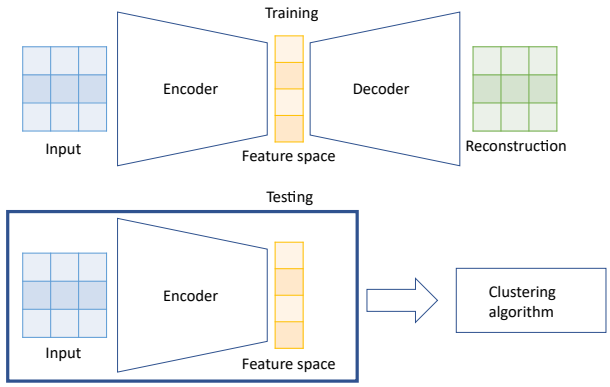



**Figure 4.3:** The autoencoder is trained by copying input to output. At test time, the decoder part is removed and the encoder is used to extract features to a separate classification algorithm.

The architecture is partly symmetric, using an encoder based on the 5-CONV architecture and an inverted decoder part. Different from previous networks, the convolutional and transposed convolutional layers of the *encoder* are replaced by group-convolutions followed by group-batch normalization. The full architecture is depicted in table 4.1 where G-Conv layer refer to the collective group consisting of group-convolutions, group-batch normalization and leaky-ReLU activations. The network takes an input image of dimensions height, width and channels $H \times W \times C = 64 \times 64 \times 3$. Information is then extracted in four stages where the number of output feature maps are doubled and the feature map size reduced by four. Similarly, the decoder part upscales the input and reduces the dimensions using four inverse stages. The final output is a $64 \times 64 \times 3$ reconstruction transformed by a *tanh* activation function.

The architecture is constrained into an hourglass shape where input dimensions are gradually reduced to prevent model overfitting and force learning of important features. The middle layer, referred to as the "bottleneck layer," consists of two layers. First, the feature maps are grouped into 512 rotation invariant feature maps using a group-pool layer. These maps are then transformed into a $1 \times 512$ vector by a GAP layer. A GAP layer was chosen as it worked well as a regularization term in the specialization-project and has gradually started replacing traditional dense layers implementations in recent state of the art models [91]. The loss function is defined similar to the specialization-project using MSE which is a well-attested choice to measure reconstruction loss [22]. Based on the loss function explained in equation 3.8 the loss thus becomes:

$$MSE = \frac{1}{n}\sum_{i=i}^{n}(x_i - d_\theta(e_\theta(x_i)))^2 \tag{4.1}$$

Where $d_\theta(e_\theta(x_i))$ is the encoder-decoder reconstruction of the input $x$. Finally, the optimizer was chosen based on trial and error resulting in the *Adam* [58] algorithm with

standard settings.

**Table 4.1:** Overview of the rotation invariant autoencoder architecture

| Layer | Kernel | Channels | Output dimension |
|---|---|---|---|
| Input | - | 3 | 64x64 |
| G-Conv | 3x3 | 256 | 64x64 |
| Max pool | 2x2 | 256 | 32x32 |
| G-Conv | 3x3 | 512 | 32x32 |
| Max pool | 2x2 | 512 | 16x16 |
| G-Conv | 3x3 | 1024 | 16x16 |
| Max pool | 2x2 | 1024 | 8x8 |
| G-Conv | 3x3 | 2048 | 8x8 |
| Max pool | 2x2 | 2048 | 4x4 |
| G-Conv | 3x3 | 2048 | 4x4 |
| G-pooling | - | 512 | 4x4 |
| GAP | - | - | 512 |
| Dense | - | - | 32768 |
| Reshape | - | 2048 | 4x4 |
| Trans-Conv | 3x3 | 512 | 8x8 |
| Trans-Conv | 3x3 | 256 | 16x16 |
| Trans-Conv | 3x3 | 128 | 64x64 |
| Trans-Conv | 3x3 | 64 | 64x64 |
| Trans-Conv | 3x3 | 3 | 64x64 |
| Number of parameters: | 45 801 795 | | |
| Loss function: | MSE | | |
| Optimizer: | Adam | | |

### 4.3.3 GAN

The recently proposed GAN networks offer another way to learn the underlying data representations without ground truth labels. These networks have gained much attention, especially due to the excellent new images created by the generator part. However, Observe that the learning scheme also results in a discriminator part which trained to distinguish fake from real samples. To differentiate these samples, the discriminator thus needs to learn the data's main features. As visualized in figure 4.4, an unsupervised classification framework can be obtained by training the GAN network and then use the discriminator for feature extraction. In their work to improve GAN training such attempt was made by Radford et al. [91]. However, they proved the feature extraction capabilities by training a supervised classifier on top of the extraction network. More importantly, their work have partially solved many of the problems making GAN networks notoriously hard to train. Goodfellow states that "most GANs today are at least loosely based on the DCGAN architecture" [42, Goodfellow p. 27]. As mentioned in section 2.3.2, Wang et al. [114] partially

solved the problem of class imbalance over the WHOI [87] plankton data set by using an architecture very similar to the DCGAN. Their findings, albeit focused on the generator part, suggest that GAN networks can be adapted to the planktonic domain.

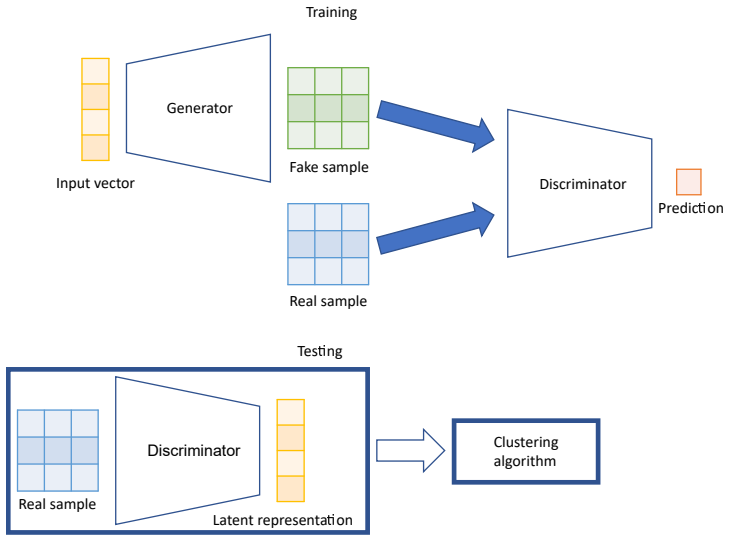



**Figure 4.4:** The GAN is trained using two competing networks. The generator creates fake images which is then distinguished from real images by the discriminator. At test time, the generator part is removed and the discriminator is used to extract features to a separate classification algorithm.

The full GAN architecture is presented in table 4.2 basing the implementation on the TensorFlow exemplary model [1] and using general guidelines from [38, 91]. The generator part, first inputs a randomized vector of size $1 \times 100$. Based on this input, the network produces 512 output feature maps of size $4 \times 4$ using a fully connected layer with leaky-ReLU activation. These maps are then transformed into a $64 \times 64$ image over four stages using transposed convolutions followed by leaky-ReLU activation and batch-normalization. Lastly, the output is transformed into a $64 \times 64 \times 3$ image using a convolutional layer with *tanh* activation function.

The discriminator model mainly follows the overall design of conventional CNN networks. The input image is either real or generated with dimensions height, width and channels $H \times W \times C = 64 \times 64 \times 3$. Information is then extracted in four stages using convolutional layers followed by leaky-ReLU activation and batch-normalization. Furthermore, dropout regularization was applied to reduce the convergence speed. After the convolutional part, the feature maps are then reduced into a $1 \times 8192$ vector using a flattening operation. The flattening is then followed by two fully connected layers, where the last layer predicts whether the image is true or false. Observe that the final fully connected layer has no activation. According to the guidelines, a *sigmoid* activation is recommended, but such activation seemed to make the discriminator too powerful, making the generator incapable of learning. The loss is measured using *Cross Entropy* based on the discriminator image

predictions, The optimizer follows the parameter assignment from Radford et al. [91] using the *Adam* [58] optimizer with learning rate $\alpha = 0.0001$ and $\beta\_1 = 0.5$.

**Table 4.2:** Overview of the GAN model architecture

Generator

| Layer | Kernel | Channels | Output dimension |
|---|---|---|---|
| Input | - | - | 100 |
| Dense | - | - | 8192 |
| Reshape | - | 512 | $4 \times 4$ |
| Trans-Conv | $5 \times 5$ | 512 | $8 \times 8$ |
| Trans-Conv | $5 \times 5$ | 256 | $16 \times 16$ |
| Trans-Conv | $5 \times 5$ | 128 | $32 \times 32$ |
| Trans-Conv | $5 \times 5$ | 64 | $64 \times 64$ |
| Trans-Conv | $5 \times 5$ | 3 | $64 \times 64$ |

Discriminator

| Layer | Kernel | Channels | Output dimension |
|---|---|---|---|
| Input | - | 3 | $64 \times 64$ |
| Conv | $5 \times 5$ | 64 | $32 \times 32$ |
| Conv | $5 \times 5$ | 128 | $16 \times 16$ |
| Conv | $5 \times 5$ | 256 | $8 \times 8$ |
| Conv | $5 \times 5$ | 512 | $4 \times 4$ |
| Flatten | - | - | 8192 |
| Dense | - | - | 256 |
| Dense | - | - | 1 |
| Number of parameters: | 18 122 817 | | |
| Loss function: | Cross Entropy | | |
| Optimizer: | Adam | | |

### 4.3.4 Deep cluster model

The theoretical observations in section 3.6.4 and the DCEC experiments conducted in the specialization-project shows the importance of finding models that can learn more optimal feature representations for classification. One of the current state of the art models is the *Deep Cluster* [15], described in section 3.6.4, that have achieved good performance on the ImageNet [29] data set. Encouraged by these results, the method was adapted to the unsupervised research of this thesis. The core of the Deep Cluster code[1] remains unchanged in this thesis model adaption. However, the programming platform is changed

[1]Code available at `https://github.com/facebookresearch/deepcluster`.

from *PyTorch* to *TensorFlow* to fit into the already created framework and make the results better comparable to the other extraction networks.

In the original DeepCluster work [15], Caron et al. used the AlexNet [63] and the VGG16 [104] models as backbone networks. However, the choice of the backbone network is very flexible, making it possible to try other network architectures. The framework was thus tested using three different CNN architectures adapted from the supervised domain: 1. 5-CONV [99], 2. VGG16 [104] and 3. ResNet [49]. The first network was chosen due to its good performance on the AILARON data, while the other two have achieved excellent results on other computer vision tasks.

Training of the Deep Cluster follows the procedure listed in Algorithm 1. First, from lines 2-4, the framework is initialized. The input is a set images having input of dimensions height, width and channels $H \times W \times C = 64 \times 64 \times 3$. The model is one of the three backbone networks described above, and the clustering algorithm is K-means. Second, from lines 6-9, the pseudo-labels are created by clustering the backbone network's feature extractions. Then, from lines 11-14, the backbone network is trained on the image/pseudo-label data set for one epoch before the loop repeats. The loss function was defined as categorical cross-entropy and the optimizer was stochastic gradient descent with learning rate $lr = 0.005$ and momentum $m = 0.05$.

**Algorithm 1** Deep Cluster

```
1:  procedure
2:      Input ← Images
3:      Model ← CNN network
4:      Cluster algorithm ← K-means
5:      for Epoch 1 to 100 do
6:          if Get pseudo-labels then
7:              CNN ← Remove classification layer
8:              Features ← CNN(Input)
9:              Labels ← Kmeans(Features)
10:         if Train then
11:             CNN ← Add new classification layer
12:             dataset ← (Images, Labels)
13:             for data in dataset do
14:                 Train CNN(data)
        return Model
```

### Number of clusters

An important hyperparameter in the Deep Cluster framework is the number of clusters $k$. A rational number would be $k = 121$, same as the actual number of classes in the Kaggle data set. In the ideal case, the clustering should thus result in a data set containing pseudo-labels similar to the human annotated ground truth labels. However, Caron et al. found that some amount of over-segmentation was beneficial and set their $k = 10000$ on the ImageNet [29] data set. Like Caron et al. in [15] different numbers of $k$ were tested

in this thesis, and the resulting models were tested and validated. The number of clusters and hence the number of training categories was chosen to be be $k = 200$ on the Kaggle data and $k = 20$ for the AILARON data.

## 4.4 Classification model

The previous section 4.3 discussed the different methods used for feature extraction. The second part of the unsupervised classification framework, described in section 4.1, is the classification model. In most previous unsupervised works, the classification part is separated from the extraction network [82]. Data is first transformed into a low dimensional representation and then clustered by a traditional clustering algorithm. However, a very recent branch of deep unsupervised learning focuses on embedding the clustering part into the deep learning model to obtain more clustering-friendly representations and subsequently increased model performance [82]. This section presents classification components from both domains. First, several classical clustering algorithms that are separated from the extraction network are presented in section 4.4.1. Then, the DEC approach that aim to embed the classification into the feature extraction model are presented in section 4.4.2.

### 4.4.1 Machine learning cluster algorithms

This subsection describes the clustering algorithms that are utilized using the traditional unsupervised approach. First features are extracted using a deep learning model, and then the algorithms described underneath are used for classification.

#### K-Means

K-means [80] is a partition-based clustering algorithm aiming to minimize the Euclidean distance from each sample point to one of the $k$ cluster centers. These $k$ clusters must be defined before the algorithm is run. The algorithm is well attested, much used, and quite fast compared to many other clustering algorithms. However, the method usually performs poorly on non-linear data, elongated clusters, or on irregular shapes.

#### Spectral Clustering

*Spectral Clustering* [113] (SC) is a group of clustering methods which is more robust, as opposed to linear methods, when handling non-linear features. The algorithm can be seen as a combination of centroid- and connectivity-based clustering techniques. The algorithm first uses a connectivity-based algorithm to construct a similarity graph where each edge represents a connection between two similar points. Based on this graph, the algorithm creates a low dimensional embedding so that connected points end up closer and non-connected points end up relatively far away. Finally, the points now represented in a low dimensional space are clustered by a centroid-based clustering algorithm (K-means is implemented in Scikit-Learn [89]).

#### Gaussian-mixture

The *Gaussian-mixture* model is a clustering algorithm that assumes the data to come from different Gaussian distributions. Clusters, typically ellipsoidal groups of points, are formed by points that are likely to have been produced by the same distribution. The algorithm starts by randomly assigning initial components (cluster centers), which are then iteratively improved using Expectation-Maximization (EM). First, the probability of each assignment being generated by each component is calculated. Then, the mean and variance of each component are tweaked to better accommodate these assignments.

#### BIRCH

*Balanced Iterative Reducing and Clustering using Hierarchies* (BIRCH) [122] is an algorithm designed to work efficiently on large data sets. The core of BIRCH clustering is to build a tree structure, called *Clustering Feature Tree*, which holds necessary information about the data, thus avoiding to store all data points. This structure can then be used to quickly assign new points to a desired class.

### 4.4.2 Deep learning embedded clustering

Current state of the art models in computer vision are supervised deep neural networks trained end to end. Therefore, it would seem beneficial to also train the unsupervised deep learning framework end to end so that the weights of the feature extraction layers can adapt to the clustering layer. Of course, the problem with this strategy, as opposed to supervised training, is that unsupervised models do not know the ground truth labels, meaning one must make various assumptions on the data structure. The following method incorporates a clustering layer into the unsupervised deep learning structure and makes it possible to train end to end.

#### DEC

Deep Embedded Clustering (DEC) [117], explained in section 3.6.4, is a framework that makes it possible to incorporate the clustering method into the neural network structure. The network can thus learn to extract features that are better adapted to clustering. In the original work, the proposed architecture is an autoencoder that first learns a feature representation by learning to reconstruct images. The pre-trained encoder part is then connected to the clustering layer and fine-tuned using the KL-divergence between the soft assignments and an auxiliary target distribution [82]. The DEC implementation in this thesis follows the original work except for the feature extraction networks being pre-trained in various ways.

Chapter 5

# Experiments and implementation details

This chapter provides information about the code framework and necessary implementation details for building the unsupervised framework described in chapter 4. The chapter then presents the details and purpose of the three experiments conducted. The structure of the chapter is the following. First, section 5.1 provides a short description of the important software platforms used in this work. Section 5.2 then details the software packages used to build the deep learning algorithms. Next, in section 5.3 the computer specifications are listed. Lastly, section 5.4 presents the experiments explaining the approach and the objective of each task.

## 5.1 Software platforms and important code libraries

The code implementation was done using several different software packages. Some are directly related to the deep learning and computer vision task while others serve more basic functionality such as image pre-processing, and image visualization. An overview of the relevant software can be found in table 5.1. The main code implementation was done using python 3.6, and Tensorflow [1] using the Keras [17] high-level API. The code implementation can be found in the following Github repository, `https://github.com/AILARON/Unsupervised-classification`, where further dependencies are noted in the install.txt file.

**Table 5.1:** Description of important code libraries and software platforms

| | Description |
|---|---|
| Tensorflow | *Tensorflow* [1] is an open source platform specifically designed for deep learning and computationally heavy algorithms. It was developed by the GoogleBrain team and is frequently utilized in Google's own services. Geron et al. claims in [38, Geron et al. p 376] that TensorFlow is today "the most popular Deep Learning library (in terms of citations in papers, adoption in companies, stars on GitHub, etc)". Furthermore, in the latest release TensorFlow adopted Keras as its official high-level API and greatly improved the documentation which have made the platform much easier to work with [38]. |
| Keras | *Keras* [17] is a high level API that provides building blocks for building and training deep learning neural networks. Albeit a simplistic and often easy to use interface, it is very flexible and can be used to build a wide variety of network architectures [38]. |
| Scikit-learn | *Scikit-learn* (Sklearn) [89] is an open-source machine learning library which is well integrated with the Python programming language. Scikit-learn provides both classic and state of the art implementations including supervised, clustering and regression algorithms. |
| OpenCV | *Open Source Computer Vision Library* (OpenCV) [12] is an open-source machine learning and computer vision library. It contains more than two thousand optimized algorithms as well as readily available functions for image pre-processing and model evaluation. |
| CUDA | To train deep neural networks within a reasonably amount of time, it is necessary to make the computations using a GPU unit. *CUDA* [84] is a Nvidia developed programming model which enables the parallelizable code parts to run in parallel over the GPU cores. |
| Faiss | *Faiss* [54] is an open-source library for efficient similarity search and clustering. It is well integrated with the Python programming language and many of the algorithms have GPU support. Furthermore, many of the clustering methods can be utilized even if the data does not fit in RAM. |
| GrouPy and keras_gcnn | The *GrouPy* [19] library is a python library that implements the necessary modules to create group convolutional neural networks. This library is further extended into a Keras library called *keras_gcnn* [112]. |

## 5.2 Deep learning model requirements

With the interest and increased focus on deep learning for industrial applications, the code libraries and software development platforms are rapidly changed and improved. As older code versions are unlikely to be compatible with the new version, this results in the need to utilize different models using different versions of the same software platforms. This section provides an overview of the third-party libraries and versions necessary to run the deep learning models.

#### Autoencoder

The rotation invariant autoencoder described in section 4.3.2 was implemented using an older version of TensorFlow due to compatibility issues with the the *Groupy* and *Keras_gcnn* third-party libraries. The code is tested and run successfully with the packages listed below.

1. Tensorflow 1.15.0
2. Keras 2.3.1
3. groupy[1]
4. keras_gcnn[2]

#### GAN

The GAN model described in section 4.3.3 was implemented using TensorFlow and Keras. The main model architecture is built using the Keras high-level API whereas the training loop is constructed using TensorFlow functions. The code is tested and run successfully with the packages listed below.

1. Tensorflow 2.0.0
2. Keras 2.3.1

#### Deep Cluster

The DC model described in section 4.3.4 is adapted from PyTorch code using TensorFlow and Keras. The backbone network architectures and the network training are mostly constructed using the Keras high-level API embedded in TensorFlow. Furthermore, the models are trained on labels constructed by a clustering algorithm. In line with the implementation in [15] the clustering is run using the Faiss library. The code is tested and run successfully with the packages listed below.

1. Tensorflow 2.0.0

[1] Code available at https://github.com/tscohen/GrouPy
[2] Code available at https://github.com/basveeling/keras-gcnn

2. Keras 2.3.1
3. Faiss[3]

## 5.3 Computer specifications

The training and testing are performed on a lab computer with the specifications listed in table 5.2. All experiments were conducted on a single GPU with 11 GB of internal memory and over four thousand cores.

**Table 5.2:** Description of the computer specifications

| | |
|---|---|
| OS: | Ubuntu 18.04.2 LTS |
| CPU: | Intel LGA1151 i9 - 9900K |
| GPU: | 2x ASUS RTX2080Ti Turbo |
| RAM: | 64 GB |
| SSD: | Crucial MX500 2TB |
| HDD: | Seagate Skyhawk 6TB |

## 5.4 Experiments

This section provides details about the experiments conducted in this thesis. The purpose is to provide a better overview of the work and experimental setup and to explain the intention behind each experiment.

### 5.4.1 Experiment 1 - Unsupervised feature extraction algorithm

A sub-goal of this thesis is to discover a self-learned deep learning network capable of extracting useful features for subsequent classification. Experiment 1 is thus dedicated to validate and assess the different feature extraction models explained in section 4.3. Note that the first attempts at obtaining such a model were conducted in the specialization-project. This experiment follows the same approach and test specifications to ensure a valid comparison to these networks. The training is thus performed on the Kaggle-DB2 while extraction capabilities are validated over the Kaggle-DB1.

The first part of the experiment sets a baseline by training several state of the art supervised CNN and validating their feature extraction capabilities. This was done to make a better comparison to the existing supervised work in the domain and to get an indication of what results to expect from the unsupervised models. Then, the extraction capabilities of the SIFT and the SURF unsupervised machine learning models are validated. Finally, the three feature extraction networks described in section 4.3 are trained in their own respective way before being thoroughly tested and validated. The following validation methods and performance metrics are used to assess the model performance:

[3]Code available at https://github.com/facebookresearch/faiss

- Assessment of the training and validation loss to reveal how the model generalizes to the plankton data.

- Model accuracy using k-means and SC clustering. The cluster labels are assigned using the Hungarian method and compared against the ground truth labels. The test assumes that in accordance with Kuzminykh et al. in [65], a higher performance implies that the deep network has learned a better representation of the data.

- Visualization of representational space using t-SNE and PCA. How the data is grou -ped in the 2D plot is assumed to reveal information about how the deep network represents the data.

- Visualization of the network class activation maps. The goal is to reveal some of the network thought processes or discover networks that do not behave in the intended way.

### 5.4.2 Experiment 2 - Choosing the appropriate building blocks of the framework

In the first experiment, several feature extraction models were tested and validated following the specialization-project approach. Experiment 2 aims to further assess the suitability of the best feature extraction models from the previous experiment and fit these networks with an appropriate clustering algorithm to finalize the proposed framework.

The experiment is mainly conducted over the Kaggle-DB1 data set using models trained on the Kaggle-DB2 data set. A problem with this setup is that the larger data set includes all samples from the smaller validation set so that the validation data is seen during the training. Note that this approach is following recent work in the unsupervised domain [16, 46, 117, 119]. However, to prove that the feature extraction models are unbiased to the validation data, the first part of this experiment is dedicated to test the model performance on a small subset consisting of 100 unseen samples from the Kaggle-DB1 data set.

Then, to gain more insight into how the feature extractions and clustering assignments are connected, the feature representations are visualized using t-SNE and compared to the corresponding confusion matrix. Lastly, the clustering algorithms, described in section 4.4, and their performance is more thoroughly investigated with the hope of finding a more suitable model for the AILARON tasks. The additional validation methods and performance metrics used in this experiment are the following:

- Classification performance between the cluster assignments and the ground truth labels. In their unsupervised survey, Min et al. [82], in accordance with other previous work [16, 46, 117], recommends the following metrics: 1. Accuracy using the Hungarian algorithm [64] to map ground truths and cluster assignments. 2. Normalized Mutual Info Score between ground truth and cluster assignments.

- Evaluate the clustering assignments by comparing the deep learning feature representations, using the t-SNE algorithm, and the classification matrix. The goal is

to reveal plankton species and their characteristics, which are challenging for the framework to label correctly.

- Classification speed measuring the prediction time over an increasing number of samples. The goal is to determine which classification algorithms that are feasible for real-time classification.

### 5.4.3 Experiment 3 - Evaluation on the AILARON data

This thesis aims to provide an unsupervised framework that can be used in the AILARON project to detect and classify planktonic species. However, as discussed in section 2.4.1, the current data set version has many drawbacks, which made it more appropriate to evaluate the proposed framework and its parts over different plankton data. Still, it is interesting to measure the feasibility of applying unsupervised deep learning into the AILARON domain. Experiment 3 is, in this way, a continuation of the past two experiments conducting some of the same tests and performance evaluations over the AILARON data.

Chapter 6

# Results and reflections

This chapter presents the results of the experiments explained in section 5.4. The next three sections, 6.1-6.3 presents the results where each section contains one experiment followed by a summary and reflection part discussing the findings.

## 6.1 Experiment 1 - Unsupervised feature extraction algorithm

This section presents the results of experiment 1, which is described in section 5.4.1. The results are presented in the following order. First, the baseline results achieved in the specialization-project and by supervised deep learning are presented. These methods set the baseline for what is realistic accomplishments using unsupervised learning. The subsequent sections provide results of the unsupervised models. Section 6.1.2 presents results obtained using traditional machine learning methods. Then section 6.1.3 presents the results after training the autoencoder model. Section 6.1.4 describes the results of the GAN model while section 6.1.5 show the results of the DeepCluster model. The section ends with a brief summary and reflection of the overall results.

### 6.1.1 Baseline methods

The baseline classification results are depicted in table 6.1 showing the baseline results achieved using feature extraction followed by K-means and SC clustering. For reference, the results obtained without using feature extraction and the results using the "Auto-GAP" model are taken from the specialization-project report [100, Salvesen p. 40]. In the specialization-project clustering was improved using unsupervised feature extraction by over 15% showing the benefits of using deep learning feature extraction. Similarly, the most successful image classification models, and hence state of the art feature extraction

models, are currently supervised CNN networks. Replacing the frameworks feature extraction part with a supervised deep learning algorithm is thus useful to form an upper benchmark. By switching to a supervised deep learning feature extraction model, denoted as ”5-CONV”, ”VGG16,” and ”ResNet,” the performance is further improved, reaching a clustering accuracy of 96% in the best cases. The great improvement obtained using supervised deep learning indicate their superiority at learning good feature representations of the data.

**Table 6.1:** Baseline classification results on Kaggle-DB1

| Feature extractor | K-means | SC |
|---|---|---|
| - | 0.44 | 0.61 |
| Auto-GAP | 0.62 | 0.76 |
| 5-CONV | 0.92 | 0.95 |
| VGG16 | 0.91 | **0.96** |
| ResNet101 | 0.95 | **0.96** |

Table 6.2 shows the obtained training results after testing the proposed backbone CNN networks using supervised training. The networks were trained over 100 epochs using the same hyperparameters and early stopping regularization. The *5-Conv* reached the stopping criterion after about 60 epochs, whereas the deeper network trained for about 80 epochs. All models achieved good performance over the unseen validation set, but the results indicate an increased performance when a deeper architecture is used. However, the very deep *ResNet* architecture performs no better than the *VGG-16* network. These findings accord well with the clustering results from table 6.1 using the networks as feature extractors. Observe that the deepest networks provide features that make the clustering over 96% accurate.

**Table 6.2:** Supervised neural network training performance on Kaggle-DB2

| Model | Training loss | Training ACC | Validation loss | Validation ACC |
|---|---|---|---|---|
| 5-Conv | 0.70 | 0.76 | 1.14 | 0.67 |
| VGG16 | 0.5827 | 0.79 | 0.95 | 0.71 |
| ResNet | 0.63 | 0.79 | 0.99 | 0.72 |

A visualization of the learned feature representations are depicted in figure 6.1. The results are obtained on the Kaggle-DB1 using the t-SNE algorithm to reduce the representations down to two dimensions. The visualization of the SIFT and SURF feature representations' provided no interesting findings since all points were lumped together. Therefore, the feature visualizations are limited to those provided by the CNN networks. The traditional machine learning methods' inability pinpoints the expressive power of CNN networks when trained over complex image data. Furthermore, compared to the specialization-project representations in figure 2.1, similar points are better grouped and separated from

the other classes. Therefore, it seems evident that the supervised training provides better representations for classification than those provided by the specialization-project unsupervised models. Lastly, there are no clear differences in the three feature representations. Albeit observed clear differences in the classification performance in table 6.2, the similarities when the representations are reduced to two dimensions indicates that the networks learned the same main features.

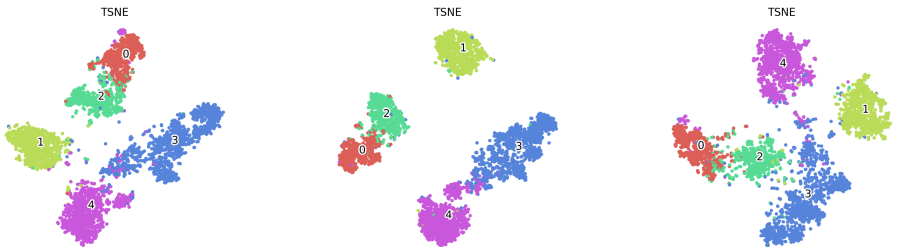



**Figure 6.1:** Low dimensional visualization of the learned representations using supervised training. **Left image:** Feature representation using 5-CONV as backbone network. **Middle image:** Feature representation using VGG16 as backbone network. **Right image:** Feature representation using ResNet50 as backbone network.

Figure 6.2 provides the same t-SNE visualization as in figure 6.1 using a VGG16 as backbone. However, the point samples are replaced by the real images used to make the two-dimensional embedding. Observe that the feature representation clearly separates the *acantharia protist* (yellow samples) and *copepods* (blue samples) from the rest. The *protist other* (cyan samples) class are more or less separated from the others but somewhat stretched out. This seems reasonable since the class contains a diverse set of species with very varying features. Lastly, the most similar classes are seemingly *diatom chains* (red samples) and *faecal pellets* (green samples), having an overlapping region of points. This is only to be expected since a broad class of *faecal pellets* have features that look more similar to the long thin, and transparent *diatom chains*[1] than the thick and curvy shapes of other class members.

[1]Notably, some faecal pellet species share similar characteristics with the diatom chains making it difficult to distinguish these objects for human non-experts. This is possibly not true in the case of using domain level experts.

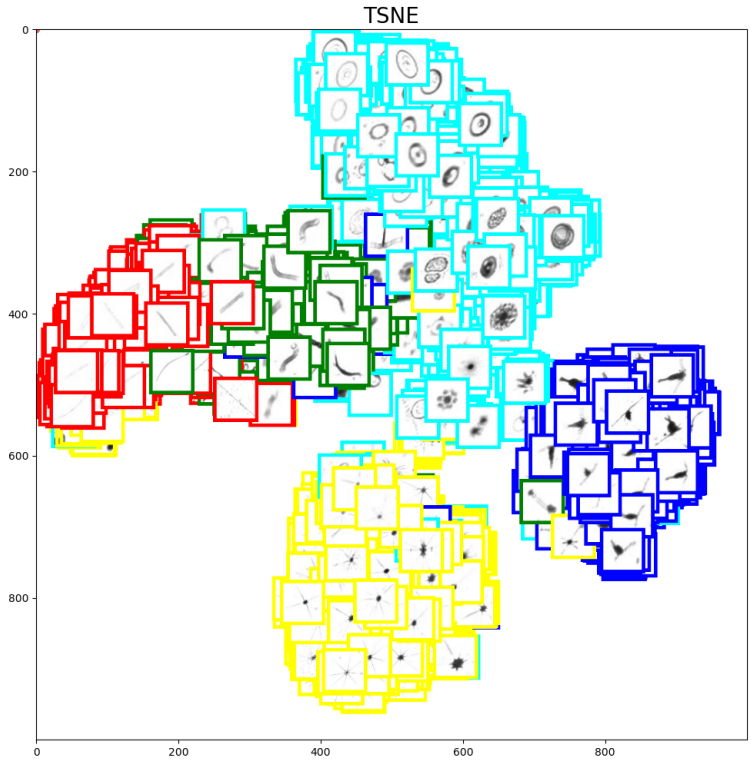



**Figure 6.2:** Low dimensional visualization of the learned representations using supervised training. The backbone network is VGG16. For each colored sample point, the corresponding test image is pictured. Observe that the model has difficulties separating some species from the *diatom chain* class (red samples) and *faecal pellet* class (green samples) resulting in an overlapping region.

### 6.1.2 Traditional machine learning feature extractors

Figure 6.3 shows the features detected using the SIFT and the SURF machine learning algorithms. Observe that the models are only capable of capturing a small number of features from each image. The images shows that the models can better detect round objects with darker contours, whereas transparent bodies and long antennas are easily missed. This pinpoints how challenging it is to find commonalities in plankton data compared to many other data sets for which the SIFT and SURF algorithm are much more efficient. Table 6.3 depicts the classification results using the machine learning feature extractors followed by the k-means and SC clustering algorithms. Clearly, none of the extraction methods proved much better than the other one. Using K-means and SC clustering, an accuracy of around 40% was achieved. This is a decrease in accuracy compared against clustering without first performing feature extraction showing the infeasibility of traditional machine learning feature extraction for this task.

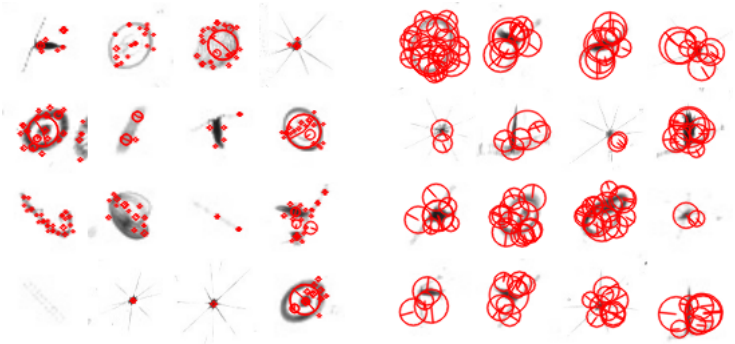


**Figure 6.3:** The figures show the found feature descriptors (red circles) on random images from the Kaggle-DB2. **Left image:** The resulting feature descriptors using the SIFT algorithm. **Right image:** The resulting feature descriptors using the SURF algorithm.

**Table 6.3:** Baseline classification results on Kaggle-DB1

| Feature extractor | K-means | SC |
|---|---|---|
| SIFT | 0.40 | 0.41 |
| SURF | 0.42 | 0.40 |

### 6.1.3 Autoencoder

The autoencoder model described in section 4.3.2 was trained for 100 epochs yielding a training loss of $0.0429$ and a validation loss of $0.0546$. Compared to the specialization-project models, the loss is nearly ten times as high, and there is a bigger gap between training and validation loss. The relatively high loss values might suggest underfitting. However, there might also be several reasonable explanations for the higher loss values. First, as data augmentation is applied, the regularization might reduce the model's capability to learn a good reconstruction. However, a more likely explanation is that the incorporated rotation in-variance removes information about the exact object positions. Since the loss is measured pixel against pixel, a low loss can only be achieved if the object is identical and perfectly placed in relation to the input. This theory is supported by the reconstructed images depicted in figure 6.4. Clearly, the decoder part constructs images that contain some main object features. However, the images are noisy, and object bodies are placed at several locations in the same image. Interestingly, the round image samples seem to have better image reconstructions. This is fitting since positional information is less important for objects that are equal in all directions.

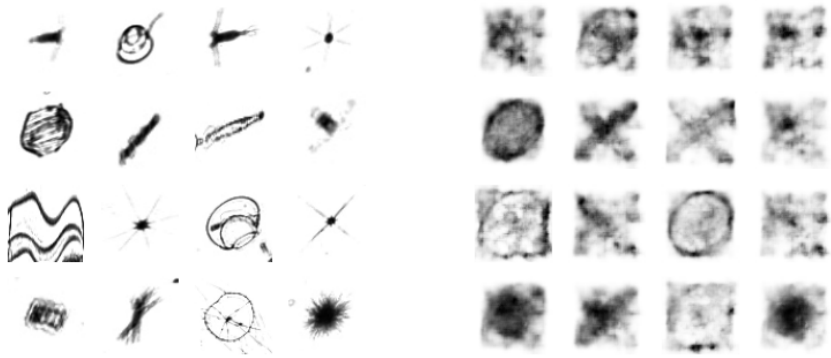


**Figure 6.4: Left image**: Random images from the Kaggle-DB2 data set. **Right image**: Image reconstructions after passing the images through the autoencoder model. Observe that the model has learned different plankton features. However, the rotation in-variance seems to affect the model's reconstruction capability. Without knowledge of the exact position, the decoder part seems to "guess" the more likely placement of the object's main body.

Table 6.4 shows the proposed rotation invariant model's clustering results, denoted as "Rot-Auto," tested on the Kaggle-DB1. Compared to the specialization-project reference model, the classification performance is increased by 11% and 17% for the K-means and SC algorithm, respectively. The increased results indicate that the rotation invariant model has learned feature representations that are better for classification. As experienced in the specialization-project, the SC algorithm outperforms the K-means clustering model. This suggests that the learned feature representation have non-linear traits.

**Table 6.4:** Autoencoder classification results on Kaggle-DB1

| Model | K-means | SC |
|---|---|---|
| Rot-Auto | 0.73 | 0.93 |

By examining the plots in figure 6.5 acquired using the t-SNE and PCA algorithm, it becomes clear that a non-linear algorithm better represents the features. From the t-SNE plot, it is evident that the new autoencoder model is much better at treating similar class objects of different rotation compared to the specialization-project models. Furthermore, the feature representation seems close to those obtained by a supervised feature extraction model.

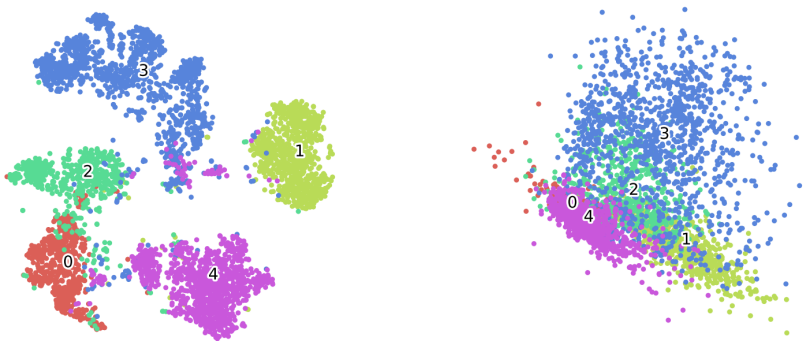



**Figure 6.5:** Low dimensional visualization of the learned autoencoder feature representation space. **Left image:** T-SNE representation. **Right image** PCA representation.

To validate that the model has indeed learned important features, the model was tested using the GAP-CAM method. Note that the network is trained to reduce the error between the input and the reconstruction, suggesting a focus on the entire image and more general feature shapes. The GAP-CAM visualization are shown in figure 6.6 proving that the network is clearly identifying important image regions. As expected, the network seems more interested in large clusters of dark pixels. Still, different parts and features are clearly focused depending on the input species. Observe that long and thin features such as antennas seem focused in several images. This might be important since many species are easily differentiated based on such features.

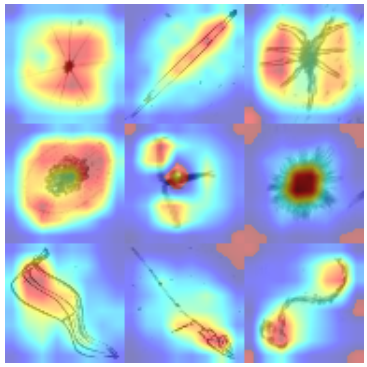


**Figure 6.6:** The figure depicts a heatmap of the specific image regions the autoencoder model finds more interesting based on different plankton images.

### 6.1.4 GAN

The GAN model described in section 4.3.3 was trained for 100 epochs using the Kaggle DB2 with data augmentation. The model losses are slowly converging closer to zero, but in contrast to conventional neural networks, the two losses are constantly fluctuating

depending on which model performs better. Thus, it was hard to interpret how the loss represents what the model has learned, and tuning was mostly done based on visualization of the constructed generator images. The GAN model proved very challenging to train and required extensive hyperparameter training to achieve realistic-looking images. Furthermore, running several training rounds on the same settings provided very different performance results. Since the deep learning model is a black-box approach, it is challenging to assess the varying model performance reasons. However, a likely explanation is that the oscillatory weight shifts endured during training makes the model convergence more unstable. Another reason for the poor performance might be because the network structure is quite shallow and constrained. Making the discriminator more complex or less regularized resulted in the generator network being unable to converge to a proper solution. The model is therefore not fine-tuned for better extractions but tuned with respect to the generator model.

**Figure 6.7:** The figure visualizes the generated images at different time steps of the GAN training. Observe that the images slowly gets more complex and more detailed.

A time series can be depicted in figure 6.7 showing how the generated images slowly get more complex. After about ten epochs, the generator part manages to create shapes and forms that are seemingly similar to real planktonic images. As the training progresses, the quality and image details are increasing, which, according to Wang et al. [114] suggests

that the discriminator part is learning to distinguish based on more fine-grained details.[2] Observe that especially the round objects seems to become more elongated at later time steps. This might be a deficit due to the image augmentation, where images are shifted, rotated, and zoomed. Training was thus also performed without augmenting the images. Removing the data augmentation seemed to improve the generated images slightly but also decreased the later classification results.

The fluctuating and non-converging function loss made it difficult to analyze the potential model overfitting in the conventional way. In accordance with Yazici et al. in [120] each generated image was therefore compared to the closest pixel-wise neighbor in the training set. As depicted in figure 6.8 the generated images (left) are clearly not the same as the true images (right), which shows that the model has not overfitted by learning exact copies of the training data. Furthermore, by generating several different images based on different input vectors, results showed a wide variety in plankton species indicating that the model does not suffer from *mode collaps*.

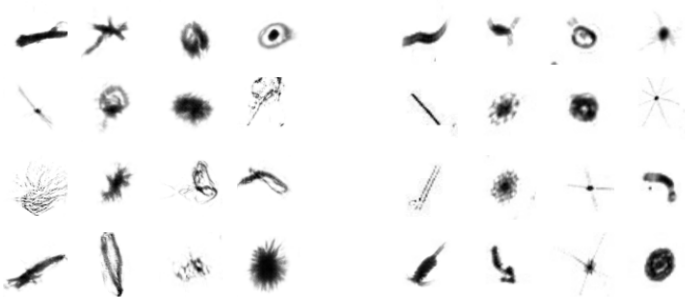


**Figure 6.8:** The image shows a set of generated images (**left image**) compared against the closest neighboring images from the training set (**right image**). Observe that the images are not the same, suggesting that the model does not directly copy the input images.

The classification results can be depicted in table 6.5 the GAN model is denoted ”GAN”. The GAN model was trained five times, and the average performance over these runs are presented. At best, the model achieved an accuracy of 0.7 and 0.79 for the K-means and SC, respectively, which is a clear improvement over the specialization-project models and close to the supervised baseline networks. However, the classification results were varying, resulting in an average result of 0.64 and 0.73, which is similar to the specialization-project models. In short, the best classification results show that the GAN model has a potential as a feature extraction model. Still, the varying classification performance makes it less

[2]Note that the quality of specific features and the quality of the images compared to real plankton species are difficult to assess without a more profound understanding of biological taxa. Still, it is evident that the images are strongly related to planktonic species, which indicates that the network has captured essential features and structures of the data.

attractive compared to the other models.

**Table 6.5:** GAN classification results on Kaggle-DB1

| Model | K-means | SC |
|---|---|---|
| GAN | 0.64 | 0.73 |

The GAN feature representations using the t-SNE and PCA algorithm, depicted in figure 6.9, shed some more light on the classification results. The t-SNE visualization (left image) shares many commonalities with the visualizations obtained in the specialization-project, indicating that the learned feature representation is very similar. The close resemblance in classification results are therefore only to be expected. Furthermore, the GAN is also sharing the problem of class separation due to similar species with different rotations[3]. In contrast to the supervised multi-classification training scheme, the binary GAN training does not augment the representation space towards a more rotation invariant space. Lastly, in the PCA representation (right image), similar class samples are not very well grouped and separated from the other categories. This indicates that that the learned representation have non-linear traits, which is less compatible with linear classifiers.

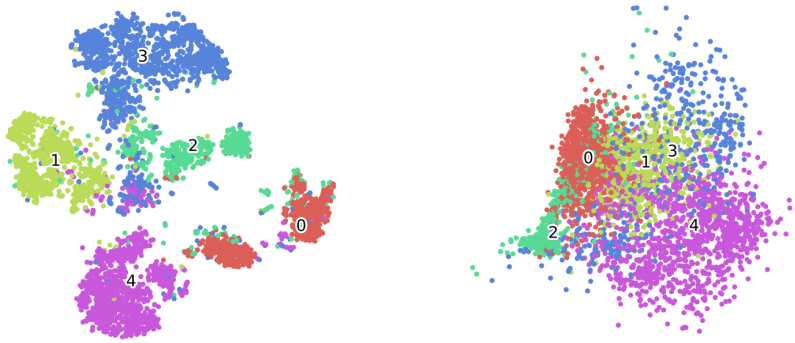



**Figure 6.9:** Low dimensional visualization of the learned GAN feature representation space. **Left image:** T-SNE representation. **Right image** PCA representation.

Similar to the rotation invariant autoencoder, a visualization of the image regions that were more important for decision making is depicted in figure 6.10. The image is created by the grad-CAM approach since the GAN have a structure which is not compatible with the GAP-CAM method. Evidently, the decisions are based on important image regions and clearly focusing on different parts depending on the planktonic species. However, for most images, the network emphasizes the large and pixel intense areas, whereas antennas and other tiny artifacts further from the body center are easily missed. Therefore, it seems

[3]The class separation is especially visible in the red class consisting of diatom chains that are either vertically or horizontally aligned.

evident that the training scheme based on distinguishing real from fake images does not focus on specific and highly discriminate plankton features.

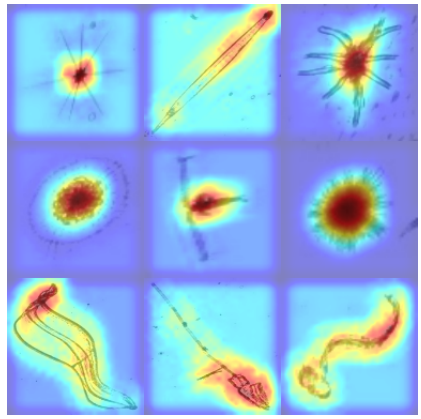


**Figure 6.10:** The figure depicts a heatmap of the specific image regions the GAN model finds more interesting based on different plankton images.

### 6.1.5 Deep cluster results

The Deep Cluster model described in section 4.3.4 was trained using three different backbone networks, 5-CONV, VGG16, and ResNet, respectively. All networks were trained for a maximum of 80 epochs but stopped earlier using an early stopping criteria to avoid overfitting. Note that the supervised training scheme makes it possible to assess overfitting by comparing the training loss against the validation loss. However, since the pseudo-labels' quality compared to the ground truth labels is relatively poor, the loss comparison is less valuable. The training was, therefore, stopped based on two conditions: 1. The training loss between several consecutive epochs is not decreasing noteworthy. 2. The number of samples that change cluster between several successive epochs is not reducing noteworthy. These metrics are plotted in figure 6.11 where the DC using VGG16 as backbone is trained for 50 epochs. Clearly, the loss is saturating at about epoch 40, whereas the reassignment score measured using NMI saturates earlier. This resulted in the 5-CONV being trained for 15 epochs, VGG16 for 40, and ResNet for 60 epochs.

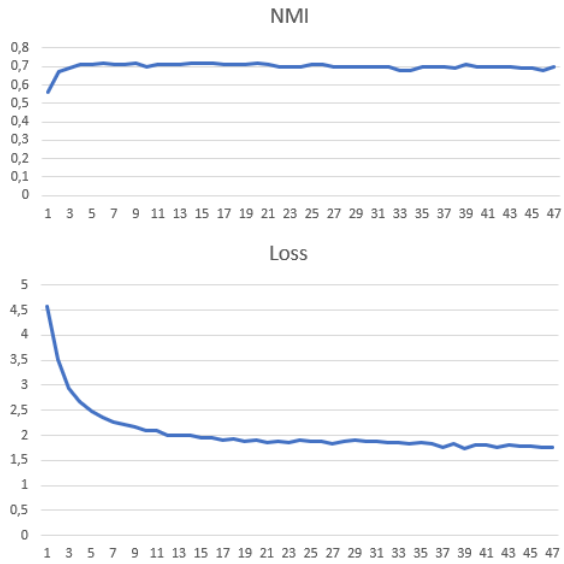



**Figure 6.11:** The upper figure depicts the NMI score comparing the labeling of consecutive rounds. Observe that about 30% of the samples change labels every time. The lower image depicts the loss measured using the model predictions and the pseudo labels. The loss seems to stabilize around epoch 40.

The Deep Cluster training loss, training accuracy, and cluster reassignments are reported in table 6.6. Note that these results only give an indication about the true model performance as the model assignments are compared against the pseudo-labels. The deeper neural network models perform better, with the VGG16 performing slightly better than the ResNet. The NMI is saturating at about 0.7, meaning that a large fraction of images are reassigned every epoch. Caron et al. observed the same phenomenon in [15], but claim that the reassignments do not result in model divergence.

**Table 6.6:** Deep Cluster training performance on Kaggle-DB2

| Backbone | Training loss | Training ACC | Cluster reassignments |
|---|---|---|---|
| 5-Conv | 2.59 | 0.33 | 0.7 |
| VGG16 | 1.77 | 0.48 | 0.74 |
| ResNet | 2.01 | 0.46 | 0.74 |

Table 6.7 depicts the average classification results over five independent runs on the Kaggle-DB1 where DC models are denoted, based on the backbone network, as ”DC-5-CONV”, ”DC-VGG16” and ”DC-ResNet”. The best classification accuracy is obtained using VGG16 as backbone, achieving an accuracy of 88% and 92% for the K-means and SC algorithm, respectively. Compared to the best specialization-project model, the K-means performance has improved by 22% indicating that the representation space is more friendly to linear

classifiers. Replacing the VGG16 backbone with the 5-CONV or ResNet architecture showed a similar increase in linear classifiers performance. Still, the models did not manage to obtain the same performance using the SC algorithm and perform more similar to the specialization-project models.

**Table 6.7:** DeepCluster classification results using different backbone networks on Kaggle-DB1

| Model | K-means | SC |
|---|---|---|
| DC-5-Conv | 0.69 | 0.76 |
| DC-VGG16 | 0.88 | 0.92 |
| DC-ResNet | 0.68 | 0.71 |

Visualizations of the deep cluster feature representations over the Kaggle-DB1 are depicted in figure 6.12 and 6.13. The t-SNE plot in figure 6.12 clearly show that that the models are able to learn strong representations that mostly group similar points and separate the different classes. The t-SNE visualization is much better than those obtained by the specialization-project models showing the strength of the DC technique. Furthermore, the striking similarities in the three feature representations indicating the DC methods robustness to changes in the backbone network. Still, the DC method with a VGG16 backbone network seems to produce a slightly better feature representation, which is in line with the findings in table 6.6 above. Altogether, the t-SNE visualizations show that the DC method can produce representations that looks as good as the features produced by the rotational autoencoder and close to those provided by the supervised networks. Of similar interest is the PCA plot in figure 6.13. Compared to the other unsupervised networks, similar classes seem better grouped and separated from the other classes suggesting that the models can learn more linear feature representations.

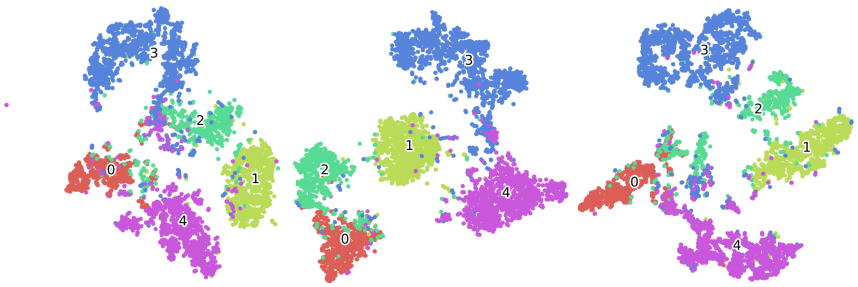



**Figure 6.12:** Low dimensional visualization of the learned DC feature representation space using the t-SNE algorithm. **Left image:** DC model with 5-CONV backbone. **Middle image** DC model with VGG16 backbone. **Right image** DC model with ResNet backbone.

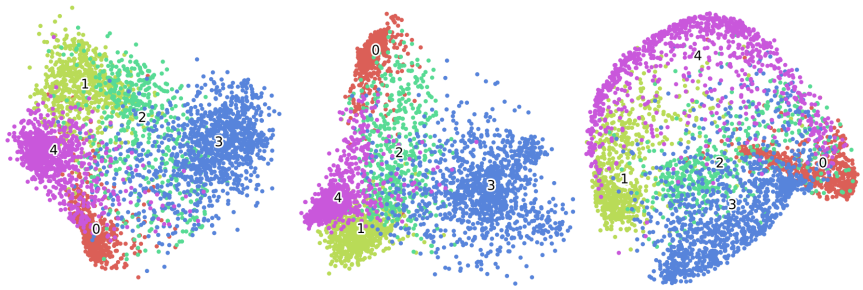



**Figure 6.13:** Low dimensional visualization of the learned DC feature representation space using the PCA algorithm. **Left image:** DC model with 5-CONV backbone. **Middle image** DC model with VGG16 backbone. **Right image** DC model with ResNet backbone.

The model's class activation maps are visualized in figure 6.14 to validate that the models have learned relevant image features. The map is created using the grad-CAM approach since the network structures are not compatible with the original CAM approach. The 5-CONV network seems to highlight important regions for most images. However, for some images, the network activates around the object, which indicates a failure in recognizing class specifics. The ResNet model seems even worse, focusing on the same regions regardless of the input image. These problems are not apparent in the VGG16 activation map. In this map, the decisions seem focused on essential areas irrespective of input. Still, the network appears more focused on large and pixel intense areas, whereas antennas and other tiny artifacts further from the body center are missed in some cases.

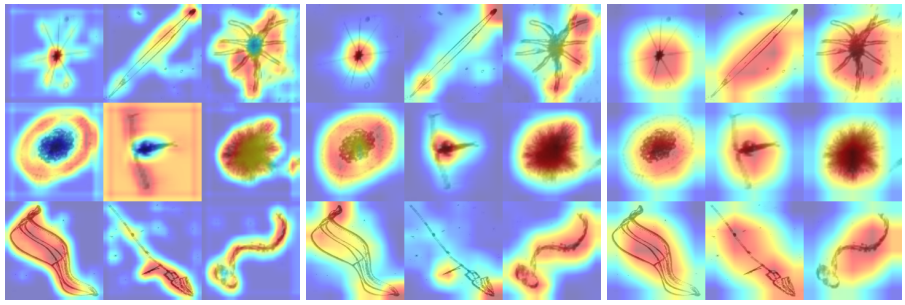


**Figure 6.14:** The figure depicts a heatmap of the specific image regions the DC model finds more interesting based on different plankton images. **Left image:** DC model with 5-CONV backbone. **Middle image:** DC model with VGG16 backbone. **Right image:** DC model with ResNet backbone.

### Summary and reflection

This experiment revisits and extends the work conducted in the specialization-project with the goal of finding a better model for unsupervised feature extraction. The baseline experiment manifested the challenges of using machine learning in the plankton domain. The traditional machine learning models were seemingly incapable of extracting a set of features

that could improve the subsequent clustering. The performance increased by switching to an unsupervised deep feature extraction network. Of the three networks, the GAN network gave the least satisfying results and did not improve performance over the specialization-project models. The rotation invariant autoencoder proved to be much better and greatly improved the performance over the specialization-project models. Furthermore, the results from the t-SNE visualization and the classification accuracy using SC gave results very similar to those obtained by the supervised baseline models. However, this was not true when using a linear classifier, suggesting that the feature space includes several non-linear relations. As experienced by the other methods, an increase in network depth is likely to improve the model's representational capability. Unfortunately, the large increase in feature maps proved to be very memory expensive, making it impossible to increase the depth over that of a 5-CONV network with the current computer specifications. The DC approach provided excellent results when using a VGG16 backbone. Of special importance was the large increase in linear clustering results and the seemingly well-separated representation space using PCA which suggest an ability to learn a linear representation of the data. In contrast to the supervised baseline results, the ResNet model did not give satisfying classification results and the class activation map revealed that the ResNet model does not focus on particular class characteristics. A suggestion is that the network's design with several skip connections that give increased learning speed and larger gradients makes the model more vulnerable to ever changing labels and several misclassifications. Overall the best performing networks were the rotation invariant autoencoder and the DC model using a VGG16 as the backbone. These networks are therefore picked and used as feature extraction models in the next experiments.

## 6.2 Experiment 2 - Choosing an appropriate unsupervised framework

This section presents the second experiment results described in section 5.4.2 using the best feature extraction models from the previous experiment. The results are presented in the following order. First, in section 6.2.1, the deep learning feature extractors are validated over a test set of unseen samples from the Kaggle data. Second, in section 6.2.2, the classification matrix and its relation to the learned feature representations are investigated. Third, section 6.2.3 presents the models ability to adapt to new unseen plankton categories. Fourth, section 6.2.4 reveal the performance results of several different classification models. Finally, the section results are summarized and briefly reflected upon.

### 6.2.1 Model capability over unseen test data

The feature representation of the hundred unseen images are depicted in the t-SNE visualization in figure 6.15. Evidently, both models seem to have generalized well and manage to group the same class species and separate the different categories. Overall, the DC model seems to produce a slightly better representation with fewer outliers. However, the set is tiny and does not contain enough samples to represent the same category species' vast variation.

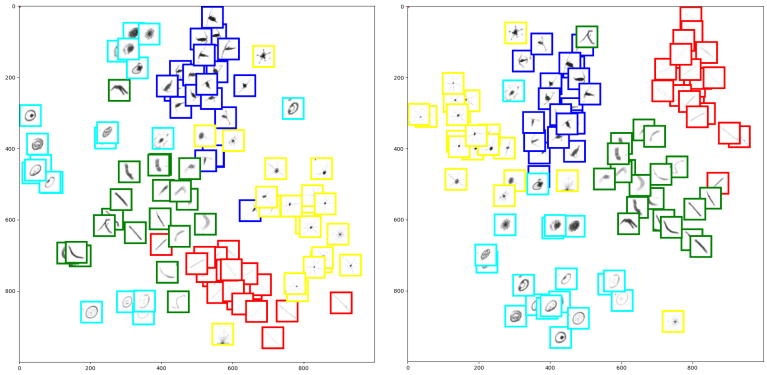


**Figure 6.15:** Low dimensional visualization of the learned representations using 100 unseen test images. **Left image:** Features extracted from the rotation invariant autoencoder model. **Right image:** Features extracted from the DC model using a VGG16 as backbone.

A similar conclusion can be drawn from the classification results given in table 6.8 where features are extracted by the autoencoder and DC networks and the resulting feature vectors clustered using the k-means algorithm. Observe that the clustering accuracy is higher than when clustering is performed on the seen samples from the Kaggle-DB1. This indicates that the extraction models have not overfitted to the seen data. The DC algorithm is performing slightly better. This was expected since the first experiment revealed that the DC models produced a better linear representation. Furthermore, as seen in figure 6.15 the few outliers in the autoencoder representation have a great impact when the sample size is small.

**Table 6.8:** Classification results on unseen data samples from the Kaggle data set

| Model | ACC | F1 | NMI |
|---|---|---|---|
| Auto-K-Means | 0.82 | 0.82 | 0.70 |
| DC-K-Means | 0.93 | 0.93 | 0.86 |

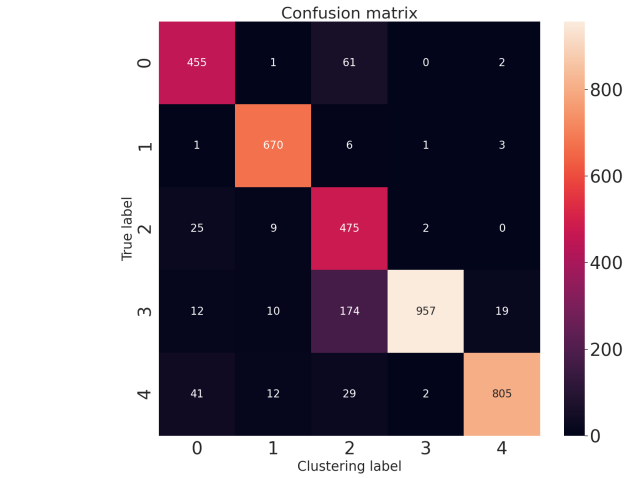
The model performance does not seem to have noticeable deviations from the results acquired when the models are tested on parts of the seen training data. Thus, to increase the validation data set size to get a more comprehensive prediction and clustering analysis, the following tests are done on the Kaggle-DB1, which is a part of the larger Kaggle-DB2. Importantly, this setup follows the practice of several previous works [16, 46, 117, 119] in the unsupervised domain.

### 6.2.2 Validation of confusion matrix and cluster assignments

The DC feature representations and a corresponding k-means based classification matrix is depicted in figure 6.16 to get a better understanding of how the representations affects the clustering and which categories that are more difficult to classify correctly. Note that

the true classes are denoted 0-4 in the classification matrices correspond to the five classes seen in figure 2.8. The faecal pellet (true label 2) class seems to be the most difficult class to differentiate from the other categories, which can be seen by the large number of assignments to cluster 2 and especially from the protist (true label 3) class. The high number of misclassifications is not surprising as the faecal pellet class consists of a wide range of differently shaped objects. More specifically, the main problem seems to be a wrong assignment of a small set of protist samples are treated as outliers by the feature extraction model. Secondly, the feature representation had problems separating diatom chains (true label 0) against light and straight samples from the faecal pellet class. This is expected as these categories share many similarities making it difficult to separate them even for human non-experts. It is evident that the cluster groups are made up of species with very similar traits, which increases the trustworthiness of the DC models ability to extract key features from the data. Furthermore, the model can produce a linear representation space that makes it possible to use Euclidean distance-based classification algorithms.

Similar to the evaluation above, the autoencoder feature representations and its corresponding SC based classification matrix is depicted in figure 6.17. Note that the SC algorithm was chosen over k-means as it was much more accurate and the feature representation is much less ideal using a linear classifier. Observe that almost 70 samples from the protist (true label 3) class are wrongly assigned to the achantaria (true label 4) class. These errors can be depicted in the t-SNE plot as the as a set of yellow samples which in reality belongs to the blue class. These particular images are, however, indistinguishable from the achantaria class for the untrained eye having small round bodies and antenna-like extensions. A second error source comes from the faecal pellet (true label 2) class assigned to the diatom chain (true label 0) class. In the t-SNE plot, these errors come from the border between the light and red class and is unsurprising as the feature representation have trouble differentiating light and straight samples from the faecal pellet class against the very similar diatom samples. Overall, the clustering performance is good, providing only a small number of labeling errors. In the case of wrong assignments, it is evident that the misclassifications are based on samples sharing very similar traits as the predicted class.



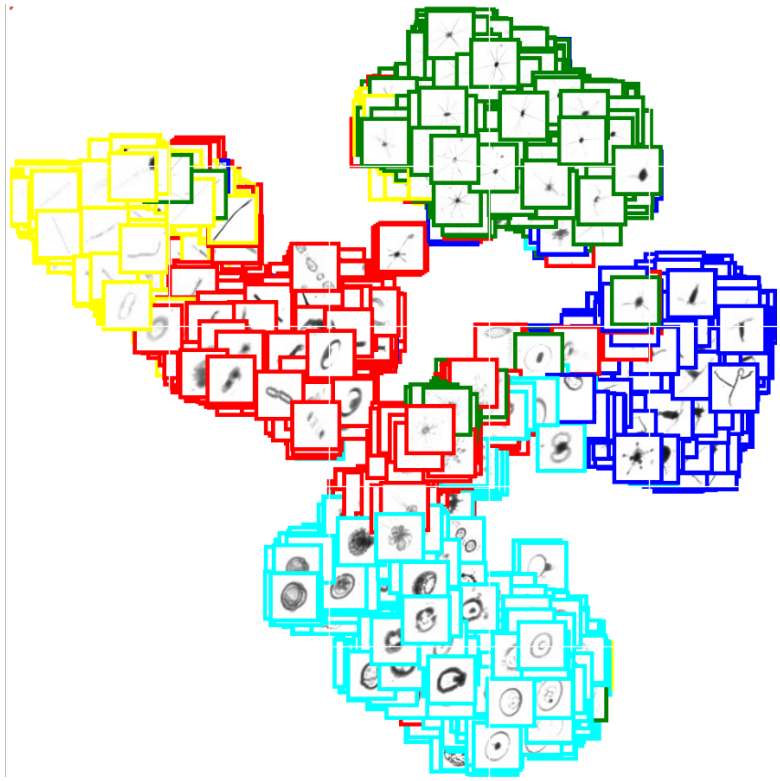



**Figure 6.16:** The figure is showing the confusion matrix and t-SNE visualization of the predictions from a DC network and a k-means algorithm. The classes are 0: ”diatom chain string” 1: ”copepod calanoid” 2: ”feacal pellet” 3: ”protist other” 4: ”acantharia protist.

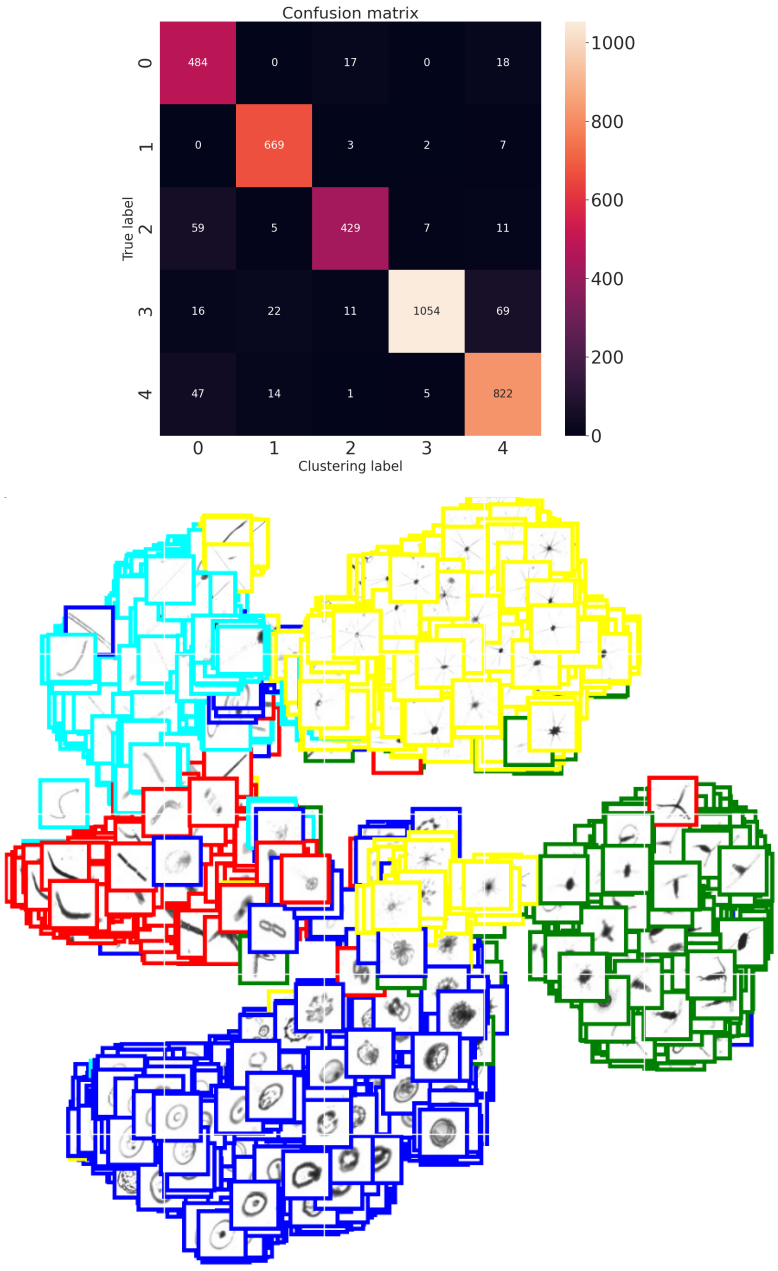



**Figure 6.17:** The figure is showing the confusion matrix and t-SNE visualization of the predictions from an autoencoder network and a SC algorithm. The classes are 0: ”diatom chain string” 1: ”copepod calanoid” 2: ”feacal pellet” 3: ”protist other” 4: ”acantharia protist.

### 6.2.3 Model adaption to new classes

Table 6.9 depicts the classification results when the feature extraction models are tested over unseen data samples. The models denoted by ”Missing-” refers to the deep learning feature extractor trained on a subset of the Kaggle-DB2 with several classes removed. All models are then tested over data sets, including the missing classes. Comparing the DC models, one can observe a performance decline when the model has not been trained over all classes. This indicates that the model has not learned to extract sample specific features making the model less adaptive to new and unseen classes. The performance decline is, in comparison, very minor for the autoencoder models. This suggests that the autoencoder training method is better capable of learning general feature representations that are easily adapted to unseen classes.

**Table 6.9:** Comparison of the feature extraction models ability to adapt to new unseen classes

| **Model** | **Kaggle-DB1** | | **Kaggle-DB2** | |
|---|---|---|---|---|
| | **ACC** | **NMI** | **ACC** | **NMI** |
| Auto-SC | 0.93 | 0.81 | 0.14 | 0.27 |
| Missing-Auto-SC | 0.92 | 0.78 | 0.15 | 0.27 |
| DC-SC | 0.90 | 0.80 | 0.10 | 0.26 |
| Missing-DC-SC | 0.67 | 0.64 | 0.10 | 0.25 |

### 6.2.4 Capability of the classification part

Table 6.10 presents the classification performance of the classification models described in section 4.4 using features extracted from the Autoencoder (Auto) and the DC model. First, observe that the three classification metrics seem to correspond quite well, increasing the credibility of the results being correct. The best performance is achieved by the SC algorithm independent of the feature extraction network. The results fit well with the findings in the specialization-project and in [113] which states that the SC algorithm most often yields much better results compared to linear distance-based clustering models. The other conventional classification algorithms performs slightly worse, with the BIRCH method being moderately better than the others. Lastly, the DEC approach did not improve the classification results but instead worsened the learned data representation, resulting in poor performance. Note, however, that the DEC layer is based on the k-means algorithm meaning that an untrained version can perform as good as this clustering model.

**Table 6.10:** Classification results on Kaggle-DB1 using different clustering algorithms

| Model | ACC | F1 | NMI |
|---|---|---|---|
| Auto-K-Means | 0.75 | 0.72 | 0.61 |
| Auto-SC | 0.93 | 0.93 | 0.81 |
| Auto-Birch | 0.74 | 0.72 | 0.63 |
| Auto-Gaussian | 0.73 | 0.72 | 0.60 |
| Auto-DEC | 0.60 | 0.57 | 0.46 |
| DC-K-Means | 0.77 | 0.76 | 0.71 |
| DC-SC | 0.90 | 0.91 | 0.80 |
| DC-Birch | 0.84 | 0.84 | 0.74 |
| DC-Gaussian | 0.81 | 0.81 | 0.71 |
| DC-DEC | 0.72 | 0.72 | 0.68 |

Figure 6.18 depicts the time consumption for an increasing image load using the aforementioned clustering algorithms and a VGG16 feature extraction network. Observe that the SC is a much slower algorithm than the others and requires a substantial amount of computation time. The k-means and the k-means based DEC model have the fastest prediction times where the most time-consuming part is the overhead from the extraction model. The BIRCH and the Gaussian mixture model have a slightly higher time consumption but are still capable of predicting over 800 images per second, which is over the AILARON real-time requirements.

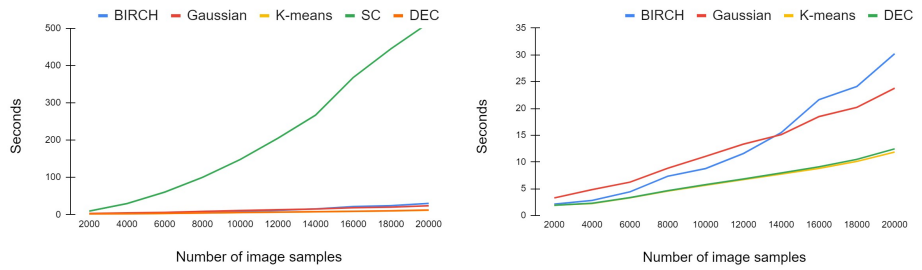



**Figure 6.18:** The figure depicts the time consumption for an increasing image number using different clustering algorithms and the VGG16 feature extraction model.

### Summary and reflection

This experiment explored the capabilities of the two best feature extraction models, namely the rotation invariant autoencoder and the DC model with VGG16 as backbone, from the previous experiment and tested the performance of several classification models. First, the

feature extraction models were tested on 100 test samples to prove the models' ability to generalize to unseen data samples. The results proved very similar to those conducted on the seen samples from the Kaggle-DB1, proving good generalization qualities. However, validating performance using a small test set proved less ideal since the data does not fully represent the large variations in the data, and small prediction errors greatly impact the classification results. Further experiments were therefore conducted on the larger, albeit seen, Kaggle-DB1 data set. First, the feature representations and clustering assignments were inspected more closely by examining the classification matrix's commonalities and the t-SNE image visualization. Both models showed excellent representative capability finding that the feature extraction and clustering errors seem to come from samples that share many of the given class characteristics. Overall this indicates that the unsupervised deep learning models have learned strong and robust representations of the underlying data. However, the DC model did not adapt very well when used to extract information from data, including plankton categories the model was not trained to recognize. This might relate to the training scheme indicating the downside of organizing the feature representation space based on labeled training data. In comparison, the autoencoder model was much better able to adapt to unseen data emphasizing the models' ability to learn general feature characteristics. Albeit an effective way to learn good feature representations, the introduction of pseudo-labels seems thus to negatively affect other important traits of the deep learning feature extraction. Finally, the classification models, explained in section 4.4, was tested and validated on the Kaggle data. The best performing algorithm in regards to classification accuracy and NMI score was the SC outscoring the rest irrespective of feature extraction model. Unfortunately, the transductive properties of the SC algorithm meaning it does not generalize to new data, is a big drawback of this method. The SC algorithm must therefore observe the test data before making predictions, which proves to be very time-consuming. In contrast, the other algorithms can generalize using training data and can, thus, make much faster predictions. The BIRCH is therefore a good alternative if the task requires faster prediction times.

## 6.3 Experiment 3 - Evaluation on the AILARON data

This section presents the third experiment results described in section 5.4.3 testing the proposed framework over the AILARON data. The results are presented in the following order. First, in section 6.3.1, the baseline is set using no feature extraction prior to clustering as well as assessing the capabilities of supervised deep feature extraction. Second, in section 6.3.2, different components of the proposed unsupervised framework is validated and assessed over the AILARON data. Finally, the section results are summarized and briefly reflected upon.

### 6.3.1 Baseline methods

The baseline classification results over the AILARON data are depicted in table 6.11. The K-means obtains 59% cluster accuracy and the SC 63% cluster accuracy without using feature extraction. By switching to a supervised deep learning feature extraction model, denoted as "5-CONV", "VGG16," and "ResNet," the performance is slightly improved,

reaching a clustering accuracy of 73% in using the ”VGG16” network. However, compared to the Kaggle data set results, the performance gain by adding a deep feature extraction model is only minor. This suggests that the supervised deep learning networks are less able to find good feature representations over the AILARON data.

**Table 6.11:** Baseline classification results on AILARON-DB1

| Feature extractor | K-means | SC |
|---|---|---|
| - | 0.59 | 0.63 |
| 5-CONV | 0.67 | 0.62 |
| VGG16 | 0.73 | 0.73 |
| ResNet | 0.66 | 0.63 |

To gain more insight about the supervised deep learning performance, table 6.12 depicts the training and validation results after training the proposed backbone networks in a supervised fashion. The networks are trained over the AILARON-DB2 using the same parameter settings as done on the Kaggle data in section 6.1. Similar to the Kaggle data results, the deeper network architectures achieved better performance over the data indicating a stronger representational capability. However, as mentioned above, the good training performance, which indicates good representational capability, is contrasted by the low performance when combining deep feature extraction and clustering in table 6.11.

The low-performance increase can partly be answered by comparing the low dimensional t-SNE image representation of the AILARON data with and without the deep feature extraction part. Without using a feature extraction algorithm (left image), several of the categories are mixed together. In contrast, the samples are much better grouped according to their class and separated from the others when using a deep learning feature extractor (right image). However, the most decisive contribution to the classification accuracy is to correctly classify the red points representing the ”bubble” class and pink points representing the ”other” class, which collectively stands for nearly 65% of the data set. A machine learning classifier can, based on the t-SNE plot, clearly separate the ”bubble” points (red class) from the other classes. By clustering the groups of mixed samples into the five remaining categories, the accuracy easily surpasses 60%. The deep feature extraction seems to obtain better groupings of similar points but also splits the ”bubble” points (blue class) into two cluster groups. The split results in a significant drop in accuracy despite a much more precise clustering of the minor classes (red, purple, and pink class).

**Table 6.12:** Supervised neural network training performance on AILARON-DB2

| Model | Training loss | Training ACC | Validation loss | Validation ACC |
|---|---|---|---|---|
| 5-Conv | 0.70 | 0.76 | 1.14 | 0.67 |
| VGG16 | 0.5827 | 0.79 | 0.95 | 0.71 |
| ResNet | 0.35 | 0.85 | 0.33 | 0.85 |

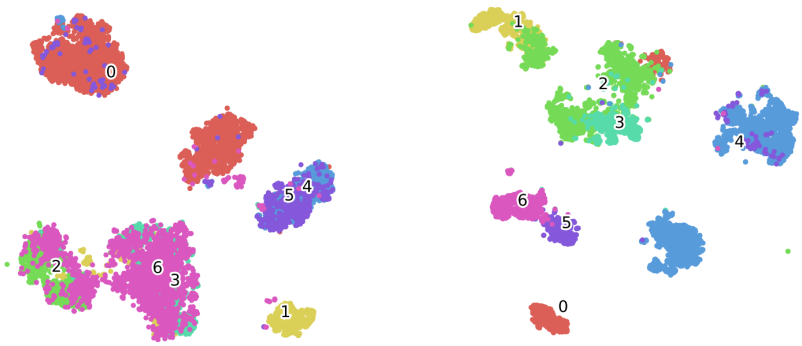



**Figure 6.19:** Low dimensional t-SNE visualizations on the AILARON data. **Left image:** T-SNE visualization without feature extraction. **Right image:** T-SNE visualization using supervised deep learning feature extraction.

### 6.3.2 Unsupervised models

Figure 6.20 shows the results of performing feature detection using the SIFT and the SURF machine learning algorithms. The models clearly struggle to detect useful structures resulting in zero features for some images and detect several background points that are not part of the image object. However, the models are clearly better at finding structures in the images depicting planktonic species compared to images of oil and gas, showing to some extent the suitability of the AILARON camera system for capturing details in planktonic imagery. However, the models better detect darker areas of the plankton images, whereas transparent body parts and long antennas are easily missed. The overall low detection capability seems validated by the classification results in table 6.13 showing a performance decrease compared to clustering without any feature extractor.

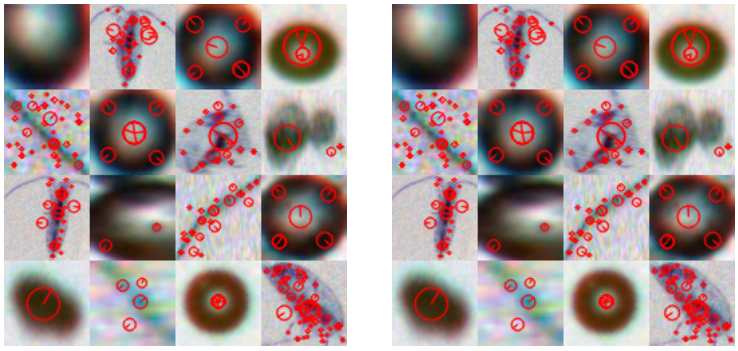


**Figure 6.20:** The figures show the found feature descriptors (red circles) on random images from the AILARON-DB2. **Left image:** The resulting feature descriptors using the SIFT algorithm. **Right image:** The resulting feature descriptors using the SURF algorithm.

**Table 6.13:** Traditional machine learning classification results on the AILARON-DB1

| Feature extractor | K-means | SC |
|---|---|---|
| SIFT | 0.39 | 0.43 |
| SURF | 0.44 | 0.37 |

Visualizations of the autoencoder and DC feature representations are depicted in figure 6.21. The plots have strong similarities to the t-SNE visualization created without using deep learning feature extraction shown in figure 6.20. This indicates that deep learning models have not adapted very well to the underlying data and cannot produce feature representations that clearly distinguish the seven categories. Furthermore, the supervised feature representation depicted in the same figure shows a seemingly much better data representation proving the potential of deep learning feature extraction in the AILARON domain. Therefore, the unsupervised models seem inadequate for adaption to the recent version of the AILARON data.


**Figure 6.21:** Low dimensional t-SNE visualizations on the AILARON data using unsupervised deep feature extraction. **Left image:** T-SNE visualization of the DC model. **Right image:** T-SNE visualization using the autoencoder model.

A similar conclusion can be drawn by looking at the classification results in table 6.14 using the feature extraction networks followed by a clustering algorithm. Albeit clearly improving the results compared to using traditional feature extraction models, the performance is not improved over the baseline clustering without deep feature extraction. An important observation is the relative higher NMI score compared to cluster accuracy. This is likely a result of the red class being clustered into two groups resulting in a lower accuracy. However, the assignments are not random, which is shown by the NMI metric.

**Table 6.14:** Classification results on AILARON-DB1 using different clustering algorithms

| Model | Accuracy | NMI |
|---|---|---|
| DC-K-Means | 0.58 | 0.66 |
| DC-SC | 0.63 | 0.69 |
| DC-BIRCH | 0.59 | 0.66 |
| DC-K-Means | 0.60 | 0.62 |
| AUTO-K-Means | 0.62 | 0.70 |
| AUTO-BIRCH | 0.65 | 0.69 |

The DC feature visualization in figure 6.22 provides a better insight into the representational capabilities of the network and some of the problems with the AILARON data set. Most of the ”bubble” samples (red class) are gathered in the lower-left corner. However, several of the ”oil” droplets (magenta class) and ”oily gas” (yellow class) are grouped together with the ”bubbles.” These samples look strikingly equal, suggesting human labeling errors or class characteristics that are indistinguishable for the untrained eye. The samples gathered at the bottom are mostly consisting of ”bubble” samples (red class), which look very different from the remaining class samples. It is thus unsurprising that this class is divided into two parts. The middle grouping consists of points belonging to a mix of varying classes. The ”faecal pellets” (light blue class), ”diatom chains” (green class), and ”copepod” (blue class) are somewhat separated, which shows that the model has learned some distinctions within the data. However, the group is dominated by the ”other” samples (black class), which, dependent on their characteristics, are placed together with the other classes. Finally, the rightmost two groups are mostly containing the same class objects showing to some extent the model capability at learning relevant features.

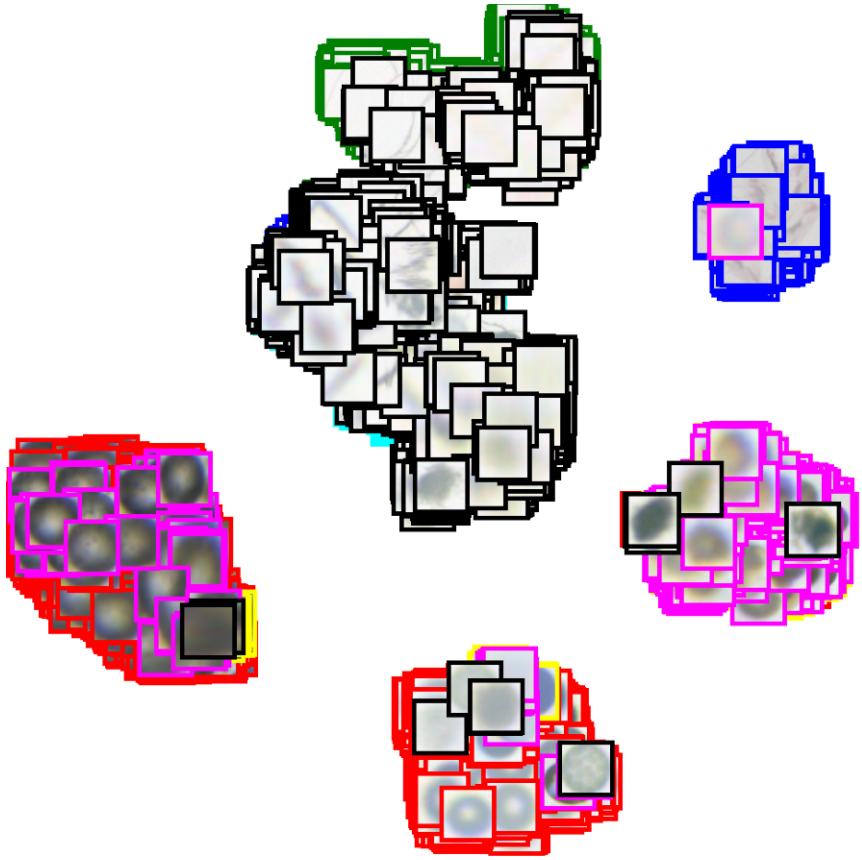


**Figure 6.22:** Low dimensional visualization of the learned representations using the DC feature extractor on the AILARON data. For each colored sample point, the corresponding test image is pictured.

### Summary and reflection

Experiment 3 explored the capabilities of the unsupervised deep learning models when adapted to the AILARON data. First, the baseline experiment proved that supervised deep learning slightly improves the classification results over clustering models that do not utilize deep feature extraction. However, the performance increase was only minor, indicating the difficulties of learning representations that fully distinguish the different classes. This was further confirmed by the traditional machine learning feature extraction algorithms, which worsened the classification results compared to the lower baseline. Furthermore, the experiment showed that the proposed unsupervised deep learning networks did not improve the classification results. However, the models did show some abilities to separate the data and did not severely under-perform compared to the clustering results when using features from a supervised deep learning network.

First and foremost, the experiment manifested the shortcomings and poor quality of the current AILARON data set. The data set is strongly dominated by blurry and undefined objects with extreme variations in feature characteristics. Furthermore, the unsupervised models seemed to reveal several sample misclassifications, which makes the performance results less trustworthy. Furthermore, the images are captured using different camera resolutions, which naturally makes the feature characteristics extremely varying within the same class species. Lastly, the round, blurry and related images of bubbles, oil droplets, and gas does not serve as a good representation of the framework's capability in detecting varying planktonic species.

Chapter 7

# Discussion

This chapter presents a discussion and reflection of several key areas and important findings presented in this thesis. First, a review of the unsupervised framework and its parts are presented. Then, the second part presents a discussion of the applicability of unsupervised learning applied to the planktonic domain and its challenges. Finally, the third section presents the perspectives and views of present unsupervised learning and its future.

## 7.1 Deep unsupervised feature extraction

The baseline results proved the inefficiency of clustering high dimensional data without first performing feature extraction. This was an expected result as the similarity measures in machine learning clustering algorithms are usually quite simple and unfit when the number of data dimensions are extensive. Therefore, deep feature extraction proved a vital step in the overall framework to map the image data into a lower-dimensional space that is more easily separable for machine learning classifiers. Naturally, the deep learning networks' low dimensional feature representations had slightly different properties and capability to preserve important image information. In the following sections, the three deep learning feature extraction methods' findings and results are elaborated and discussed.

### 7.1.1 Autoencoder

The rotation invariant autoencoder model was proposed as a continuation of the studies conducted in the specialization-project. In the specialization-project experiment, a consistent problem for all deep feature extraction models turned out to be the lack of rotation invariance were the same class objects with different orientations were split into separate groups. The change in architecture replacing conventional convolution layers with group convolutions proved very successful and resulted in more robust groupings of same class

species, which greatly improved feature representations and classification results.

Furthermore, the non-linear t-SNE visualization and the SC results showed that the learned feature representations provided by the model was not too different to those obtained by supervised deep learning models. This indicates that an autoencoder can learn relevant features for classification given the right architecture and architectural constraints, albeit trained for a different task. In accordance with Ulyanov et al. in [110] this illustrates that developing new deep learning architectures can be as important as finding new training methods to improve performance. However, the improvement is tied to the non-linear domain, and the autoencoder model did not provide equally good features for linear classification. Albeit the DEC [117] algorithm proving unsuccessful in this work, it might still be relevant to utilize approaches to slightly augment the feature space for better classification in future work. Another interesting finding was the models seemingly unchanged performance when new and previously unseen classes are introduced. In contrast to deep learning models trained over labeled data, training by reconstructing images seems to provide more general features of the data, which are useful for extracting information regardless of whether the model has been trained to recognize the object.

Autoencoder training usually results in image reconstructions, which resemble close copies of the input image. However, the reconstructions using the rotation invariant model did not give optimal results. This is likely a consequence of the rotation invariance, making it challenging for the model to retain information about object position. As a result, the decoder part seems to ”guess” the more likely placements of the object's main body. Still, failure to reconstruct the input images might indicate improved performance by switching to a more complex model. Unfortunately, the large increase in feature maps proved to be very memory expensive, making it impossible to increase the depth and complexity with the current computer specifications.

### 7.1.2 GAN

In recent years it has been an increased focus on generative models in unsupervised learning, and the deep convolutional GAN model tested in this thesis showed promising results. From a theoretical point of view, GAN models have gotten the most attention due to their excellent abilities at creating new and very realistic imitations of real-world data. This was also experienced in this thesis observing images that look utterly realistic to none experts. For the purpose of feature extraction, the capability to create realistic images from the generator part emphasizes an equal capability to distinguish images by the discriminator. However, the GAN network gave the least satisfying results of the three unsupervised deep learning methods and did not improve performance over the specialization-project models.

The t-SNE visualization showed that the model has similar problems with the same category objects with different rotations as the models from the specialization-project. This partially explains the bad classification results and emphasize the potential for increased performance by switching to a more complex or rotation invariant architecture. Unfortunately, the complexity of the discriminator part is heavily constrained by the generator part. Thus, increasing the discriminator capability resulted in non-convergence as the generator

is unable to find images that fool the discriminator.

In contrast to the autoencoder training method, the discriminator training can solve a binary classification problem that seems more related to the final task. Therefore, one might expect the learned feature representations of the discriminator to be more optimal for classification purposes. Naturally, due to the challenges mentioned above of GAN training, the performance was not spectacular. However, the PCA visualization seems slightly better than the autoencoder results, which indicate an improved representation of linear features.

As of yet, the model does not provide feature extraction results, which makes the model suitable in the proposed framework of this thesis. However, the field is still relatively new, and the model dynamics and training mechanics are still not perfectly understood by the research community [38]. The GAN model is still not ideal but shows interesting potential as a feature extraction model.

### 7.1.3 Deep Cluster

The recently proposed deep cluster approach is a method that is more specifically designed for learning good feature representations for feature extraction and subsequent classification. Unlike the other unsupervised training schemes, a great advantage is that the deep learning backbone can easily be replaced. Thus, it is possible to train a wide variety of backbone CNN networks with only minor modifications. With the rapid development of new supervised architectures, the future state of the art networks are therefore likely to improve the results also in the unsupervised domain.

In this thesis, the most successful backbone network proved to be the VGG16 network, which also yielded good results in the original DC paper [15]. Compared to the shallower 5-CONV architecture, the increased performance seems well connected with the increased complexity of adding model depth. However, this finding is not supported by the ResNet model, which did not provide very satisfying results. However, compared to the other networks, the ResNet design is quite different. Therefore, a suggestion for the poor performance is that the networks increased learning speed and larger gradients due to the skip connections makes the model more vulnerable to a training scheme with rapidly changing labels and several misclassifications.

Using the VGG16 as backbone, the t-SNE visualization and the SC classification results show that the learned feature representations provided by the model share strong connections with those obtained using supervised deep learning. This prove that the DC approach is clearly useful for learning feature representations unsupervised. However, the good performance is not tied to the non-linear domain, and the PCA visualizations and k-means classification results shows that the model is able to find image representations which separates the data in the linear domain. According to Bengio et al. [7] a quality of good feature representations are their simplicity and typically linear dependencies. In this regard, the learned DC representations are seemingly superior to the other unsupervised deep learning methods.

The DC models ability to map the high dimensional data into a more linear space seems related to the aforementioned observation by Hartono et al. [47] concerning the influence labels have on the learned feature representation. Labeled training might therefore be an important part of representational learning for classification. Unsupervised training over pseudo-labels produced by a machine learning clustering algorithm seems an effective way of imitating supervised deep learning. Furthermore, as the pseudo-labels provided in this work shared low resemblance with the ground truth labels and still gave good results, it is likely that new and more accurate classification can further improve the results. However, the labeled training's negative side effect seems to be the model's inability to adapt to new and unseen classes.

## 7.2 Unsupervised clustering

Albeit obtaining unsupervised feature representations that share clear similarities with the ones obtained using supervised learning, the final classification results are far from those that can be obtained by supervised classifiers. The clustering part seems, therefore, to be the weak point of the framework. This seems reasonable as clustering, albeit used for classification, is a partly distinct task for the purpose of identifying object similarities and, based on these characteristics, group the similar objects together. In comparison, classification is often used to separate similar type objects based on small distinctions, which makes it necessary to know the ground truth labels. As an example, the framework is clearly capable of obtaining good results when performing over a limited number of quite distinct plankton classes. In comparison, the accuracy is quite low when classifying over the full Kaggle data set containing several more categories. However, this is not very surprising as the categories separate between, for instance, copepods with different types of antenna. Arguably such specific distinctions of copepod species are impossible to perform without knowledge of the expected ground truth classes and seem like an unreasonable difficult task.

Of the tested clustering algorithms, the SC method achieved the most accurate and more stable performance over the Kaggle data. The improved performance over the other models is likely due to the feature extractions containing non-linear properties. Unfortunately, the method is transductive, meaning it cannot generalize to new unseen data. This is a severe drawback as the prediction time is very time consuming, and the model requires a relatively large set of training samples to compare against the unseen data. These issues are not experienced in the other algorithms making the BIRCH model, which achieved the second-best results, a good alternative.

## 7.3 Application in the plankton domain

The automatic sampling of planktonic and other microscopic particles from the recently developed underwater camera systems provides new and unexplored possibilities within the oceanic domain. However, classifying and assessing the increasing quantities of planktonic images using human domain level experts are clearly impractical, putting pressure on

finding ways to automate this work. The experiments conducted in this thesis show that an unsupervised framework is, to some extent, able to group and correctly classify different groups of planktonic species. However, the framework is only precise when performing on a small number of plankton categories, and a significant decrease in accuracy is observed when trying to detect several more classes. Furthermore, the framework results are still far behind those obtained using supervised deep learning.

The benefits of introducing machine learning algorithms to the plankton domain are, among others, faster, continuous, and much less labor-intensive predictions of planktonic species. However, the cost is likely to decrease the accuracy and potential inability to assign the samples based on specific taxonomic properties, requiring close examination under the microscope. In a similar fashion, the introduction of unsupervised algorithms avoids the need to build a training data set based on labeled data, which further decreases the workload on human operators. Based on this thesis's experiments, the cost is a further decrease in accuracy and reduction in the ability to differentiate species based on small distinctions. Therefore, the question to decide in each particular case is if a decrease in performance is worth sacrificing for the great reduction in labor intensive work.

1. **Integrating the model onto the AUV:** A framework consisting of the DC feature extraction model combined with the BIRCH clustering algorithm is able to work under the real-time and strict hardware constraints related to the in-situ classification. Despite the impressive increase in accuracy compared to the specialization-project models, the model does not provide adequately robust and accurate predictions. Thus, it is unlikely that an unsupervised approach would provide satisfying results on its own, making it more recommendable to utilize the framework in assistance with a supervised classifier. However, how these approaches should be combined remains a question for further work.

2. **Utilize unsupervised learning in the labeling process:** The second applicable task is to utilize unsupervised deep learning as a tool to increase the labeling speed and soften the labeling burden of expert biologists in the work at creating a data set for supervised learning. For this purpose, an unsupervised framework could be especially useful in providing a quick grouping and division of the data set samples, which can then be assessed more closely by the domain level experts. For this purpose, a framework consisting of the autoencoder feature extraction model and the SC algorithm is much more applicable as there are no real-time constraints. Apart from its high classification score, a great benefit of this setup is its ability to adapt to new and unseen classes, which is a likely scenario to endure when creating a new data set.

### Data set shape

An important but less discussed topic in image classification is how to treat images with varying sizes and shapes. This is a particular problem with the existing planktonic data sets as the images usually vary in quite large size ranges. For instance, for the *Kaggle* dataset, described in section 2.4.5, the smallest images are around $30 \times 30$ pixels while the

largest are over $400 \times 400$. The most common approach for handling different size inputs is to reshape the data to a fixed input. In accordance with similar work on plankton data [99], the data was reshaped to a fixed size of $64 \times 64$ in this thesis. However, reshaping to a fixed size is problematic since it is easy to clutter and destroy important information in the images. The choice of image dimension should, therefore, be well-founded.

To gain a deeper insight into the problem, the deep learning models were tested over several different image sizes, but without observing noticeable changes in the model performance. Furthermore, some minor work was conducted using tools such as zero-padding to enlarge the images to the largest data set sample without changing their properties and relative size. However, this resulted in an enormous decrease in training speed without a noticeable increase in accuracy. Still, for future experiments, this should be further examined.

#### Data set size and class imbalance

An extensive training data set is generally associated with good machine learning performance stressing the importance of increasing the data set. However, carelessly adding images to the data set might turn out problematic as increasing class imbalance can greatly affect the model performance. In the plankton domain, these issues are especially problematic as most objects are impossible to categorize and thus ending up in a collective majority group of ”mix,” ”other,” or ”unknown” species while the number of classifiable objects are much smaller. The already challenging task of learning the class specifics is thus increased as the model must also distinguish these samples from the highly varying mix of unknown samples. This problem seems inherent in the AILARON data as the model learns a representation where several small classes are grouped together with the larger ”other” class. This problem might prove a major obstacle when applying unsupervised models over planktonic data with unknown diversity and likely a high number of unidentifiable objects.

## 7.4 General view of unsupervised deep learning

Unsupervised deep learning played an important part in the revival and renewed interest in deep learning at the beginning of the new century. Since unsupervised learning shares a strong resemblance to human and animal learning, the domain has naturally been associated with great expectations without achieving the anticipated results. As opposed to the much more successful supervised deep learning domain, the primary challenge is to find a target function and training method that relates to the task at hand. Therefore, deep unsupervised models are usually trained on a partially different task with the hope that they can obtain useful properties to solve the actual objective. As experienced in this thesis, the goal of finding a useful unsupervised classification model turned out difficult as the task of classification is more closely associated with supervised learning. The acquired unsupervised models, albeit capable of learning useful properties, were not trained explicitly to categorize different class objects and proved only sub-optimal for the prediction task.

Another prevalent problem for deep learning models is how to determine the quality of the learned model representations and the subsequent classification capabilities. Because of the challenge of getting insight by studying the internal structure of neural networks, other metrics are usually used. In this work, such metrics included clustering accuracy, and visual inspection of the representation and activation maps. However, these results' interpretation is mainly based on what the researcher assumes is the best solution. It is thus not given that the best deep learning model corresponds to the assumingly best metric results. This was also evident in this thesis as the performance over the different metrics did not always match, making it infeasible to rely on a single metric to determine the true model quality.

Furthermore, an important question is whether such metrics are accessible in the true model environment. Of course, for an unsupervised task, one would expect it to be no ground truth labels making metrics based on classification performance infeasible. Thus, performance might be based entirely on different visualization tools, which might not provide the best standalone results. Furthermore, the utilization of performance metrics for the purpose of model tuning and validation can be seen as inferring some form of supervision, which destroys the purpose of self-learning.

Due to the challenges and apparent difficulties within the unsupervised domain, much of the recent work [15, 52, 116, 125], is shifting towards the semi-supervised domain. A typical approach combines unsupervised feature extraction trained over a vast amount of unlabeled data and validates its performance using a supervised machine learning classifier, which is fine-tuned over a small amount of labeled data. In this way, it is possible to obtain excellent results with only minimum labeling effort.

Chapter 8

# Conclusion

This thesis assesses the building blocks that can contribute and form an unsupervised framework for plankton detection and classification. The work resulted in a proposed framework consisting of three components; pre-processing, feature extraction and classification. The framework is highly modular making it possible to test a wide variety of network architectures. This made it encouraging to implement and validate different unsupervised methods to improve the results from the specialization-project and find the best combination of components.

For feature extraction three different unsupervised deep learning methods was implemented and tested. The rotation invariant autoencoder model was proposed as an improvement of the less robust autoencoders tested in the specialization-project. In addition to the improved processing of similar species of different rotation the model learned greatly improved feature representations resulting in a classification accuracy of 93% using SC over the Kaggle-DB1 data set. The DC method, using a VGG16 backbone network, provided very similar classification accuracy using the SC algorithm scoring 92% over the same data set. However, the DC results are much better when comparing the methods using linear metrics making the method more ideal for linear classification. The GAN model, albeit able to generate superficially authentic plankton images, was not able to improve the result over those obtained in the specialization-project. Still, the method showed interesting potential as a feature extraction model and is likely to improve its performance when models of higher complexity are available.

Several different clustering algorithms were then tested for the purpose of classification. The best classification accuracy was obtained using SC regardless of feature extraction network. Unfortunately the algorithm is very time consuming making it less ideal in real-time applications when speed is crucial. Thus, the BIRCH algorithm might turn out a better option providing much faster predictions at the cost of slightly lower accuracy. However,

the models only provides robust results when the data contains only a few species and failed considerably when predicting several more categories.

Overall, the proposed framework shows interesting potential and might be a valuable classification tool when the size of the data set is small. In the current process of improving the AILARON data set, the framework thus seems a practical tool that can provide a rough first assignment of the planktonic species. Therefore, the main application appears to be a classification tool that helps domain level experts speed up the labeling process.

Chapter 9

# Future Work

This thesis covers a broad spectrum of machine learning research and state of the art applications within the unsupervised deep learning domain. However, with the limited time available, several more research questions and directions for future work still exist. This chapter summarizes some of the major tasks which must be assessed in future work.

1. This thesis has explored the realm of unsupervised learning, provided insight into the model capabilities, and studied their applicability to the planktonic domain. Still, the proposed framework is not yet adopted into the AILARON real-time classification pipeline, and the framework's area of use is, therefore, yet to be explored. First, the experiments showed that the supervised neural networks are still superior in regards to classification performance, making a standalone unsupervised classifier a less ideal choice. However, no further investigation of the classification when combining the two approaches has been conducted, making it possible for the unsupervised models to assist the existing supervised framework. Furthermore, unsupervised models have other advantages that might prove very useful. A much-referred problem in plankton classification is data set drift [40] which refers to the testing conditions varying over time. Thus, it is paramount to obtain an algorithm capable of adjusting to the varying conditions and adapting to new and unseen plankton categories.

2. The current choice of employing a supervised classification algorithm as the prediction model in the AILARON pipeline is first and foremost problematic due to the requirement of extensive labeling effort. The unsupervised framework has already shown a promising potential at grouping similar species and adapt to new and unseen classes making it ideal for assisting human experts in the annotation process. How such an annotation tool should work and how the unsupervised resources are best combined with human expertise needs further research.

3. A problem which was not further assessed in this thesis was deciding the number of categories in the existing data set. Following the state of the art work in the unsupervised domain [16, 46, 117], the number of classes was regarded as priorly known and set to a fixed number before training. However, in a fully unsupervised setting, such information might not be available making it difficult to choose an appropriate number of clusters. Therefore, a further improvement of the proposed unsupervised framework is to explore and implement an algorithm that can propose a likely number of categories existing in the input data.

4. The most emphasized model structure in this work consisted of a deep learning feature extraction model and a separate clustering algorithm. A drawback of this structure is that the classification part cannot influence the feature extraction model's feature representations. This problem might result in a less satisfying result. Albeit an apparent drawback, reasonable solutions are challenging to obtain in the unsupervised domain. The DEC approach was tested in this thesis but provided unsatisfying results. Still, there exists several similar techniques such as the DAC [16], and the DCEC [46] algorithm, which might provide better results over the plankton data.

5. One of the most significant advantages of unsupervised learning is the capability of increasing performance by utilizing large amounts of unlabeled training data. The current version of the AILARON-DB2 data set only contains a few classes with few dominant feature characteristics, making it difficult to learn a suitable data representation and extract useful features. Therefore, a focus point for future development should be to gather several more training samples from a wider domain of plankton classes.

# Bibliography


[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] V. Agarwal. Research on data preprocessing and categorization technique for smartphone review analysis. *International Journal of Computer Applications*, 975:8887, 2015.

[3] C. C. Aggarwal et al. *Neural networks and deep learning*. Springer, 2018.

[4] J. Almotiri, K. Elleithy, and A. Elleithy. Comparison of autoencoder and principal component analysis followed by neural network for e-learning using handwritten recognition. In *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pages 1–5. IEEE, 2017.

[5] A. I. Awad and M. Hassaballah. Image feature detectors and descriptors. *Studies in Computational Intelligence. Springer International Publishing, Cham*, 2016.

[6] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[7] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[8] A. Bhardwaj, W. Di, and J. Wei. *Deep Learning Essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling*. Packt Publishing Ltd, 2018.

[9] F. Bianchi, F. Acri, F. B. Aubry, A. Berton, A. Boldrin, E. Camatti, D. Cassin, and A. Comaschi. Can plankton communities be considered as bio-indicators of water quality in the lagoon of venice? *Marine Pollution Bulletin*, 46(8):964–971, 2003.

[10] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[11] M. A. Boden. *AI: Its nature and future*. Oxford University Press, 2016.

[12] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[13] A. S. Brierley. Plankton. *Current Biology*, 27(11):R478–R483, 2017.

[14] A. Byerly, T. Kalganova, and I. Dear. A branching and merging convolutional network with homogeneous filter capsules. *arXiv preprint arXiv:2001.09136*, 2020.

[15] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

[16] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan. Deep adaptive image clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 5879–5887, 2017.

[17] F. Chollet et al. Keras. `https://keras.io`, 2015.

[18] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[19] T. Cohen and M. Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999, 2016.

[20] R. K. Cowen and C. M. Guigand. In situ ichthyoplankton imaging system (isiis): system design and preliminary results. *Limnology and Oceanography: Methods*, 6(2):126–132, 2008.

[21] R. K. Cowen, S. Sponaugle, K. L. Robinson, J. Luo, and C. Guigand. Planktonset 1.0: Plankton imagery data collected from f.g. walton smith in straits of florida from 2014-06-03 to 2014-06-06 and used in the 2015 national data science bowl (nodc accession 0127422). national oceanographic data center, noaa. dataset. 2015.

[22] A. Creswell, K. Arulkumaran, and A. A. Bharath. On denoising autoencoders trained to minimise binary cross-entropy. *arXiv preprint arXiv:1708.08487*, 2017.

[23] B. C. Csáji et al. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24(48):7, 2001.

[24] P. F. Culverhouse, R. Williams, B. Reguera, V. Herry, and S. González-Gil. Do experts make mistakes? a comparison of human and machine indentification of dinoflagellates. *Marine ecology progress series*, 247:17–25, 2003.

[25] E. Davies. https://github.com/emlynjdavies/PySilCam/wiki. [Online; accessed November 11, 2020].

[26] E. J. Davies, P. J. Brandvik, F. Leirvik, and R. Nepstad. The use of wide-band transmittance imaging to size and classify suspended particulate matter in seawater. *Marine pollution bulletin*, 115(1-2):105–114, 2017.

[27] E. J. Davies and R. Nepstad. In situ characterisation of complex suspended particulates surrounding an active submarine tailings placement site in a norwegian fjord. *Regional Studies in Marine Science*, 16:198–207, 2017.

[28] E. R. Davies. *Computer vision: principles, algorithms, applications, learning*. Academic Press, 2017.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[30] L. Deng and D. Yu. Deep learning: methods and applications. *Foundations and trends in signal processing*, 7(3–4):197–387, 2014.

[31] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[32] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[33] A. Elgammal, B. Liu, M. Elhoseiny, and M. Mazzone. Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms. *arXiv preprint arXiv:1706.07068*, 2017.

[34] A. Fabre, L. Ortega, S. Mendez, and A. Martinez. Climate shift triggers shellfish harvesting bans in uruguay (south-west atlantic ocean). *Marine and fresh-water harmful algae*, pages 18–20, 2016.

[35] A. A. Freitas. Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter*, 15(1):1–10, 2014.

[36] H. Gao, H. Yuan, Z. Wang, and S. Ji. Pixel transposed convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 42(5):1218–1227, 2019.

[37] E. S. Gedraite and M. Hadad. Investigation on the effect of a gaussian blur in image filtering and segmentation. In *Proceedings ELMAR-2011*, pages 393–396. IEEE, 2011.

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

[39] J. Gong. Refining a deep learning model for object detection. https://blogs.sas.com/content/subconsciousmusings/2019/03/05/refining-a-deep-learning-model-for-object-detection/, 2019. [Online; accessed May 19, 2020].

[40] P. González, E. Álvarez, J. Díez, Á. López-Urrutia, and J. J. del Coz. Validation methods for plankton image classification systems. *Limnology and Oceanography: Methods*, 15(3):221–237, 2017.

[41] P. González, A. Castaño, E. E. Peacock, J. Díez, J. J. Del Coz, and H. M. Sosik. Automatic plankton quantification using deep features. *Journal of Plankton Research*, 41(4):449–463, 2019.

[42] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[43] I. Goodfellow, A. Courville, and Y. Bengio. *Deep learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 2017.

[44] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[45] G. Gorsky, M. D. Ohman, M. Picheral, S. Gasparini, L. Stemmann, J.-B. Romagnan, A. Cawood, S. Pesant, C. García-Comas, and F. Prejger. Digital zooplankton image analysis using the zooscan integrated system. *Journal of plankton research*, 32(3):285–303, 2010.

[46] X. Guo, X. Liu, E. Zhu, and J. Yin. Deep clustering with convolutional autoencoders. In *International conference on neural information processing*, pages 373–382. Springer, 2017.

[47] P. Hartono. Mixing autoencoder with classifier: conceptual data visualization. *arXiv preprint arXiv:1912.01137*, 2019.

[48] G. C. Hays, A. J. Richardson, and C. Robinson. Climate change and marine plankton. *Trends in ecology & evolution*, 20(6):337–344, 2005.

[49] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[50] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[51] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[52] J. Huang, Q. Dong, S. Gong, and X. Zhu. Unsupervised deep learning by neighbourhood discovery. *arXiv preprint arXiv:1904.11567*, 2019.

[53] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[54] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[55] A. Kaplan and M. Haenlein. Siri, siri, in my hand: Who's the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence. *Business Horizons*, 62(1):15–25, 2019.

[56] E. Karami, S. Prasad, and M. Shehata. Image matching using sift, surf, brief and orb: performance comparison for distorted images. *arXiv preprint arXiv:1710.02726*, 2017.

[57] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[58] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[59] M. Kolla and T. Venugopal. Semantic image clustering with global average pooled deep convolutional autoencoder. *HELIX*, 8(4):3561–3566, 2018.

[60] A. Krizhevsky and G. Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010.

[61] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[62] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[63] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[64] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[65] D. Kuzminykh, D. Polykovskiy, and A. Zhebrak. Extracting invariant features from images using an equivariant autoencoder. In *Asian Conference on Machine Learning*, pages 438–453, 2018.

[66] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[67] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[68] H. Lee, M. Park, and J. Kim. Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning. In *2016 IEEE international conference on image processing (ICIP)*, pages 3713–3717. IEEE, 2016.

[69] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015.

[70] M. Leordeanu. *Unsupervised Learning in Space and Time*. Springer, 2020.

[71] X. Li and Z. Cui. Deep residual networks for plankton classification. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–4. IEEE, 2016.

[72] Q. Liu and Y. Wu. *Supervised Learning*. Springer US, Boston, MA, 2012.

[73] S. Liu and W. Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian conference on pattern recognition (ACPR)*, pages 730–734. IEEE, 2015.

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

[75] L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis. Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*, 2019.

[76] C. H. Lucas, S. Gelcich, and S.-I. Uye. Living with jellyfish: management and adaptation strategies. In *Jellyfish blooms*, pages 129–150. Springer, 2014.

[77] A. Lumini and L. Nanni. Ocean ecosystems plankton classification. In *Recent Advances in Computer Vision*, pages 261–280. Springer, 2019.

[78] J. Y. Luo, J.-O. Irisson, B. Graham, C. Guigand, A. Sarafraz, C. Mader, and R. K. Cowen. Automated plankton image analysis using convolutional neural networks. *Limnology and Oceanography: Methods*, 16(12):814–827, 2018.

[79] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[80] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[81] G. Mariani, F. Scheidegger, R. Istrate, C. Bekas, and C. Malossi. Bagan: Data augmentation with balancing gan. *arXiv preprint arXiv:1803.09655*, 2018.

[82] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

[83] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[84] NVIDIA, P. Vingelmann, and F. H. Fitzek. Cuda, release: 10.2.89, 2020.

[85] M. D. Ohman, R. E. Davis, J. T. Sherman, K. R. Grindley, B. M. Whitmore, C. F. Nickels, and J. S. Ellen. Zooglider: an autonomous vehicle for optical and acoustic sensing of zooplankton. *Limnology and Oceanography: Methods*, 17(1):69–86, 2019.

[86] R. J. Olson and H. M. Sosik. A submersible imaging-in-flow instrument to analyze nano-and microplankton: Imaging flowcytobot. *Limnology and Oceanography: Methods*, 5(6):195–203, 2007.

[87] E. C. Orenstein, O. Beijbom, E. E. Peacock, and H. M. Sosik. Whoi-plankton-a large scale fine grained visual recognition benchmark dataset for plankton classification. *arXiv preprint arXiv:1510.00745*, 2015.

[88] V. P. Pastore, T. G. Zimmerman, S. K. Biswas, and S. Bianco. Annotation-free learning of plankton for classification and anomaly detection. *Scientific reports*, 10(1):1–15, 2020.

[89] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[90] O. Py, H. Hong, and S. Zhongzhi. Plankton classification with deep convolutional neural networks. In *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*, pages 132–136. IEEE, 2016.

[91] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[92] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[93] R. Reed and R. J. MarksII. *Neural smithing: supervised learning in feedforward artificial neural networks*. Mit Press, 1999.

[94] J. Rocca. Understanding variational autoencoders (vaes). https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73. [Online; accessed October 10, 2020].

[95] F. C. M. Rodrigues, N. S. Hirata, A. A. Abello, T. Leandro, D. La Cruz, R. M. Lopes, and R. Hirata Jr. Evaluation of transfer learning scenarios in plankton image classification. In *VISIGRAPP (5: VISAPP)*, pages 359–366, 2018.

[96] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[97] S. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Prentice Hall series in artificial intelligence. Pearson Education, 2016.

[98] A. Saad, E. Davies, and A. Stahl. Recent advances in visual sensing and machine learning techniques for in-situ plankton-taxa classification. presented at Ocean Sciences Meeting 2020, San Diego, CA, 16-21 Feb., 2020. 636384.

[99] A. Saad, A. Stahl, A. Våge, E. Davies, T. Nordam, N. Aberle, M. Ludvigsend, G. Johnsen, J. Sousa, and K. Rajan. Advancing ocean observation with an ai-driven mobile robotic explorer. *Oceanography*, 33:42–51, 2020.

[100] E. Salvesen. Unsupervised methods for in-situclassification of plankton taxa, 2019. This file is attached to the thesis document as a supplementary material.

[101] E. Salvesen, A. Saad, and A. Stahl. Robust methods of unsupervised clustering to discover new planktonic species in-situ. In *OCEANS 2020/IEEE SINGAPORE*. IEEE, 2020.

[102] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[103] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[104] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[105] H. M. Sosik and R. J. Olson. Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. *Limnology and Oceanography: Methods*, 5(6):204–216, 2007.

[106] I. Suthers, D. Rissik, and A. Richardson. *Plankton: A guide to their ecology and monitoring for water quality*. CSIRO publishing, 2019.

[107] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[108] T. Taulli and M. Oni. *Artificial Intelligence Basics*. Springer, 2019.

[109] R. Thakur, R. Jindal, U. B. Singh, and A. Ahluwalia. Plankton diversity and water quality assessment of three freshwater lakes of mandi (himachal pradesh, india) with special reference to planktonic indicators. *Environmental monitoring and assessment*, 185(10):8355–8373, 2013.

[110] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454, 2018.

[111] H. Valpola. From neural pca to deep unsupervised learning. In *Advances in independent component analysis and learning machines*, pages 143–171. Elsevier, 2015.

[112] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling. Rotation equivariant cnns for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention*, pages 210–218. Springer, 2018.

[113] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[114] C. Wang, Z. Yu, H. Zheng, N. Wang, and B. Zheng. Cgan-plankton: towards large-scale imbalanced class generation and fine-grained classification. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 855–859. IEEE, 2017.

[115] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[116] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018.

[117] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[118] R. Xu and D. Wunsch. *Clustering*, volume 10. John Wiley & Sons, 2008.

[119] J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016.

[120] Y. Yazici, C.-S. Foo, S. Winkler, K.-H. Yap, and V. Chandrasekhar. Empirical analysis of overfitting and mode drop in gan training. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1651–1655. IEEE, 2020.

[121] S. R. Yoshida. *Computer vision*. Computer science, technology and applications. Nova Science Publishers, New York, 2011.

[122] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114, 1996.

[123] H. Zheng, R. Wang, Z. Yu, N. Wang, Z. Gu, and B. Zheng. Automatic plankton image classification combining multiple view features via multiple kernel learning. *BMC bioinformatics*, 18(16):570, 2017.

[124] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[125] C. Zhuang, A. L. Zhai, and D. Yamins. Local aggregation for unsupervised learning of visual embeddings. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6002–6012, 2019.

[126] D. Ziou, S. Tabbone, et al. Edge detection techniques-an overview. *Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii*, 8:537–559, 1998.