COMPARING TRANSFER-LEARNING AND SELF-SUPERVISED LEARNING FOR OCEAN FLOOR IMAGE CLASSIFICATION

by

Shakhboz Abdulazizov

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
December 2022

I dedicate my thesis to my parents Yulchiyev Mirvahid and Akhmadaliyeva Malokhat who are the main reason behind all my achievements with their unconditional support.

Table of Contents

List of Tables

List of Figures

Abstract

Benthic habitat mapping is a process of labeling substrates, plants, and animals on the seafloor. Mapping of the benthic habitat is crucial to monitor changes happening due to natural and human-related activities. Annotation of the large amount of data produced with underwater camera systems requires automation. A large dataset of around ten million ocean floor images (BenthicNet) was recently compiled as a part of the BEcoME Project (Benthic Ecosystem Mapping & Engagement). This thesis discusses the development and specific challenges of a classification system for this dataset. We specifically discuss the importance of careful training and test set partitioning. We further evaluate the performance of pretrained models on ImageNet by supervised learning versus those by self-supervised learning. We show that transfer learning from ImageNet enables good performance comparable with versions that start from self-supervised representations from the BenthicNet dataset.

Acknowledgements

I would like to express my heartfelt gratitude and appreciation to my supervisor Dr. Thomas Trappenberg for his guidance and support throughout my master's degree and thesis work. His valuable insights, constructive feedback, and encouragement have been instrumental in the successful completion of my degree.

I want to acknowledge Dr. Scott Lowe for his valuable advice during my research and Isaac Xu for his help in training a model and providing figure 7.6 in the thesis.

I also thank my friend Muhammadjon Toshpulatov for all the support during my studies.

Chapter 1

Introduction

A benthic habitat is an underwater environment on the seafloor that sustains a particular community of plants and animals. A process of labeling or classifying biotic or abiotic elements on the seafloor is considered benthic habitat mapping. Habitat mapping is crucial to monitor ongoing changes in seafloor habitat posed by natural (e.g. a shifting climate) or human-related (e.g. commercial fishing activities) factors. Besides mapping plays a crucial role in creating policies to address these changes. The collection of data representing all habitat features in the seafloor area is important for accurate habitat mapping and advancement in data gathering techniques is helping this with automating and, hence, speeding up the process. The first way of collecting benthic data for seafloor mapping was physical sample collection which limits the size of data to be collected due to the speed of sample collection. Furthermore, physical samples are resource expensive to store. On the other hand, benthic imagery data in the form of images and videos are easier to gather and store. Imagery data can be collected by manual on-site (e.g., snorkeling, SCUBA) or surface(e.g., drop camera) deployment. The introduction of automated and remote underwater vehicles increased data collection speed and volume significantly as they don't require direct human supervision for data collection. However, the speed of manual classification, annotation, and labeling is a bottleneck to the usage of all collected data for training benthic habitat mapping models which creates a need to automate the process.

Machine learning has been proven to be successful in automating manual tasks by mapping the input data to the target output by finding numerical correlations between them. Especially deep learning can be used to build complex models to solve difficult tasks and has been applied to many image-processing tasks like classification [19, 36, 39], segmentation [18,19,32], and generation [22,34,42]. Deep learning models require a large amount of data to train but the model trained on one task can be retrained for another task with fewer samples which is called transfer learning. Transfer learning is possible due to the fact that earlier layers of models are responsible for learning simple features in the image like edges which is common to all image-related tasks. In computer vision, transfer learning is usually done on the models which are pretrained on the ImageNet-1k dataset which consists of 1.28 million natural images with 1000 classes [33].

Some previous works on benthic habitat mapping has been done with smaller local datasets and smaller models [3, 8, 10, 12, 15, 30, 35]. A big dataset for training deep learning models was not available as small labeled datasets are scattered around the world in different research groups and they don't have a common labeling scheme. When we have a limited number of labeled benthic data, we can use the benefits of transfer learning to train a supervised model which generalizes well. As the benthic images are natural images, transfer learning [20] from a model pretrained on ImageNet should help and our experiments show it does. On the other hand, benthic images differ from ImageNet natural images in terms of lighting and clarity which questions the performance of transfer learning from ImageNet on benthic image classification to be the best possible.

Considering labeled seafloor data to be a small portion of all available benthic data, it is reasonable to believe a model trained with all available data could perform better than transfer learning from ImageNet. The self-supervised learning methods [5, 6, 14, 16, 17] use unlabelled data to train a model with a pretext task that is created from the input image so that learning more task-independent representations. Training a self-supervised learning method on all available data and using transfer learning to retrain or fine-tune the model with available labeled data should result in better performance on unseen data.

There has been some work on benthic habitat mapping mainly focusing on substrate and biota classification on specific datasets. A limiting factor for training large-scale models that can generalize well is the lack of benthic imagery data. Using the BenthicNet dataset [27] which has 10 million images and modern self-supervised learning (SSL) methods, we aim to train a model that generalizes well across datasets around the world.

When evaluating the model's performance on unseen benthic data it is crucial to choose the right test data for the desired application purpose. A common way of choosing test data in image classification tasks is randomly picking test images for each class. However, we found that two geographically close seafloor images are very visually similar as the habitat is spread along huge areas on the seabed. Having one of the images in the training and the other one in a test partition may question the real generalization ability of the model on unseen data in terms of what one wants to use the trained model for. For example, if one wants to annotate images from a geographical area where they have some labeled data from to train a model, it is acceptable to use randomly picked images as test data that we refer to as test-same partition through the thesis. Nevertheless, if one wants to annotate images from a location through a model trained on data from different locations the generalization evaluation through random test data may mislead. Because, test data can include an image that is spatially next to an image used for training, biasing the real performance of the model in test data. A better approach for this use case is putting geographically close images in one partition avoiding bias in model's performance on unseen data that we call a test-other partition through the thesis. Before compiling the big BenthicNet dataset, we ran experiments on a smaller dataset to analyze the performance gap for different test partitions. Through these experiments, we found a noticeable difference in model performance between randomly and spatially chosen test partitions suggesting the importance of proper test data aligning our intentions.

Despite being similar to natural ImageNet images with respect to color and other features, benthic images differ from natural ImagenNet images in lighting conditions, surrounding water clarity, and so on. This arises the question of whether transfer learning from ImageNet is useful. To answer this question we run experiments on the transferability of learned representations from ImageNet on benthic imagery and found that the learned representations from ImageNet are beneficial for benthic imagery as seafloor images are also natural images. The experiments showed that when transfer learning from ImageNet it is very important to train a new classifier layer before fine-tuning the whole network because fine-tuning from the beginning can propagate big loss due to random initial weights on the last layer that can destroy the previously learned representations.

Furthermore, we compare the benefit of learned representations from supervised learning and self-supervised learning on ImageNet data. As SSL tries to learn task-independent features on imagery, we expect it to generalize better. However, we found they perform similarly in our classification task. As the labeled data is a small part of the whole dataset we have, we want to take advantage of unlabelled data for training the habitat mapping model. For this purpose, we trained the SSL model on BenthicNet data and compared it to learned representations from ImageNet. Results show the learned representations from ImageNet are better in our classification task setup. The potential reason is the classification task we created is not complex enough to take advantage of SSL pretraining. It requires further investigation by creating a more complicated task hierarchical classification task in the whole labeled dataset.

Chapter 2

Background

2.1 Benthic image classification

There has been a number of studies that emphasized computer vision techniques to apply to benthic habitat classification tasks before the deep learning era. For example, Shihavuddin et al [35] proposed an image classification scheme that uses various combinations of feature descriptors(e.g., completed local binary pattern, grey level co-occurrence matrix, Gabor filter response) and machine learning classifiers such as k-nearest neighbors, neural networks, and support vector machines (SVMs). The authors evaluated their method on three benthic datasets: EILAT, Rosenstiel School of Marine and Atmospheric Sciences (RSMAS), and Moorea labeled corals (MLC) [4]. The results showed using a combination of different feature extraction methods and classifiers outperformed using a single method with at least 5% better accuracy in all datasets. Gauci et al. [12] used three red-green-blue channels and three LAB color dimensions as features to classify each pixel as either sand or maerl. They used three machine learning algorithms random forest, neural network, and classification trees to classify images captured by cameras mounted on remotely operated vehicles (ROVs). Each applied algorithm achieved promising classification results. Another study by Raj and Murugan [30] used bagging of features descriptor together with the SVM algorithm to classify 11 000 images, captured by a camera on an ROV device, into seven benthic classes achieving 93% overall accuracy.

The introduction of convolutional neural networks (CNNs) has revolutionized computer vision. As convolutions enabled learning spatial correlations between pixels in the image it became the dominant architecture for computer vision tasks. CNN has been successfully applied to many computer vision tasks such as object detection [13], pose estimation [40], classification [21], segmentation [26], object tracking [28], denoising [31], and super-resolution [9]. The development of CNN architecture such as AlexNet [24], VGGNet [36], ResNet [19], and GoogLeNet [38] enabled state-of-the-art classification accuracy results on different computer vision problems.

Elawady [10] used CNNs for coral classification first. The author first enhances images used in the study by correction and smoothing filtering. Then they train a LeNet-5 CNN with input which consists of three color channels from the images and two layers of texture and shape descriptors. Two datasets were used to evaluate the study results: the MLC dataset with 2 000 images belonging to nine classes and the Atlantic Deep-Sea dataset with 55 images from five classes. The model achieves an overall accuracy of 55%. Another study by Asma et al. [3] proposed a CNN model for classifying coral images as damaged and healthy. The authors collected 1 200 images from different online sources and trained the model with 90% of the images. They achieved 95% classification accuracy on the remaining 10% of test data. A study by Gómez-Ríos et al. compared several CNN architectures on the classification task with two different coral texture datasets [15]. Using a ResNet-50 architecture, they achieved an accuracy of 97.85% after 500 epochs. Diegues et al. presented an automated approach for habitat mapping by collecting images by AUVs and classifying them with CNNs [8]. Their model achieved an accuracy of 85.1%.

All aforementioned works achieved high accuracy results but were trained and evaluated on datasets belonging to some local geographical area. To the best of our knowledge, the test partitions are chosen randomly which leaves spatial partitioning unexplored. The quality of learned representation by SSL on ImageNet for benthic image classification was not explored either. Novel self-supervised methods were not applied to benthic image classification before as a big benthic dataset as BenthicNet was not available to train SSL.

2.2 Self-Supervised Learning Methods

Self-supervised learning is unsupervised learning where a supervised task for training is created from unlabelled input data. A supervised task can be that of giving half of an image, predicting the other half, predicting the color channel values of the image given a greyscale version, or finding the correct order of shuffled image patches. Supervised learning requires a lot of high-quality labeled data which is expensive and time-consuming for computer vision tasks like image classification, object detection, or segmentation. Fortunately, unlabelled data is more common. SSL tries to learn representations from unlabelled data through a self-supervised task. These learned representations can be fine-tuned for another downstream task (e.g. image classification) with different labeled data. The motivation behind SSL is a model can learn the underlying structure of the data when trying to solve the pre-text task from the input data. One of the famous SSL approaches is contrastive learning.

Let's suppose we have a function f represented by a neural network model. Given an input x, f returns learned features $f(x)$. Let positive pairs be two different parts of the same image, two frames of the same video, or two augmented versions of the same image, while negative samples are patches from different images, frames from different videos, or augmented versions of different images. Contrastive learning tries that for any positive pair of inputs x_1 and x_2, the function outputs $f(x_1)$ and $f(x_2)$ should be as similar as possible while for a negative input x_3, the function outputs $f(x_1)$ and $f(x_2)$ should be as dissimilar to $f(x_3)$ as possible. In "Representation Learning with Contrastive Predictive Coding" [41] paper, the authors divide images into overlapping grid patches and train a model to predict the lower rows given the few patches from the upper rows of an image. To train the model effectively, a loss function should enforce the similarity of model output to the positive pair (correct patch) and dissimilarity to negative pairs (incorrect patches). For calculating the loss, the set of N patches is used: one positive sample (correct patch) and N-1 negative patch samples which are chosen from the same image or other images in the batch. This loss is referred to as InfoNCE loss (NCE - Noise Contrastive):

$$\mathcal{L}_{q,k^+,\{k^-\}} = -\log \frac{\exp\left(q \times k^+/\tau\right)}{\exp\left(q \times k^+/\tau\right) + \sum_{k^-} \exp\left(q \times k^-/\tau\right)}$$

Here q is the network output, k^+ is the positive (correct) patch, k^- is the set of N-1 negative patches, and τ is a temperature scaling factor. q, k^+, and k^- are all representations (output from the network) not original image patches. The evaluation of the learned representations by the method is done with a linear evaluation protocol. That is adding a linear classifier on top of the frozen encoder part of the model and training it, then evaluating with classification accuracy on ImageNet val/test partition. The Contrastive Predictive Coding method achieved top-1 accuracy of 48.7% outperforming all other unsupervised learning methods but still far from supervised counterparts. The idea of image discrimination was extended to instance discrimination.

Instance discrimination applies contrastive learning concepts to whole images. Instance discrimination tries to achieve that representations of the two different augmented versions of the same image(positive pair) are similar while the representations of the two augmented versions of the different images(negative pair) are dissimilar. The papers SimCLR [5] and MoCo [17] use instance discrimination. Their objective is to keep learned representations of an image invariant under augmentations. These augmentations include gaussian blur, a random resized crop, horizontal flip, and color distortion. Intuitively these augmentations can change an image visually but not the class of the image. Hence, the representations of images should not change. The instance discrimination method works as the following:

- Given an image, 2 randomly augmented versions of the image are created. Additionally, N-2 augmented versions of the other images from the dataset are taken.
- All N images are passed through the encoder model and representations are obtained
- InfoNCE loss is applied to the obtained representations.

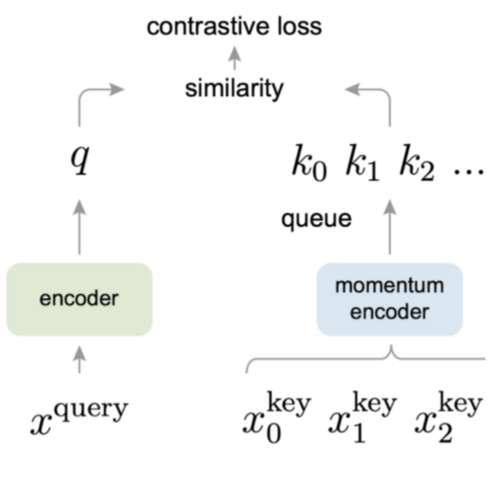
The main difference between SimCLR and MoCo is how they handle negative samples

In SimCLR, negative samples are all the images in the current batch. SimCLR trained in this way achieved the top-1 accuracy of 69.3 % using a linear evaluation protocol as described earlier. In practice, InfoNCE loss is highly dependent on the number of negative samples and it requires a big batch size of 4096 in the case of SimCLR. SimCLR was trained with an 8k batch size which is the main drawback of the method considering the required computational resources for that big batch size.

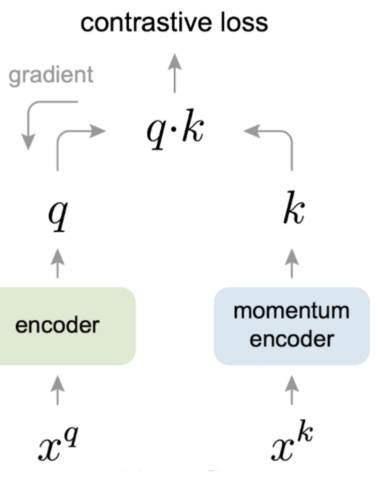
On the other hand, MoCo uses a queue of representations, a memory bank, from previous batches as negative samples. This decouples negative samples' size from batch size enabling smaller batch size during training resulting in more computational effectiveness. MoCo has two encoders: query and momentum. A query image is selected and processed by the query encoder to compute the encoded query image q. The InfoNCE loss is calculated for q with keys, encoded representations of other images, in the memory bank. When the current batch is passed through the momentum encoder it is enqueued to the memory bank where negative samples are kept and the oldest representation in the queue is dequeued(Figure 2.1). When the query encoder weights are updated with backpropagation and the new weights of the query encoder are copied to the momentum encoder the network performed poorly. The authors suggested that rapid change in momentum encoder parameters reduced the key representations' consistency. To address this, the authors proposed a momentum update on the parameters of the momentum encoder:

$$\theta_{\mathrm{k}} \leftarrow m\theta_{\mathrm{k}} + (1 - m)\theta_{\mathrm{q}}$$

Here θ_{k} is the parameters of the momentum encoder, and θ_{q} is the parameters of the query encoder, m is a momentum coefficient. Using this method MoCo achieved 60.6% top-1 accuracy on ImageNet after training for 200 epochs. MoCo v2 achieved 67.5% top-1 accuracy on ImageNet by introducing an MLP projection head and more data augmentations.



(a)



(b)

Figure 2.1: a) Momentum Contrast (MoCo) uses a contrastive loss by matching an encoded query q to a dictionary of encoded keys. The dictionary keys $k_0, k_1, k_2, ...$ are representations of the images in the batch by the momentum encoder. The dictionary is built as a queue, with the current mini-batch enqueued and the oldest mini-batch dequeued, decoupling it from the mini-batch size. b) Only the weights of the query encoder are updated by backpropagation. The momentum encoder is updated with a momentum update with the parameters of the query encoder. This method enables a large and consistent dictionary for contrastive learning. *Note.* From *"Momentum Contrast for Unsupervised Visual Representation Learning"*, by Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie., Ross Girshick, 2020

Chapter 3

Dataset

3.1 BenthicNet

BenthicNet is a vast collection of labeled and unlabeled images from different locations around the world. It has over 9.8 million images gathered from individuals, research groups, and government organizations [27]. The dataset tries to achieve thematic diversity by including images from various geographical locations and different marine environments. Some datasets in BenthicNet came from individual project partners. 2 281 images are extracted from passive drop-down video drifts conducted in the Bay of Fundy between 2017 and 2019 using 4K cameras. 4 094 high-quality images belong to the survey conducted in St Anns Bank between 2009-2014. 62 seabed images came from sources that are used for the 2017 R2Sonic Multispectral Challenge in the Bedford Basin, Nova Scotia. 3 000 images extracted from a video used for the "Coastal Habitat Mapping of Placentia Bay" project, in Newfoundland. The above four datasets came from east Canada. 1 220 images, collected in shallow eelgrass sites in Nova Scotia between 2019-2021, were provided by the Ecology Action Centre (EAC). All dataset sources are shown in Table 3.1. The dataset has non-thematic variety (e.g., image quality, lighting, perspective) as the images are gathered from different camera configurations and platforms. Some image examples are given in Figure 3.1 As the BenthicNet dataset is a collection of smaller datasets with different structures, all consisting datasets are brought to the same CSV format that is described in Table 3.2.

3.1.1 Subsampling

BenthicNet data is collected from a variety of sources as described above. However, images are not distributed geographically evenly. While some locations have only a few images taken manually by divers, other locations have a big number of images

Table 3.1: Dataset summaries for BenthicNet including their source, geographic region, number of comprising datasets and sites, and number of overall and subsampled images. *Note.* From *"BenthicNet: A global dataset of seafloor photography for deep learning applications"*, by Scott C. Lowe, Benjamin Misiuk, Manuscript in preparation.

				№ Samples	
Source	Region	№ Datasets	№ Sites	Full collection	Subsampled
Online Repository/Collection					
AADC	Antarctic	2	86	2056	2056
IMOS (via SQUIDLE+)	Australia	58	787	6 633 927	235 879
MBARI (via FathomNet)	W. USA	4	3196	61 508	53 313
MGDS	Global	4	4	2308	2110
NOAA (via OneStop)	USA	18	526	73 019	41 896
NRCan	Canada	73	1737	20 260	19 234
PANGAEA	Global	1112	1118	721 638	242 128
RLS (via SQUIDLE+)	Global	494	10 657	238 696	238 187
SOI (via SQUIDLE+/MGDS)	Global	15	149	1 307 009	47 726
SQUIDLE+ (other)	Global	9	227	557 925	48 599
USAP-DC	Antarctic	5	27	4144	2886
USGS	USA	5	38	104 155	7035
WHOI (via MGDS)	E. USA	1	9	2595	1985
Individual Contributions					
4D Oceans	E. Canada	2	274	3008	3000
LaboGeo (Marine Geosciences Lab/UFES)	E. Brazil	1	359	359	359
DFO (BIO)	Canada	6	381	7773	7722
DFO (IOS)	W. Canada	7	9	16 247	2324
EAC	E. Canada	1	7	1220	1192
Hakai Institute	W. Canada	2	45	4735	3941
MUN	Arctic	4	135	10 691	8579
NGU	Norway	4	580	50 290	50 275
NOAA (NFSC)	N.E. USA	1	2	2240	2240
SEAM	E. Canada	3	283	6692	6544
Total	Global	1831	20 636	9 832 495	1 029 210

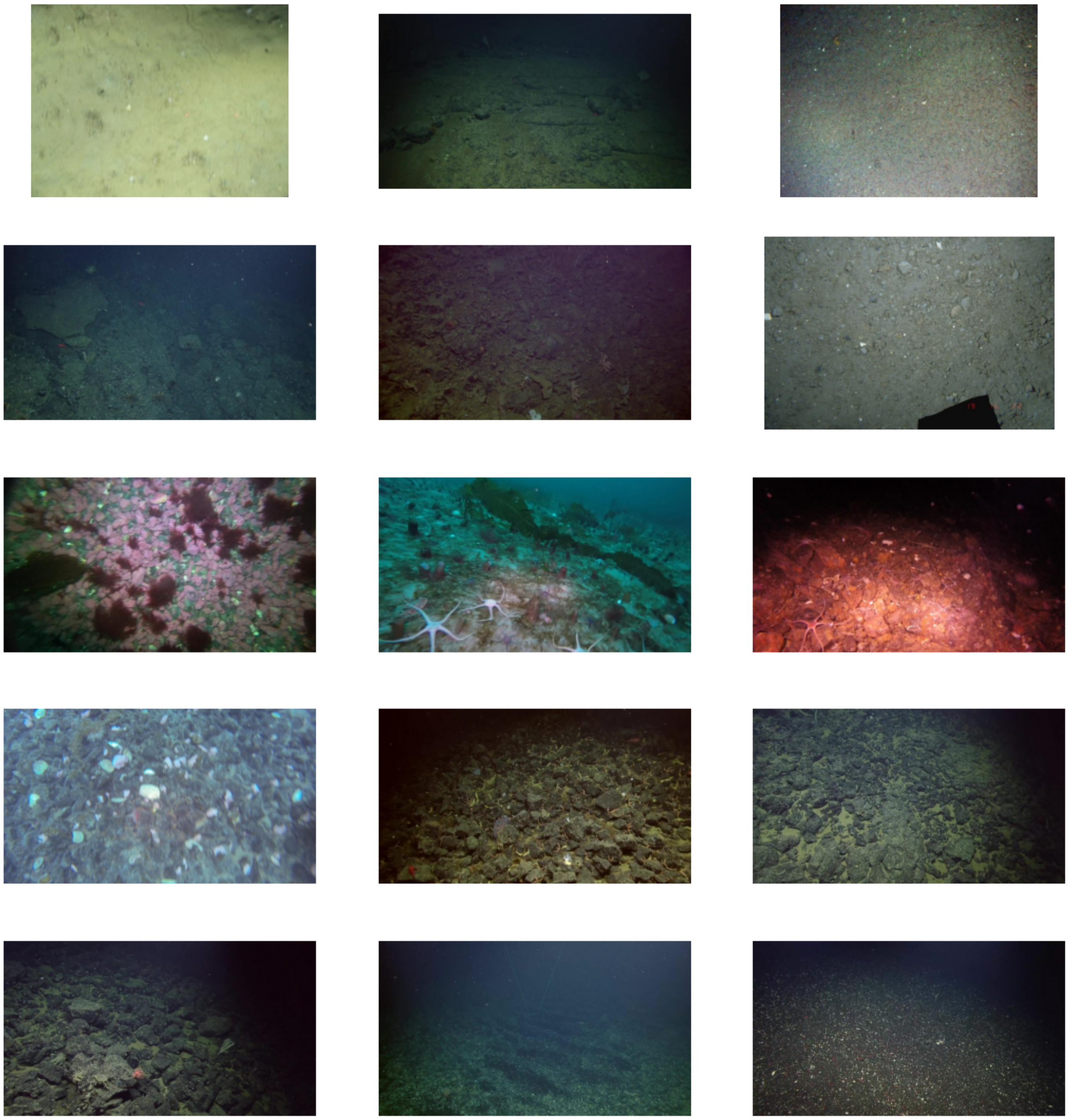


Figure 3.1: Image samples from BenthicNet

Table 3.2: CSV format for BenthicNet. *Note.* From *"BenthicNet: A global dataset of seafloor photography for deep learning applications"*, by Scott C. Lowe, Benjamin Misiuk, Manuscript in preparation.

Column	Contents	Data-type	Units	Coverage of images
dataset	Name of contributing dataset	string		100.0%
site	Name of recording location	string		100.0%
image	Image filename (possibly including an extension)	string		92.3%
url	URL address for this image	string		90.3%
source	Collection containing contributing dataset	string		100.0%
longitude	Longitude (WGS 84)	float	degree	99.5%
latitude	Latitude (WGS 84)	float	degree	99.5%
datetime	Acquisition date and time (UTC)	string	YYYY-MM-DD HH:mm:ss	94.6%
depth	Seafloor depth	float	metre	70.8%
altitude	Distance from camera to seafloor	float	metre	32.0%
temperature	Seawater temperature	float	degrees Celsius	28.0%
salinity	Seawater salinity	float		28.0%
chlorophyll	Seawater chlorophyll concentration	float		20.0%

as they are collected by AUVs. As a result, the dataset is subsampled to lessen the sampling density imbalance across all data sources [27]. Subsampling also resulted in a more manageable dataset size, while preserving the available variety of benthic habitats. Most of the sub-datasets in BenthicNet have annotations for the recording station or camera deployment where a group of images was collected that is called "site" onwards. To achieve as much diversity in location and represented habitat in each image as possible subsampling is done on the site level. There are 20 913 sites in the BenthicNet dataset.

The base target number of images for subsampling from each site is 250 whereas subsampling the site with less than 250 images does not reduce the number of images for that site. The base target number was chosen to keep a good balance between the subsampled dataset size and benthic habitat variety by our research group members. Some subdatasets do not annotate which images were collected from the same geographical site despite grouping the images into some distinct locations which can be considered a site. For addressing this, the number of pseudo-sites was calculated automatically by finding the number of clusters of samples with inter-distance of at least 1 000 meters, then the target number for the site is increased by a factor of the number of pseudo-sites within the site. 2 350 additional pseudo-sites were found across 591 of the 20 913 annotated sites. Moreover, a few pseudo-sites contained pictures that were all in the same area, though others had a few hundred meters distance between samples, which were denoted as “subsites”. The target number of samples for the site is increased by 50 for each subsite separated by at least 100 m. 2 353 sites had more than one subsite. 17 365 sites with less than 40 samples for each of its pseudo-site were not subsampled. 145 sites could not be spatially subsampled as they lack accurate coordinates for the samples. These sites were subsampled by selecting evenly spaced images, as described below. The remaining 3 403 sites were subsampled spatially starting with the inclusion of the first image in the subdataset in the subsampled dataset. Scanning through the images, all images were removed until reached an image that was at least 0.625 m away from all images already flagged for inclusion in the subsampled dataset. Either this image or the next image, whichever was closer to 1.25 m from the previous image included, was selected. This process was repeated until the end of the images at a site. Many sites still had more images than their target number of samples after this initial spatial subsampling, so this process was repeated with larger separation distances to achieve the target subsample size. The separation distances were scaled up by factors of 2, 3, 4, 6, 8, 10, 12, 14, or 16 compared to the base subsampling of 0.625 m minimum and 1.25 m target separation to achieve the desired subsample size (i.e. corresponding to 1.25 m minimum, 2.5 m target; then 1.875 m minimum, 3.75 m target; all the way through to 10 m minimum and 20 m target separation) [27]. The subsampling distance selected (and hence the subsampled set of images at that site) was the largest distance which did not reduce the total number of images below the target for the site determined as described above. Subsampling procedure selected 1 037 003 images (10.53% of the total) to be included in the subsampled BenthicNet dataset (Figure 3.2). The kernel density estimates (KDEs) before and after subsampling can be seen in Figure 3.3 and Figure 3.4.

3.2 Labelling Scheme - CATAMI

BenthicNet has about 100 000 labeled images which have different labeling schemes for different datasets. There are 3 types of labeling styles: whole-frame, point, and coverage. Whole-frame labeling assigns single or multiple labels for the whole image. Point labels have separate labels for some pixel points within an image. Coverage labeling annotates coverage of some specific label in an image as an area in m^2 or

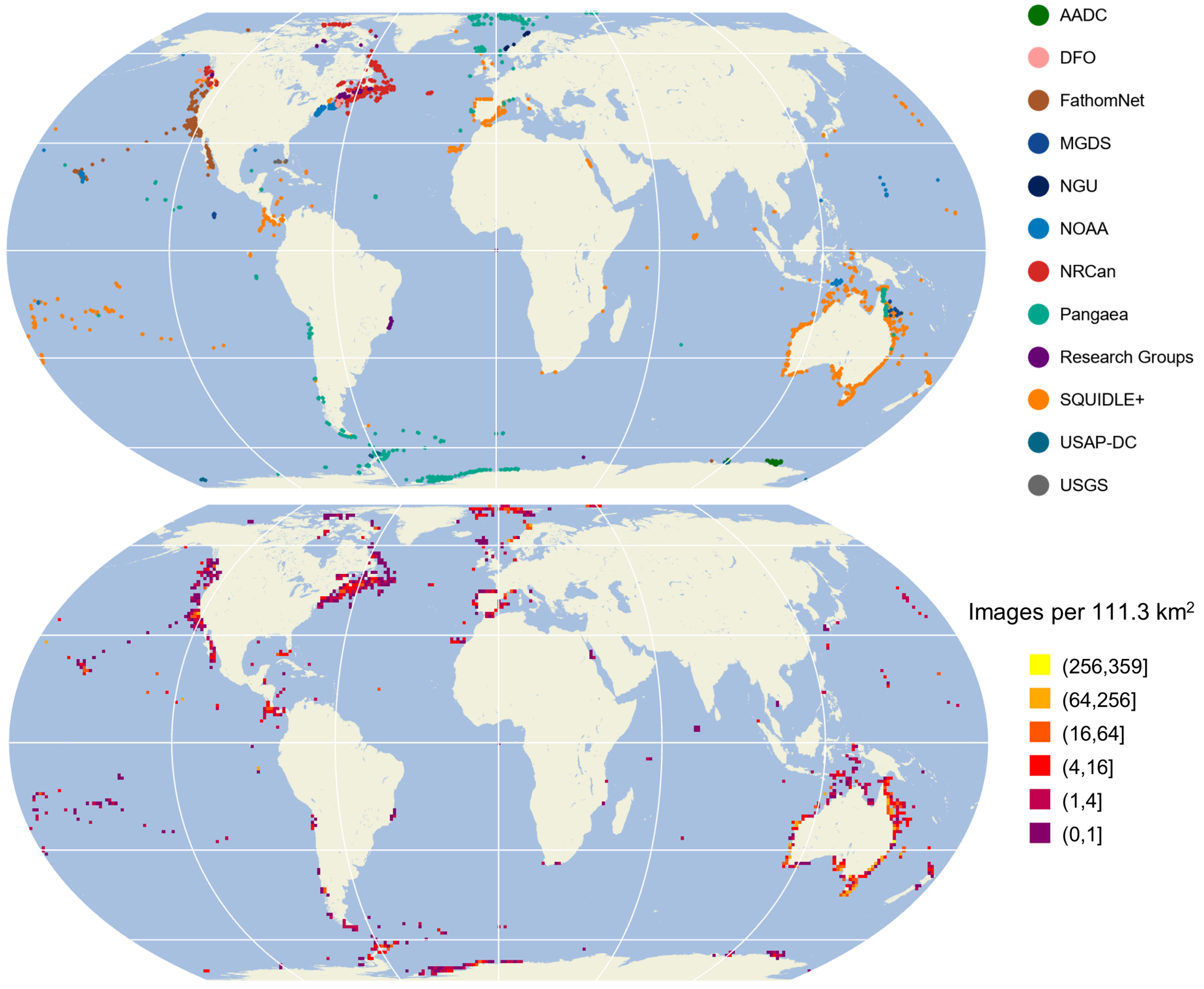


Figure 3.2: (Top) source and location of image samples after spatial subsampling. (Bottom) Aggregated image sample density, scaled logarithmically and projected to Equal Earth. *Note.* From *"BenthicNet: A global dataset of seafloor photography for deep learning applications"*, by Scott C. Lowe, Benjamin Misiuk, Manuscript in preparation.

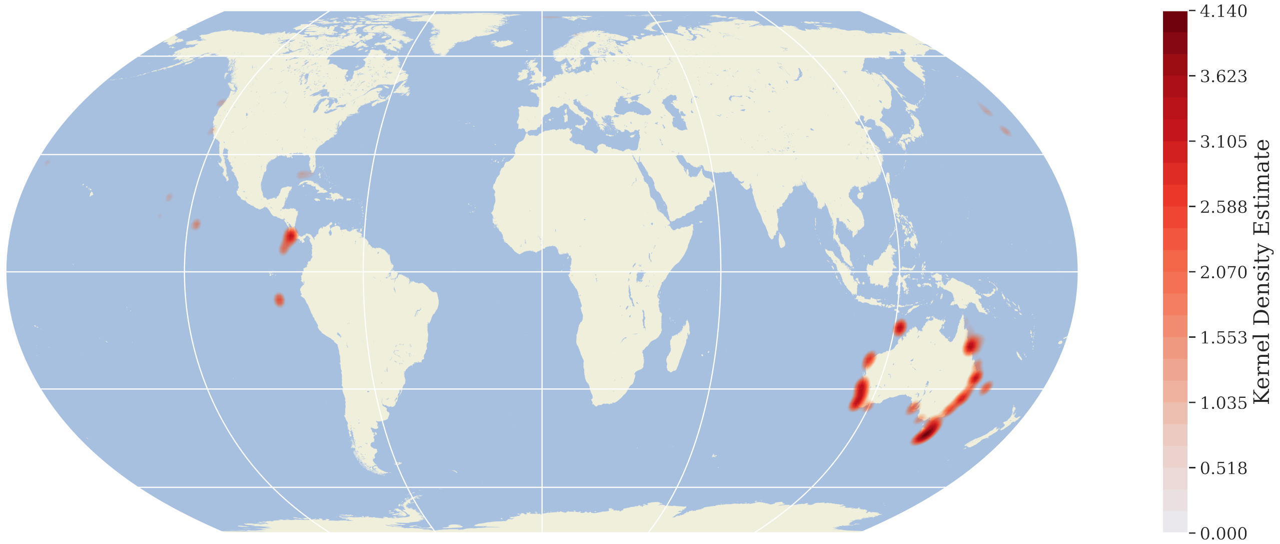


Figure 3.3: Kernel density estimate for BenthicNet imagery with all 9.8M sourced images. Gaussian kernel, with 1° bandwidth and Haversine distance metric. *Note.* From *"BenthicNet: A global dataset of seafloor photography for deep learning applications"*, by Scott C. Lowe, Benjamin Misiuk, Manuscript in preparation.

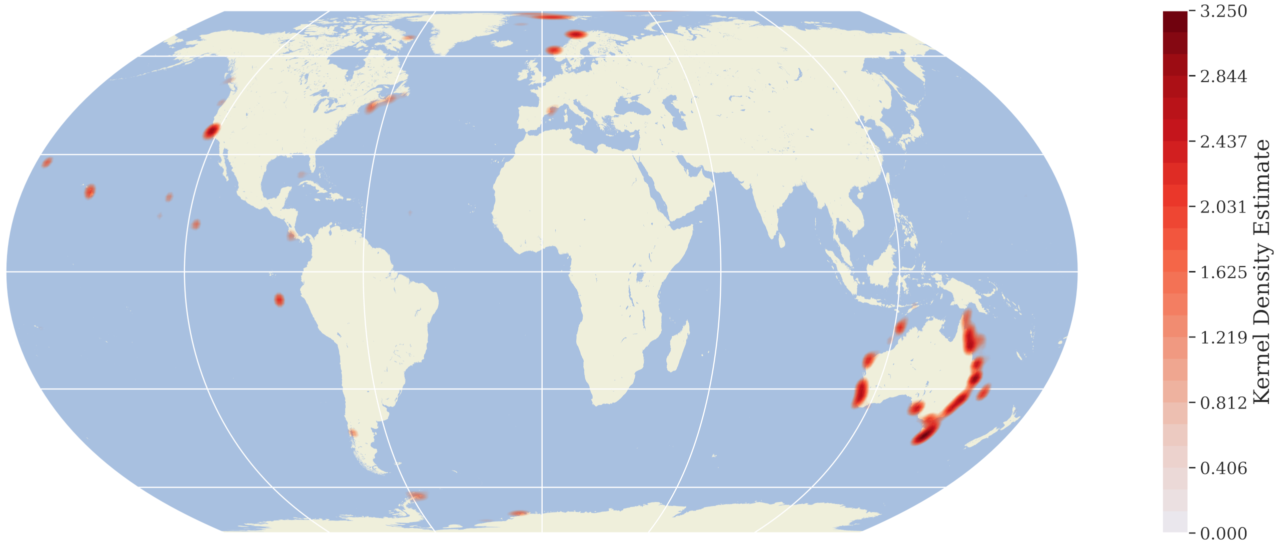


Figure 3.4: Kernel density estimate for BenthicNet imagery with 1M subsampled images. Gaussian kernel, with 1° bandwidth and Haversine distance metric. *Note.* From *"BenthicNet: A global dataset of seafloor photography for deep learning applications"*, by Scott C. Lowe, Benjamin Misiuk, Manuscript in preparation.

%. In order to be able to use all labeled images to train a model we need a common labeling scheme. The CATAMI classification scheme provides an Australian-wide acknowledged, standardized terminology for annotating substrate and biota in benthic imagery [1]. It has four independent label types: substrate, relief, bedforms, and biota. Each has a hierarchical structure, the deeper the label is, the more specific it is. Researchers can choose the labeling depth for their own purposes. The mapping rules for various labeling schemes to CATAMI were created by oceanographers and they are in the form of CSV. The basic structure of mapping CSV has 5 columns: the first one is for labels from the source labeling scheme and the other four columns are for matching labels for each label type in CATAMI. The CATAMI label columns can have the following values other than corresponding CATAMI label name:

- *NONE* - There are no instances of this label type in the image (e.g. there is no biota in the image).
- *N/A* - This label type (e.g. substrate) does not apply to this label because the source scheme already has separate columns for (e.g.) substrate and biota (so the substrate is known, but by looking at the OTHER label not this label).
- *REMOVE* - This label needs to be removed.
- *UNKNOWN* - From the source label, there is no way to know whether this output label type is present in the image.

When mapping source labels for an image to CATAMI, we keep the most specific labels for an image to avoid redundancy in annotations. For example when mapping a label for an image if we already have a more specific label for this image that includes a new label we skip it. If we have a less specific version of the new label for an image we replace it with a more specific one. The source code for mapping each subdataset in BenthicNet can be found here.

3.2.1 Subtrate

The substrate is a type of physical environment in the surface being observed [1]. It has two subdivisions: unconsolidated (i.e. soft substrates) and consolidated (i.e. hard substrates). In benthic environments, hard substrates can be covered by a

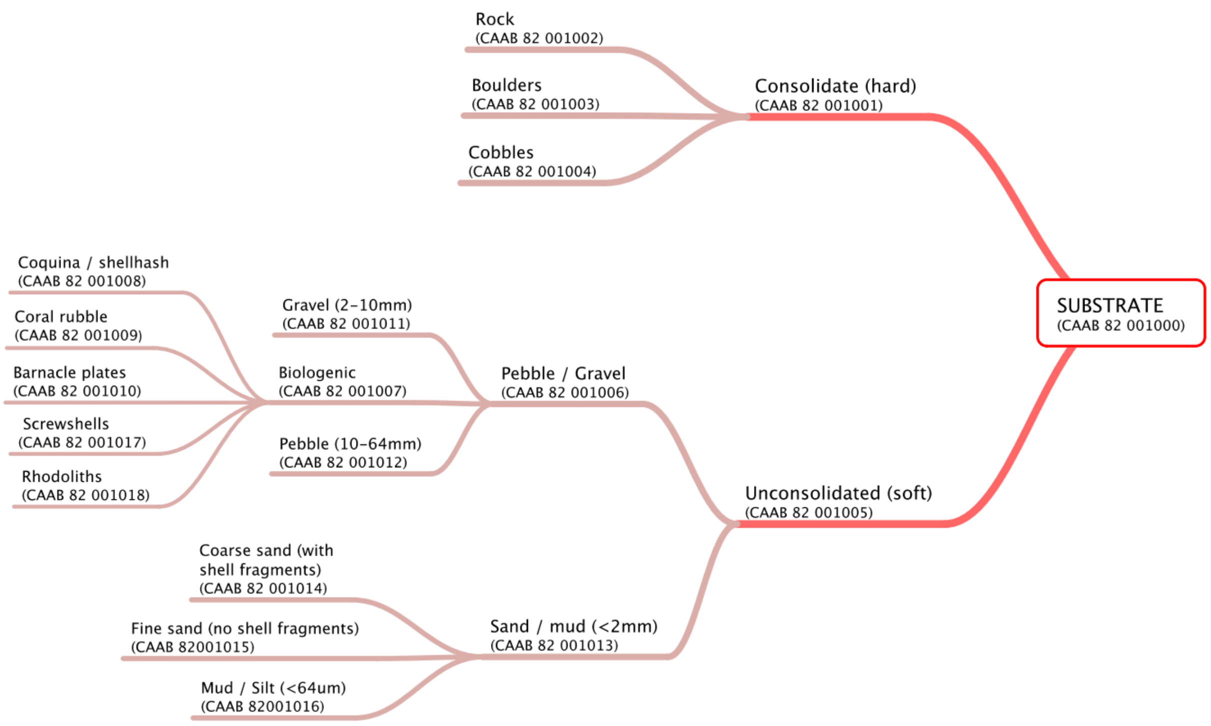


Figure 3.5: Hierarchical structure for the Substrate branch of the CATAMI Classification Scheme. *Note.* From CATAMI class PDFGuide V4 20141218, by Althaus Franziska, Hill Nicole, Edwards Luke, Ferrari Legorreta Renata, 2013.

thin layer of sand or mud. In this case, CATAMI annotates the visible substrate. Consolidated substrates are divided into three types based on element size: cobbles, boulders and bedrock. Cobbles are distinct rocks of approximately 65-255 mm in diameter. Boulders are large rocks with a diameter >255 mm where clear edges can be determined. Bedrock is a flat surface outcropping ledge or cliff face that can be covered in biota and/or a veneer of sediments. Unconsolidated substrates are divided into two types based on grain size: ”Sand/mud” and ”Pebble/gravel”. Pebble/gravel has a diameter of 2-64 mm while Sand/mud grainsize is smaller than 2 mm in diameter. ”Sand/mud” and ”Pebble/gravel” has more subdivisions which one can see in the complete CATAMI Substrate hierarchy figure Figure 3.5

3.2.2 Relief

Relief describes the height and rugosity of the substrate [1]. This label type annotates a whole image rather than a point in it as the height and rugosity feature of a location

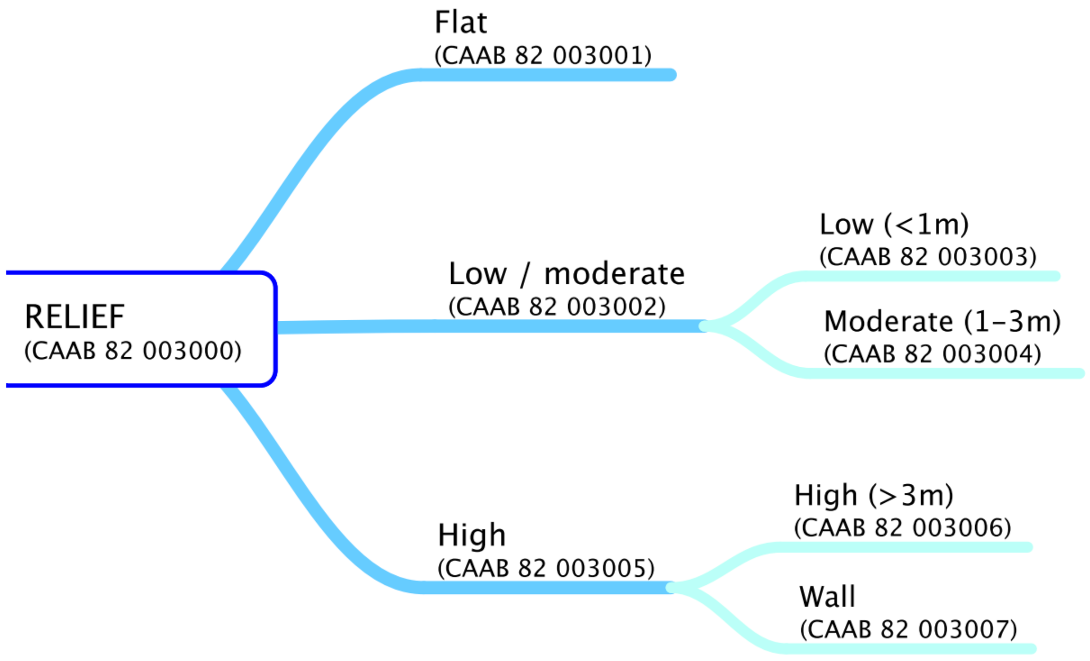


Figure 3.6: Hierarchical structure for the Relief branch of the CATAMI Classification Scheme. *Note.* From CATAMI class PDFGuide V4 20141218, by Althaus Franziska, Hill Nicole, Edwards Luke, Ferrari Legorreta Renata, 2013.

can be determined by comparing it to its surrounding. Relief has three sub-categories: flat, low/moderate, and high. The flat category denotes a flat substrate with no features. The Low/moderate category describes the height feature of the substrate with less than 3m that can be steps or outcrops Rockwalls, cliffs, or high steps with a height feature of more than 3m belong to the high category. The second and third categories have further divisions that is shown in the full hierarchy tree of relief label type in Figure 3.6

3.2.3 Bedforms

Bedforms are the structural changes caused by the transportation of sediment on the seabed due to water movement [1]. As the possible changes in hard rocky surfaces resulting from sedimentary processes over a long time period can not be estimated from the imagery itself, bedform annotations are done for only unconsolidated substrates. Bedform labels are divided into 4 categories: Bioturbated, 2D, 3D, and None. The image is annotated as bioturbated if the described substrate is structured by burros and/or tracks formed by biota. 2D bedforms are defined as straight-crested features

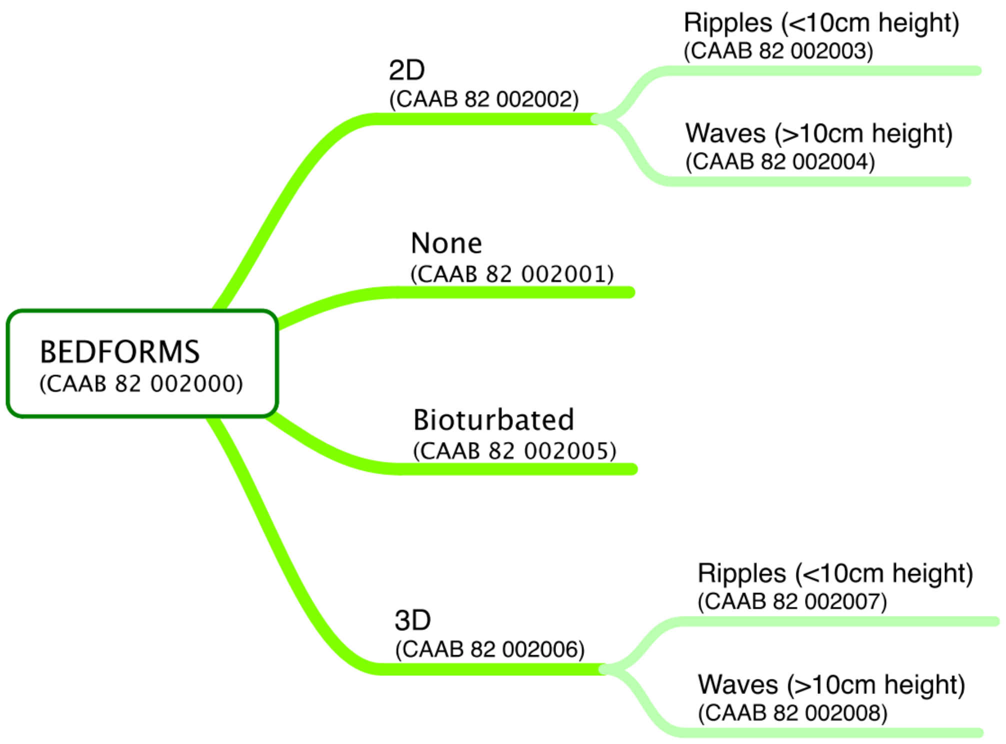


Figure 3.7: Hierarchical structure for the Bedforms branch of the CATAMI Classification Scheme. *Note.* From CATAMI class PDFGuide V4 20141218, by Althaus Franziska, Hill Nicole, Edwards Luke, Ferrari Legorreta Renata, 2013.

in a planar view [2] and are further divided into ripples and waves according to the height of the 2-dimensional feature. Three-dimensional bedforms have sinuous to wavy crestlines with distinguishing scour pits [2]. It is also divided into two categories as same as 2D bedforms. The images showing flat soft substrate surfaces without any bedform, typical for deep-sea habitats, are labeled as None. The full hierarchy tree for Bedform annotations is described in Figure 3.7

3.2.4 Biota

Biota annotations refer to the visible benthic organisms or types of flora on the ocean floor. This label type also annotates visible traces of biota, referred to as bioturbation. Biota has 16 subdivisions:

1. Bacterial mats
2. Macroalgae
3. Seagrasses
4. Sponges
5. Cnidaria
6. Jellies

7. Worms
8. Bryozoa
9. Ascidia
10. Crustacea
11. Seaspiders
12. Echinoderms
13. Brachiopods
14. Molluscs
15. Fishes
16. Bioturbation

Each subdivision has further branched into more specific biota types. The further division of worms is shown in Figure 3.8. The full hierarchy tree for other biota subdivisions and the sample images for all label types and subtypes in CATAMI can be found in the original ”CATAMI Class PDF Guide” [1].

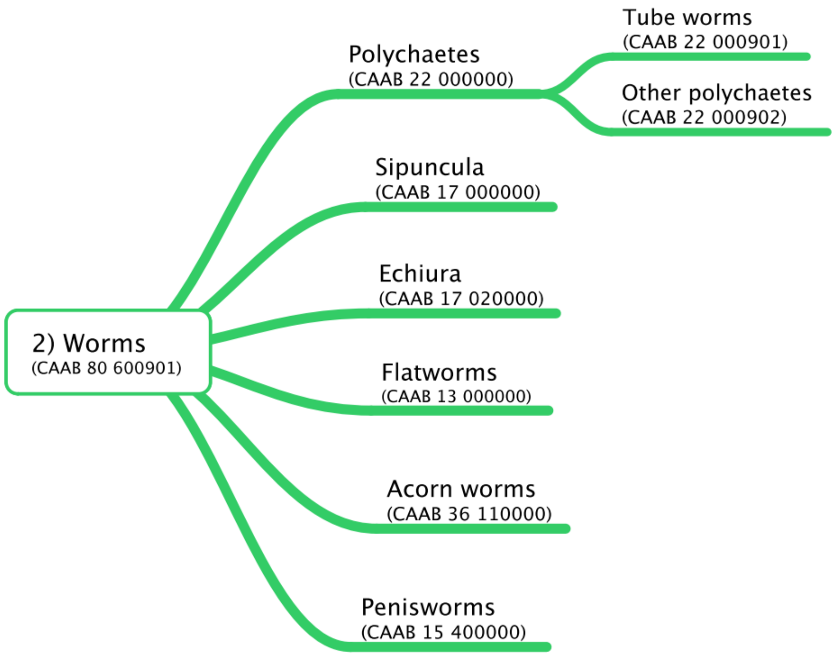


Figure 3.8: Hierarchical structure for the worms subbranch of Biota branch of the CATAMI Classification Scheme. *Note.* From CATAMI class PDFGuide V4 20141218, by Althaus Franziska, Hill Nicole, Edwards Luke, Ferrari Legorreta Renata, 2013.

Chapter 4

Methods

In this chapter, we talk about the methods used in the thesis. The BenthicNet has the following properties and limitations:

- BenthicNet is a large dataset with 10 million images
- Labelled images can have multiple labels for the whole frame or a pixel inside an image.
- The absence of some label for an image does not always mean the absence of that feature on the image.
- Not all the labels are complete, meaning not all images are labeled as specifically as the leaf nodes in the hierarchy tree for a specific label type.

Taking all the complexity of the dataset, we wanted to explore the habitat mapping problem in a simplified way, by working on the single-label classification task. For that, we created a dataset with only whole-frame labeled images in BenthicNet and used substrate labels truncated in the depth of three. We refer to this dataset as a simplified BenthicNet dataset from here on. As some images have multiple labels, if one image has contradicting substrate labels in the depth of three, this image is discarded from the dataset.

4.1 Partioning

We have four partitions for our simplified BenthicNet dataset: train, validation, and two test (test same and test other). The test other partition is selected by keeping geographically close images in the same partition while the test same partition is chosen through stratified random sampling across classes. Having the right test partition to evaluate the generalization of the model on benthic data is important as the model performance differ substantially between the two test partitions. The reason behind the performance gap could be that geographically close underwater images represent very similar benthic habitats as one type of habitat can span many kilometers while images from spatially remote locations differ greatly in terms of habitat elements. To experimentally show the difference in the generalization ability of a model on different test partitions, we trained the supervised model on one of the labeled sub-datasets of BenthicNet which is discussed in chapter 5. The test other partition is an appropriate test data when we want to train our model with data from one location and want to use the model to classify images from another unseen location. The test same data is appropriate when we train our model with labeled data in one location and want our model to label unlabelled images from that location.

4.2 Supervised Model

We trained a supervised model with the simplified BenthicNet dataset to analyze its performance in the classification task. The main model architecture used was ResNet [19], as it is one of the go-to model architectures for image classification. Before ResNet architecture was introduced, researchers tried to build deep Convolutional Neural Networks, which they believed should be more capable as deeper CNNs have more parameter space to explore than shallower ones. However, it has been witnessed that after some depth the performance gets worse. This was one of the problems with VGG [36], the authors could not build an as deep model as they wanted due to the loss in generalization capability. This problem is caused by vanishing gradients. When the network is deep, the gradient calculated from the loss in the last layers shrinks to nearly zero as the chain rule is applied many times before gradients reach earlier layers. This avoids the weights in the earlier layers to be updated which as a result avoids learning. ResNet used the residual connection between layers so that the gradients can flow to the initial layers enabling better learning despite the depth of the network.

Depending on the number of overall layers and how big each layer is operations-wise, ResNets have various sizes. Figure 4.1 shows the structure of the ResNet-18.

Every ResNet has four layers (given with the same color) after the first common step (given in yellow color). Only the number of operations and the parameters in the operations differs for different ResNet architectures. Here an operation means a

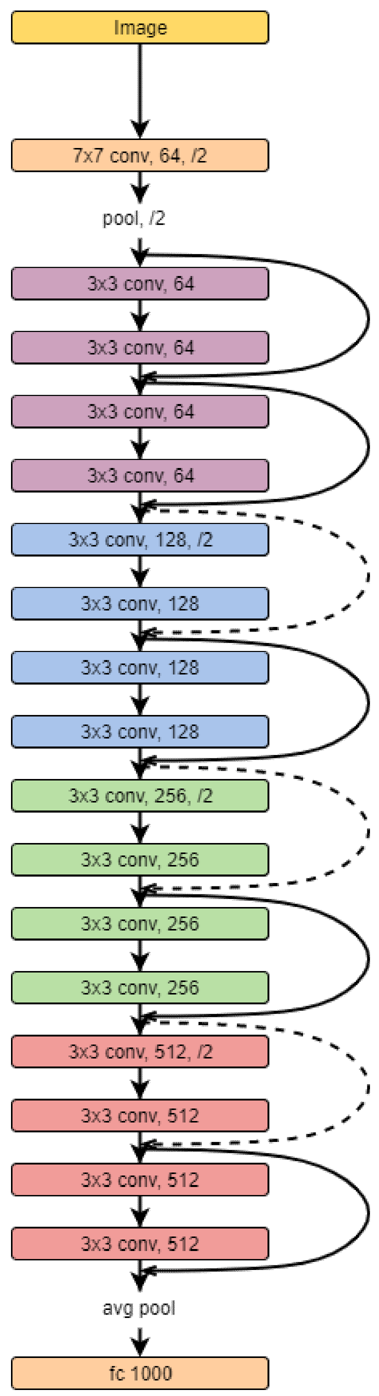


Figure 4.1: A residual network with 18 parameter layers. The dashed shortcuts mean a change in dimensions. *Note.* Adapted from *"Deep Residual Learning for Image Recognition"*, by Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, 2015

convolution, a batch norm, and ReLU activation. The last operation in each layer does not have activation at the end. Now let's discuss operations and layers in more detail for ResNet-18 in the figure for ImageNet with input size 224×224 . The first step is common for all ResNets and consists of a convolution (7×7 kernel, filter 64 and stride, batch normalization, and max pooling (3×3 kernel, stride 2). The output dimension of this step is 56×56. As the figure shows all preceding layers follow the same pattern. They perform 3×3 convolutions a number of times with fixed filter size 64, 128, 256, and 512 respectively along with keeping the width and height dimensions constant within a layer. The network bypasses input every 2 convolutions. The dashed line means there has been a change in the input dimension. When the input from the previous layer is passing to the new layer its width and height are decreased twice by changing the stride to 2. The bypassing input is reduced in size by 1×1 convolution with stride 2 to facilitate addition. The output from the fourth layer is flattened after average pooling. For our first experiment in chapter 5, we use ResNet-18 architecture as it has fewer parameters than other models with ResNet architecture, matching the dataset size used. Besides, to check the dependency of our results on model architecture we used EfficientNetB0 [39]. For the other experiments, we used ResNet-50 architecture for fair comparison as the pretrained SSL models on ImageNet use this architecture.

4.3 Self-supervised Model

As the labeled data is a small part of the whole BenthicNet dataset we want to take advantage of the unlabeled data. As we discussed in earlier sections self-supervised learning helps to learn the representations of the data without labels. First, we investigate the quality of the learned representation by self-supervised learning on ImageNet on the classification task in BenthicNet. Second, we want to train the SSL model on subsampled BenthicNet data and compare the quality of the learned representations with the ImageNet representations. To train our SSL models we use Sololearn library [7] which includes many state-of-the-art self-supervised learning methods of visual representation learning. All methods are implemented using Pytorch [29] and Pytorch Lightning [11].

Chapter 5

The importance of the right test data

In this chapter, we explore the performance of a model on spatially and randomly picked test data. Before compiling the whole BenthicNet dataset we train a supervised model on a smaller dataset.

5.1 Dataset

The dataset of benthic habitat photographs was collected at St. Anns Bank (Atlantic Canada) in 2014 [25]. The dataset is comprised of 4 681 images, collected from 64 stations. The images within the dataset have been annotated with full-frame labels, using a classification scheme bespoke to this dataset and consisting of seven distinct benthoscapes types. Example images for each habitat class are shown in Figure 5.1.

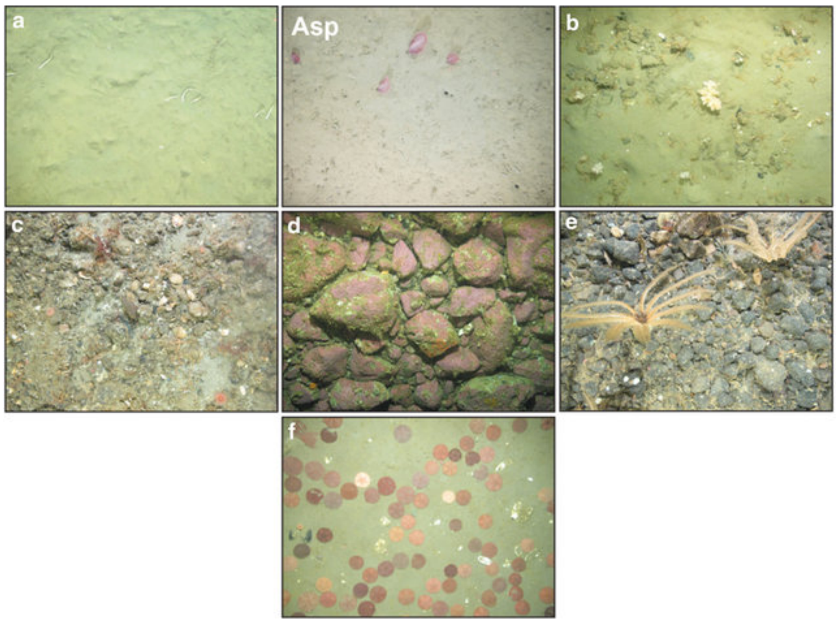


Figure 5.1: Representative sample images of each benthoscape class in the St. Anns Bank dataset. **a**: Mud. **Asp**: Bioturbated mud with seapens. **b**: Gravelly sand/mud. **c**: Till. **d**: Till with coral-line algae. **e**: Gravel with Crinoids. **f**: Sand with sand dollars.

We partitioned the dataset for training and testing as follows, and indicated in Table 5.1. Since the distance between images at the same station is much smaller than the distance between stations, photographs at the same station are highly similar. To test the model's ability to generalize to novel data, we held out some stations from training, placing approximately fourteen of the stations into the "test:other-station" set. The held-out stations were selected with stratification against the dominant habitat class at that station, such that around 20% of the samples for each class were placed in the "test:other-station" set.

For the remaining stations, we randomly selected 50 images from each class and placed these in a "Test:same-station" partition. This partition allows us to test the ability of the model to generalize to unseen images taken from stations on which the model was trained. The remaining data comprised the training partition.

		№ stations		№ images			
Class	Description	Train	Test:other	Train	Test:same	Test:other	Total
A	Mud (bioturbated in places)	14	3	689	50	190	929
Asp	Bioturbated mud with seapens	3	1	196	50	77	323
B	Gravelly sand/mud (<50% gravel)	11	2	377	50	97	524
C	Till (>50% cobbles, gravel)	18	4	1040	50	296	1386
D	Till with coral-line algae	10	3	715	50	191	956
E	Gravel with Crinoids	2	1	106	50	60	216
F	Sand with sand dollars	3	1	204	50	93	347
	Overall	50	14	3327	350	1004	4681

Table 5.1: Dataset statistics. The number of recording stations and image samples for each partition: training, test:same-station, and test:other-station.

5.2 Methods

We trained models using two convolutional architectures: ResNet-18 [19] and EfficientNet-B0 [39]. Since our dataset is comparatively small, we employed a transfer learning approach. We initialized the model using weights pre-trained on ImageNet-1k before training on our own training data. Each model was trained for 50 epochs using the Adam optimizer [23] to minimize the cross entropy between the network's output logits and the ground truth class labels. We used a cosine-annealed one-cycle learning rate schedule [37], with a peak learning rate of 1×10^{-4} occurring after 30% of training, and cyclic momentum from 0.95 to 0.85. The training batches each contained

128 images, and we used a weight decay of 1×10^{-4}.

Before training, all images were downscaled such that their shortest side was the length of 512. For some of our experiments, we used online data augmentation as follows. First, we randomly cropped the image to select a square of size 512×512. The brightness, contrast, saturation, and hue were each multiplied by a random factor chosen uniformly from $[0.6, 1.4]$. With probability 0.5, the image was rotated by a random angle, up to 360°. The image was flipped horizontally with a probability of 0.5. Finally, we resized the image to 224×224 and normalized the RGB values using the mean and standard deviation statistics from ImageNet-1k.

Our dataset is heavily imbalanced, with ten times more samples for the largest class than for the smallest class. In some of our experiments, we addressed this imbalance with oversampling. To do so, we changed our batch sampler to select images at random with the probability of each image inversely proportional to the number of samples for that class, such that there was a uniform distribution of classes shown to the network. Samples were randomly selected with replacement, and the total number of samples per batch (128) and the number of batches per epoch (25) were held the same as when training without rebalancing.

5.3 Results

We trained both a ResNet-18 and EfficientNet-B0 model, with transfer learning from weights pre-trained on ImageNet-1k. The performance of the two networks during training is shown in Fig. 5.2. Overall, our results are consistent between the two architectures. The various training configurations indicated in the plots are explained below.

As outlined earlier, there are two ways to test the generalization performance of the networks. First, we considered the performance on samples that were unseen during training but recorded at the same stations as the training examples (test:same-stations, dotted lines in Fig. 5.2). Here, we found that test performance was high ($> 92\%$), with a marginally higher performance achieved by the more powerful EfficientNet-B0 model. Second, we considered the performance on sample images from held-out stations which did not appear in the training set (test:other-stations, dashed lines). In this case, we found performance reached a plateau after only 5 epochs

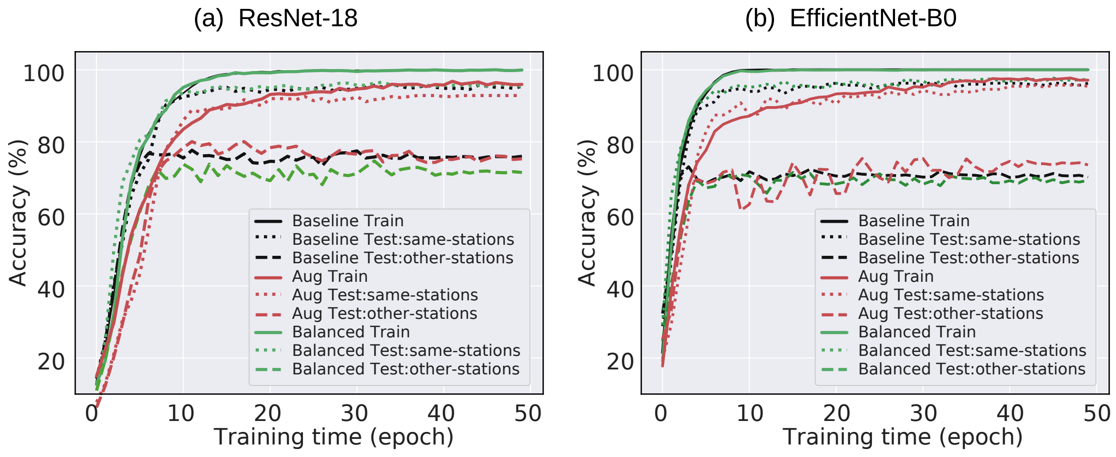


Figure 5.2: Accuracy of benthic habitat classification with different training configurations. Aug: with training data augmentation. Balanced: with class rebalancing.

of training. Performance on samples from unseen stations (70–75%) was much worse than on images from seen stations.

The results indicate that images from the same station can be very similar, to the extent that they are pseudoreplicas of one another. The amount of variability within a single station is notably lower than the variability between stations. Consequently, the test:same-station partition does not capture enough of the true variability between instances of the target class to reliably estimate the model's generalization ability over the target class. In contrast, the variability between images of different stations is big enough to challenge models' ability to generalize. It is a discouraging finding as we want to build a habitat mapping model which generalizes well all around the world by training it on the data from some locations around the world.

Chapter 6

The benefit from the learned representations from ImageNet

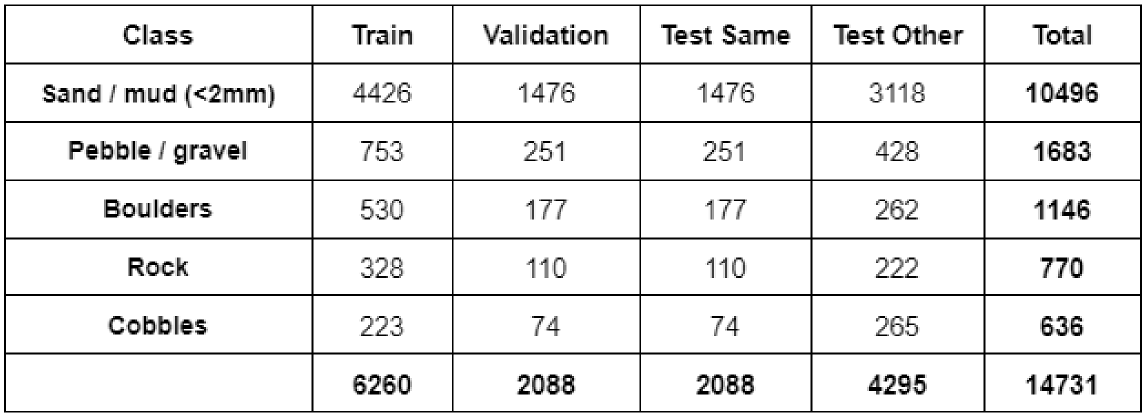
In this chapter, we explore the quality of transfer learning from the ImageNet pretrained model by supervised learning. The dataset to be used is a bigger labeled dataset from BenthicNet which includes multiple survey imagery from different geographical locations.

6.1 Dataset

Considering the complexity of the labeled BenthicNet dataset (described in Chapter 4), for simplification of the classification task we used the simplified BenthicNet dataset which has five classes, and only the whole frame labeled images because coverage and point labels could have opposing labels for the whole image as they annotate some parts of an image. If we have opposing labels for the whole frame before the depth of three we excluded those images also. The chosen dataset has 14 731 images overall. The dataset was divided into four partitions: train, validation, and two test (same and other). First, test-other data was chosen so that all geographically close images are on one partition, and the test-same partition(20% of remaining data) was chosen by stratified sampling from the rest of the data. The remaining unallocated data was divided into train and validation partitions with a 3:1 ratio. The number of images across partitions and classes is shown in Table 6.1. There is a strong imbalance in the dataset as the largest class "Sand/mud" is six times as big as the second largest class "Pebble/gravel".

6.2 Methods

For this supervised classification task, we used the ResNet-18 architecture implemented in Pytorch. We modified the number of neurons in the last fully connected layer to five, the number of classes we have. We wanted to see the effect of learned



Class	Train	Validation	Test Same	Test Other	Total
Sand / mud (<2mm)	4426	1476	1476	3118	**10496**
Pebble / gravel	753	251	251	428	**1683**
Boulders	530	177	177	262	**1146**
Rock	328	110	110	222	**770**
Cobbles	223	74	74	265	**636**
	6260	**2088**	**2088**	**4295**	**14731**

Figure 6.1: The number of samples in each partition for the whole-frame annotated subset of BenthicNet.

representations from ImageNet compared to a model trained from scratch. We have four training setups for our experiment:

FS : We train a model from scratch with the dataset.

LP : We initialize the model with pretrained weights from ImageNet and train only the last layer which is modified and freeze all other layers. Here LP stands for a linear probe that is probing the learned representations in the model by a fully connected layer.

FT : We initialize the model with pretrained weights from ImageNet and retrain the whole model with our dataset. Here FT stands for fine-tuning and it means taking the weights of a trained neural network and using it as initialization for a new model being trained on data from the same domain.

LP+FT : We initialize the model with pretrained weights from ImageNet, then we freeze all layers except the last and train it for half of the whole training. In the second half of the training, we unfreeze all the layers and continue training the whole model.

As the dataset is imbalanced, models were trained with weighted cross-entropy loss with a learning rate of 0.002 for 100 epochs. Weighted cross entropy gives different weights to each sample according to which class they belong. It is useful to address overfitting when we have an imbalanced dataset. The weights in the loss function for each class are inversely proportional to the number of samples for that class. When training from scratch with random initial weights we need a bigger learning rate compared to the other three setups and we used 0.01 in our experiment. The Adam optimizer with weight decay value 1e-4 was used. The augmentation stack during the training is the standard ImageNet augmentations for supervised learning: random resized crop to 224×224 pixels with scale interval $[0.3, 1]$, followed by random horizontal flip with 0.5 probability. The color channels of images were normalized with ImageNet statistics at the end. For validation and testing center crop of 224x224 from resized image to 256 in short dimension was used. For obtaining statistical measures we trained each setup five times with different random seeds from 0 to 4.

6.3 Results

The average training loss curve for five runs with different random seeds on training data is given on Figure 6.2. The shaded region around the lines is the standard deviation. When training from scratch (FS), the loss is bigger than the other setups with initial pretrained weights supporting that learned representations from ImageNet are indeed useful. When only the linear probe is trained (LP), model weights reach a minimum in the loss plane and do not decrease much after 60 epochs. It can be explained with that the learned representations from previous layers are not changing and model weights in the last layers have already reached their optimal values for the state of the previous layers. Besides when we unfreeze all the layers for LP+FT, on epoch 51, we see a surge in the loss and a steeper decrease afterward. A jump in loss value means the model is forgetting learned representations from previous epochs. To address this, we need to use a smaller learning rate when unfreezing all layers so that we keep learned features as much as possible and train it more. The FS, LP+FT, and FT setups have not reached their plateau and can be trained longer.

Now let's see how the models are performing on the classification task. As the dataset is imbalanced we evaluate models' performances by macro average F1-score across classes. The macro average F1-score is the arithmetic mean of F1-scores across all classes. From the training macro average F1-score curves on training and validation data (Figure 6.3 and Figure 6.4), It is clear learned representations from ImageNet help as all models initialized from pre-trained weights perform better than models

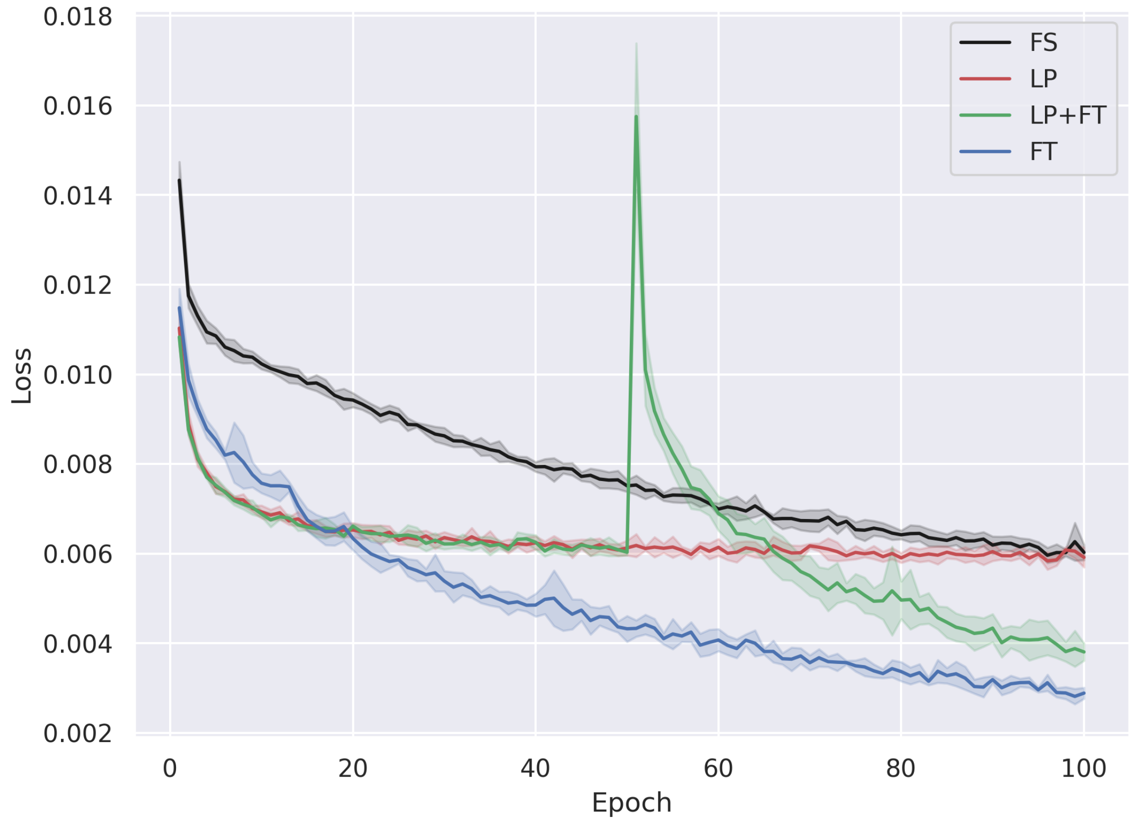


Figure 6.2: Weighted Cross-Entropy loss of ResNet-18 models with different training setups (FS, LP, LP+FT, FT) on training data over five runs. The solid line is the mean over different runs for the setup. The shaded region around the lines is the standard deviation.

with random initial weights.

Now let's explore how each setup is performing on unseen test data. We recorded the models' performance on test data for each setup with different random seeds in the last five epochs. Figure 6.5 shows the average and standard deviation of macro-average F1-scores of the models across all classes with different setups through the last five epochs of training. This gives more information about the model's performance on test partitions than taking results on the last epoch itself. We can compare the performances in more checkpoints enabling better-supported conclusions about different training setups. We see that the average macro average F1-scores of all setups with pretrained initial weights are higher than training from scratch. Besides FT setup is performing better than FS considering standard deviation. Figure 6.6 shows the average and standard deviation of macro-average F1-scores of different setups for

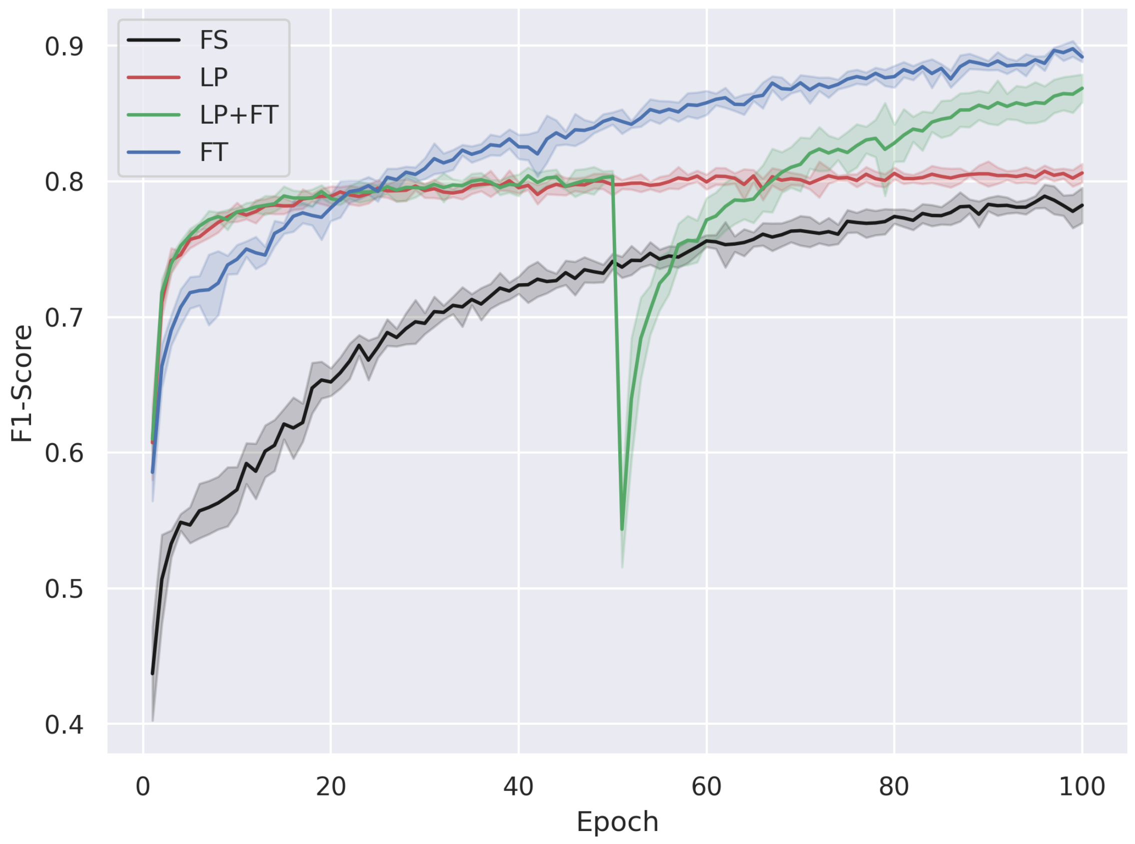


Figure 6.3: Macro-average F1-score of ResNet-18 models with different training setups (FS, LP, LP+FT, FT) on training data over five runs. The solid line is the mean over different runs for the setup. The shaded region around the lines is the standard deviation.

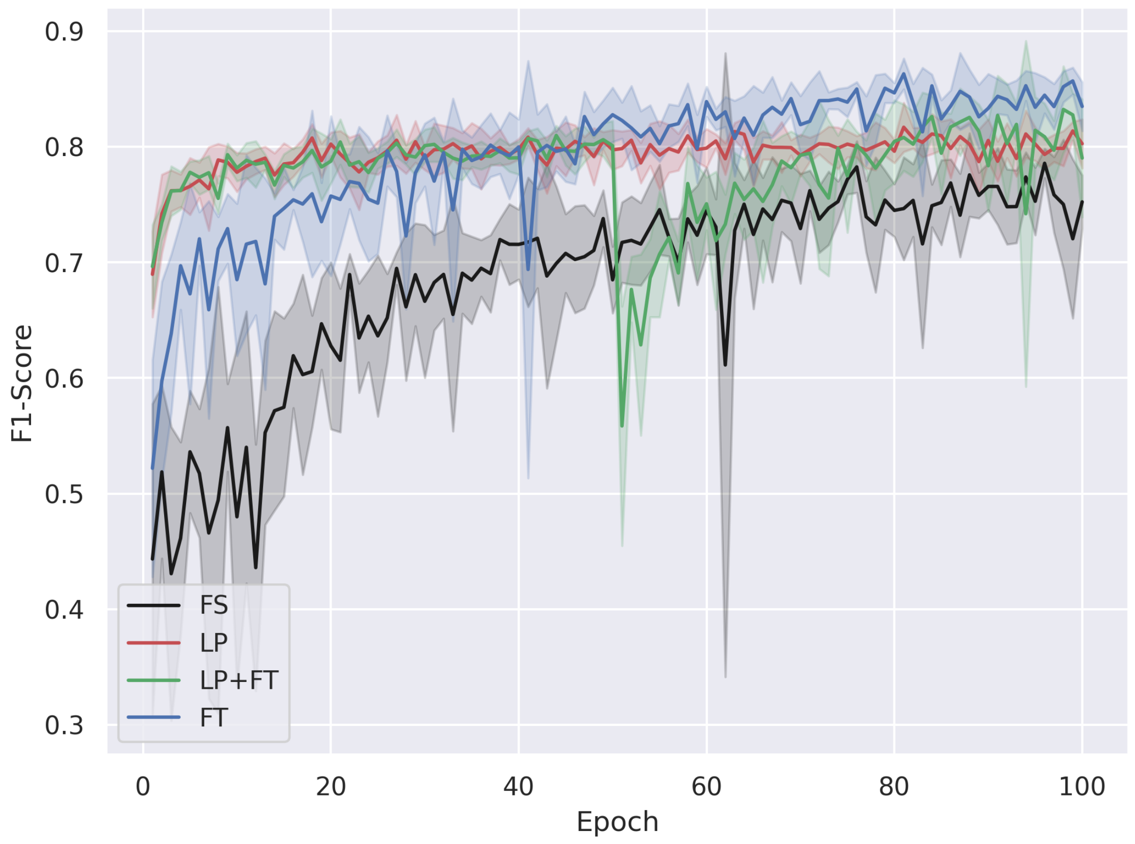
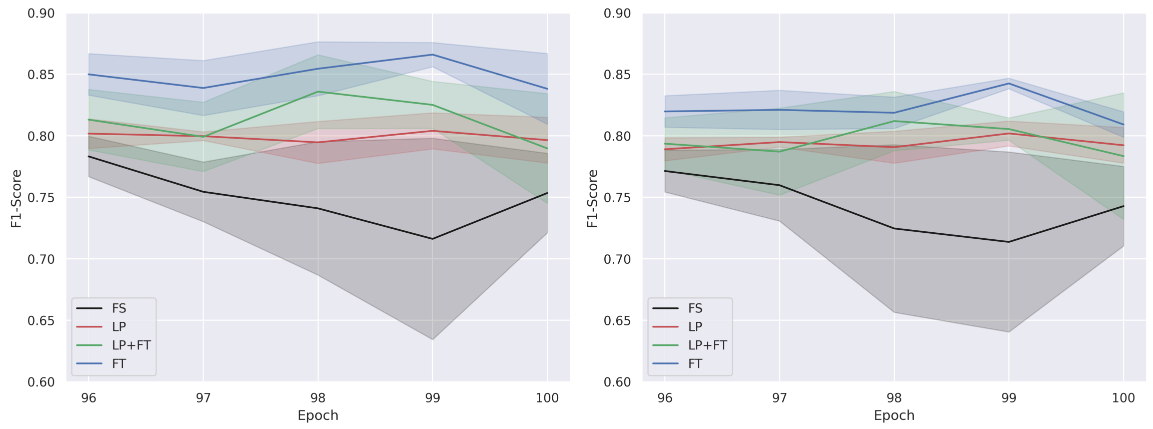


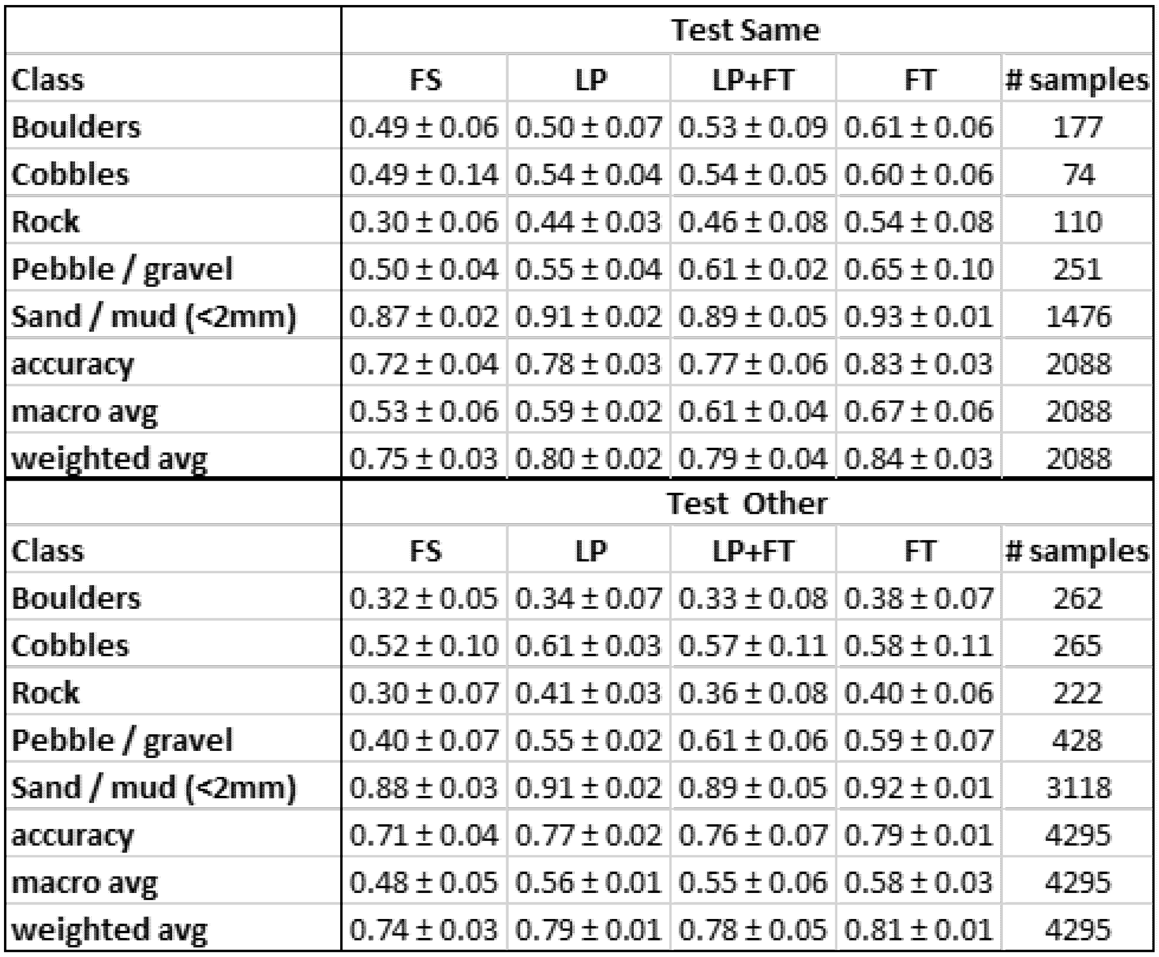
Figure 6.4: Macro-average F1-score of ResNet-18 models with different training setups (FS, LP, LP+FT, FT) on validation data over five runs. The solid line is the mean over different runs for the setup. The shaded region around the lines is the standard deviation.



(a) (b)

Figure 6.5: Macro-average F1-score of ResNet-18 models with different training setups (FS, LP, LP+FT, FT) on a) test same b) test other data in the last 5 epochs over five runs. The solid line is the mean over different runs for the setup. The shaded region around the lines is the standard deviation.

each class at the end of training. The results show the mean macro average F1-scores of models with initial pretrained weights from ImageNet are systematically higher than that of FS setup across all classes. One also can notice the standard deviation of F1-scores is smaller across all classes for both test partitions when only the last layer is trained. All setups have better macro average F1-cores ranging from 0.03 to 0.09 on the test same partition than on the test other partition. The takeaway from this experiment is transfer learning from ImageNet is useful to classify BenthicNet images.



	Test Same				
Class	FS	LP	LP+FT	FT	# samples
Boulders	0.49 ± 0.06	0.50 ± 0.07	0.53 ± 0.09	0.61 ± 0.06	177
Cobbles	0.49 ± 0.14	0.54 ± 0.04	0.54 ± 0.05	0.60 ± 0.06	74
Rock	0.30 ± 0.06	0.44 ± 0.03	0.46 ± 0.08	0.54 ± 0.08	110
Pebble / gravel	0.50 ± 0.04	0.55 ± 0.04	0.61 ± 0.02	0.65 ± 0.10	251
Sand / mud (<2mm)	0.87 ± 0.02	0.91 ± 0.02	0.89 ± 0.05	0.93 ± 0.01	1476
accuracy	0.72 ± 0.04	0.78 ± 0.03	0.77 ± 0.06	0.83 ± 0.03	2088
macro avg	0.53 ± 0.06	0.59 ± 0.02	0.61 ± 0.04	0.67 ± 0.06	2088
weighted avg	0.75 ± 0.03	0.80 ± 0.02	0.79 ± 0.04	0.84 ± 0.03	2088
	Test Other				
Class	FS	LP	LP+FT	FT	# samples
Boulders	0.32 ± 0.05	0.34 ± 0.07	0.33 ± 0.08	0.38 ± 0.07	262
Cobbles	0.52 ± 0.10	0.61 ± 0.03	0.57 ± 0.11	0.58 ± 0.11	265
Rock	0.30 ± 0.07	0.41 ± 0.03	0.36 ± 0.08	0.40 ± 0.06	222
Pebble / gravel	0.40 ± 0.07	0.55 ± 0.02	0.61 ± 0.06	0.59 ± 0.07	428
Sand / mud (<2mm)	0.88 ± 0.03	0.91 ± 0.02	0.89 ± 0.05	0.92 ± 0.01	3118
accuracy	0.71 ± 0.04	0.77 ± 0.02	0.76 ± 0.07	0.79 ± 0.01	4295
macro avg	0.48 ± 0.05	0.56 ± 0.01	0.55 ± 0.06	0.58 ± 0.03	4295
weighted avg	0.74 ± 0.03	0.79 ± 0.01	0.78 ± 0.05	0.81 ± 0.01	4295

Figure 6.6: F1-score statistics of ResNet-18 models different training setups (FS, LP, LP+FT, FT) on test partitions at the end of training across classes

Chapter 7

Self-Supervised learning on BenthicNet

In this chapter, we explore the quality of learned representation through self-supervised learning on our classification task. We continue using the same dataset and partitioning from the previous chapter. As we want to evaluate learned representations, we continue our experiments in three setups: LP, LP+FT, and FT leaving out FS. We start with comparing models' performance when initialized with pretrained weights on ImageNet with supervised learning(SLI) and self-supervised learning(SSLI). We take weights of the pretrained model by self-supervised learning from Sololearn [7]. The method used for SSL learning is MoCo v2 which was discussed in the background section. As the ImageNet pretrained checkpoint for the SSL model is ResNet-50, we run all following experiments on this model architecture for a fair comparison. We are keeping training hyper-parameters for supervised learning (e.g batch size, learning rate, number of training epochs, etc) unchanged from the previous chapter as they worked for new architecture. The augmentations for training partition along with standardization for validation and test partitions are kept the same. However, we decreased the learning rate for fine-tuning after linear probe training ten times to 2e−4 so that the model does not destroy learned representations when unfreezing all layers. All setups have five different runs with different seeds from 0 to 4 as in the previous chapter.

Figure 7.1 shows the Cross-Entropy loss of the models on training data. It is noticeable that FT has a bigger loss for both SSLI and SLI at the beginning of the training as it is tuning the whole network by changing more parameters resulting in a bigger loss. At the end of the training, it is getting smaller than LP for the same reason of tuning more parameters has a bigger effect on loss. Interestingly, LP+FT has a very similar loss in the second half of the training despite a noticeable difference in the first half. Besides LP+FT is getting a much smaller loss than LP or FT itself. The decrease in the learning rate when the whole network is unfrozen made a huge

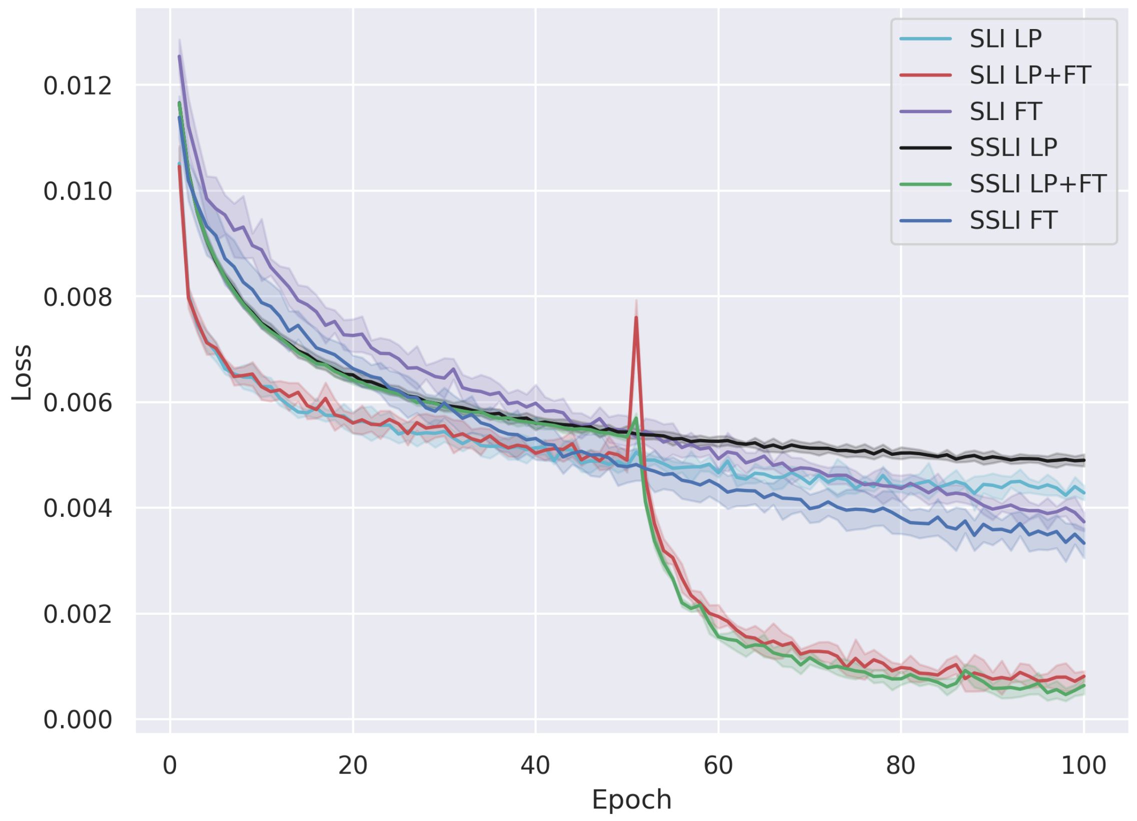


Figure 7.1: Weighted Cross-Entropy loss of ResNet-50 models with different training setups (FS, LP, LP+FT, FT) and with different initial pretrained weights on ImageNet, by supervised (SLI) and self-supervised learning (SSLI), on training data over five runs. The solid line is the mean over different runs for the setup. The shaded region around the lines is the standard deviation.

difference compared to the loss graph for the experiment from the last chapter. We still observe a smaller bump in the loss that we can address by further decreasing the learning rate for FT.

Figures 7.2 and 7.3 represent macro average F1-scores of different setups on training and validation data. In both figures, it can be seen LP+FT is the best-performing setup. However, the performance of models with initial SSL and SL pretrained weights are very close to each other.

Now let's look at the models' performances on unseen data. As SSL tries to learn task-independent representations we expect it to perform better on the test other partition. Table 7.4 shows the average and standard deviation of the F1-score over different runs for each setup at the end of training. The first thing to notice is LP+FT

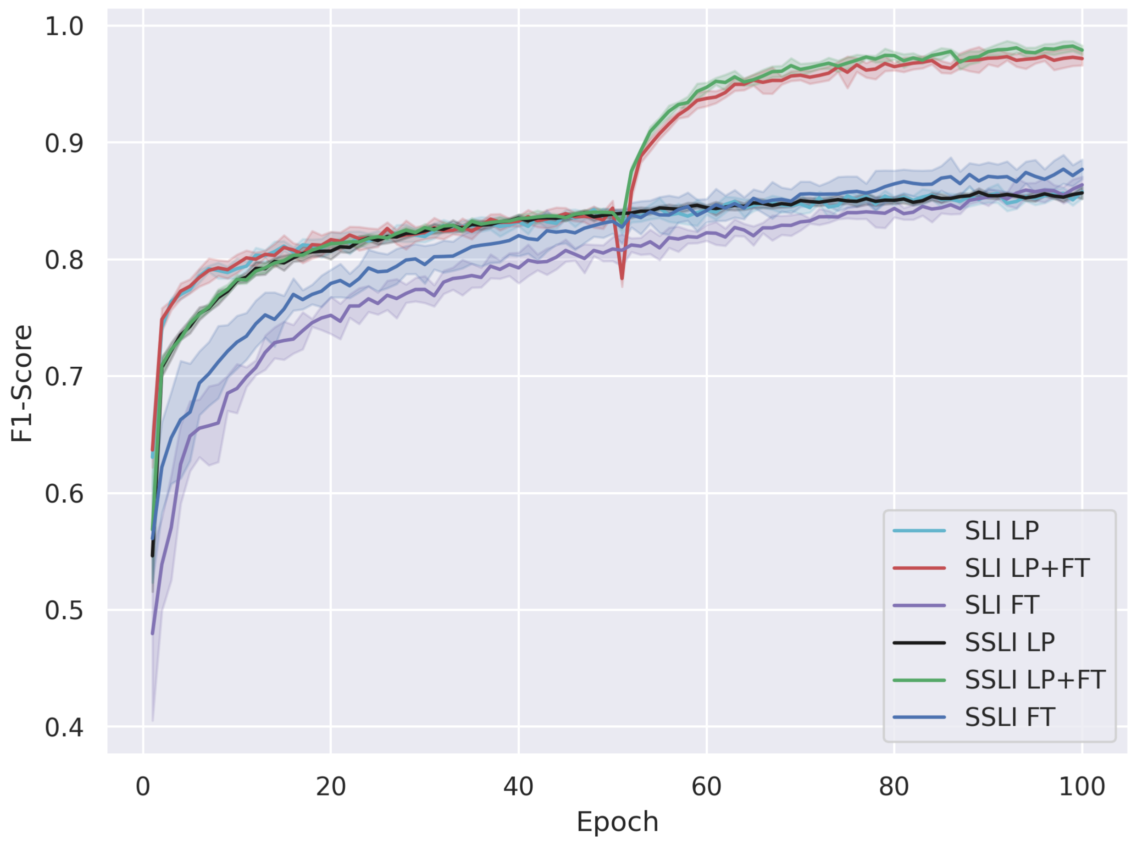


Figure 7.2: Macro-average F1-score of ResNet-50 models with different training setups (FS, LP, LP+FT, FT) and with different initial pretrained weights on ImageNet, by supervised (SLI) and self-supervised learning (SSLI), on training data over five runs. The solid line is the mean over different runs for the setup. The shaded region around the lines is the standard deviation.

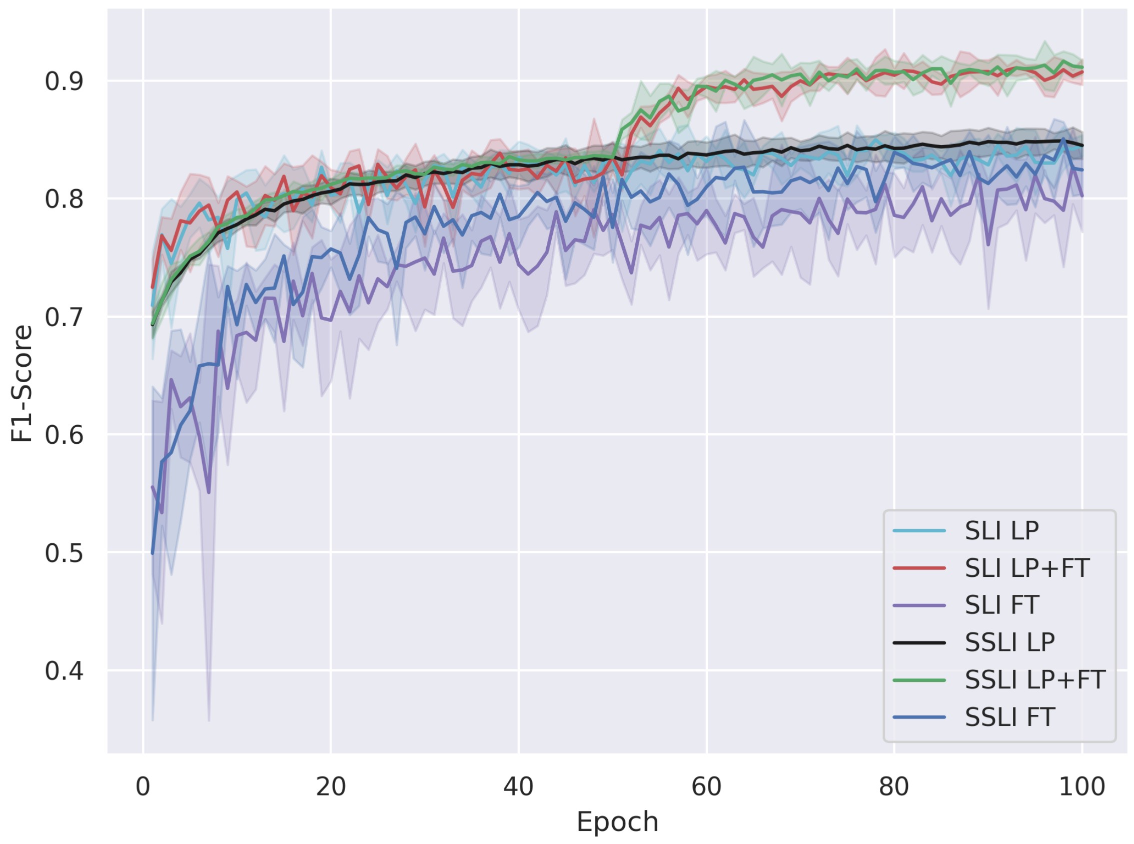
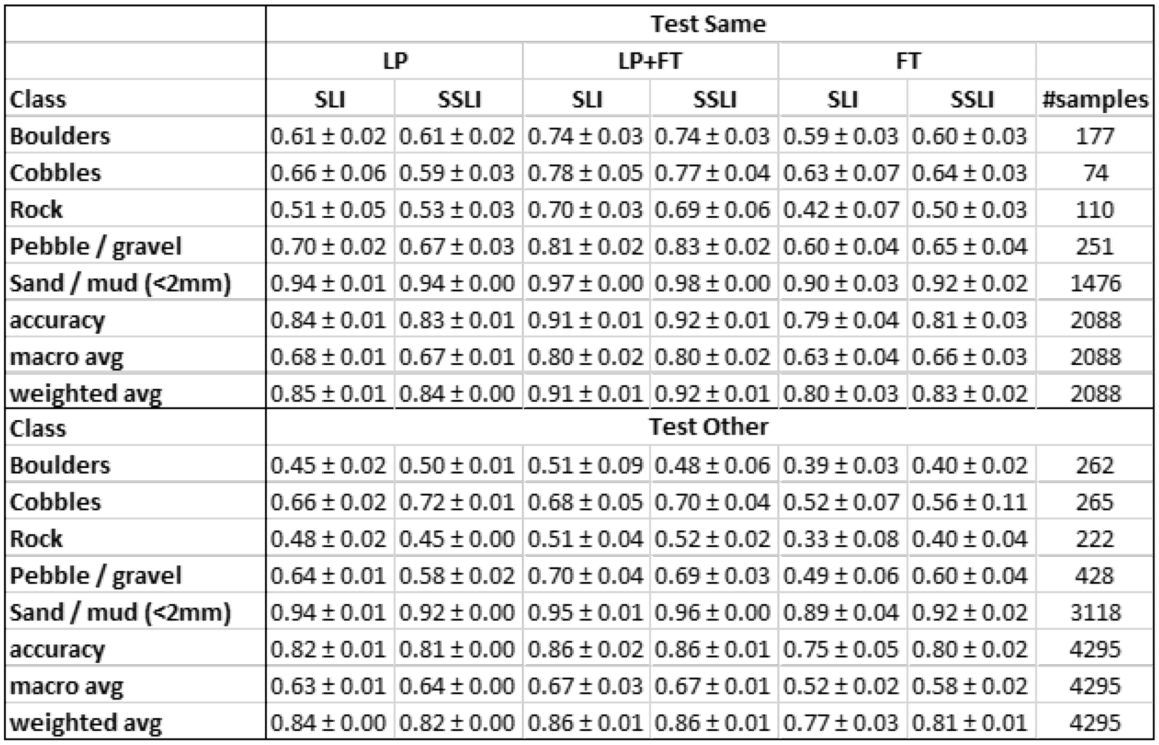


Figure 7.3: Macro-average F1-score of ResNet-50 models with different training setups (FS, LP, LP+FT, FT) and with different initial pretrained weights on ImageNet, by supervised (SLI) and self-supervised learning (SSLI), on validation data over five runs. The solid line is the mean over different runs for the setup. The shaded region around the lines is the standard deviation.

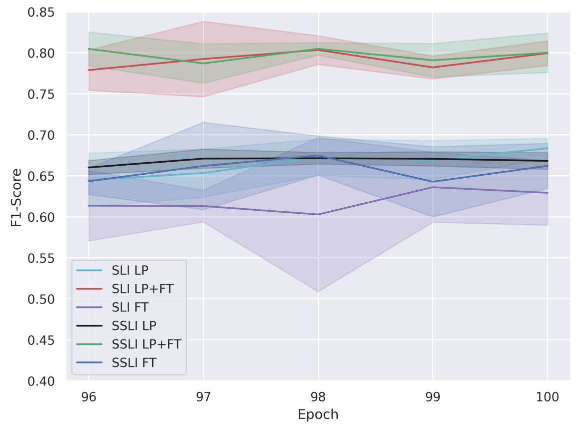


	Test Same						
	LP		LP+FT		FT		
Class	SLI	SSLI	SLI	SSLI	SLI	SSLI	#samples
Boulders	0.61 ± 0.02	0.61 ± 0.02	0.74 ± 0.03	0.74 ± 0.03	0.59 ± 0.03	0.60 ± 0.03	177
Cobbles	0.66 ± 0.06	0.59 ± 0.03	0.78 ± 0.05	0.77 ± 0.04	0.63 ± 0.07	0.64 ± 0.03	74
Rock	0.51 ± 0.05	0.53 ± 0.03	0.70 ± 0.03	0.69 ± 0.06	0.42 ± 0.07	0.50 ± 0.03	110
Pebble / gravel	0.70 ± 0.02	0.67 ± 0.03	0.81 ± 0.02	0.83 ± 0.02	0.60 ± 0.04	0.65 ± 0.04	251
Sand / mud (<2mm)	0.94 ± 0.01	0.94 ± 0.00	0.97 ± 0.00	0.98 ± 0.00	0.90 ± 0.03	0.92 ± 0.02	1476
accuracy	0.84 ± 0.01	0.83 ± 0.01	0.91 ± 0.01	0.92 ± 0.01	0.79 ± 0.04	0.81 ± 0.03	2088
macro avg	0.68 ± 0.01	0.67 ± 0.01	0.80 ± 0.02	0.80 ± 0.02	0.63 ± 0.04	0.66 ± 0.03	2088
weighted avg	0.85 ± 0.01	0.84 ± 0.00	0.91 ± 0.01	0.92 ± 0.01	0.80 ± 0.03	0.83 ± 0.02	2088
Class	Test Other						
Boulders	0.45 ± 0.02	0.50 ± 0.01	0.51 ± 0.09	0.48 ± 0.06	0.39 ± 0.03	0.40 ± 0.02	262
Cobbles	0.66 ± 0.02	0.72 ± 0.01	0.68 ± 0.05	0.70 ± 0.04	0.52 ± 0.07	0.56 ± 0.11	265
Rock	0.48 ± 0.02	0.45 ± 0.00	0.51 ± 0.04	0.52 ± 0.02	0.33 ± 0.08	0.40 ± 0.04	222
Pebble / gravel	0.64 ± 0.01	0.58 ± 0.02	0.70 ± 0.04	0.69 ± 0.03	0.49 ± 0.06	0.60 ± 0.04	428
Sand / mud (<2mm)	0.94 ± 0.01	0.92 ± 0.00	0.95 ± 0.01	0.96 ± 0.00	0.89 ± 0.04	0.92 ± 0.02	3118
accuracy	0.82 ± 0.01	0.81 ± 0.00	0.86 ± 0.02	0.86 ± 0.01	0.75 ± 0.05	0.80 ± 0.02	4295
macro avg	0.63 ± 0.01	0.64 ± 0.00	0.67 ± 0.03	0.67 ± 0.01	0.52 ± 0.02	0.58 ± 0.02	4295
weighted avg	0.84 ± 0.00	0.82 ± 0.00	0.86 ± 0.01	0.86 ± 0.01	0.77 ± 0.03	0.81 ± 0.01	4295

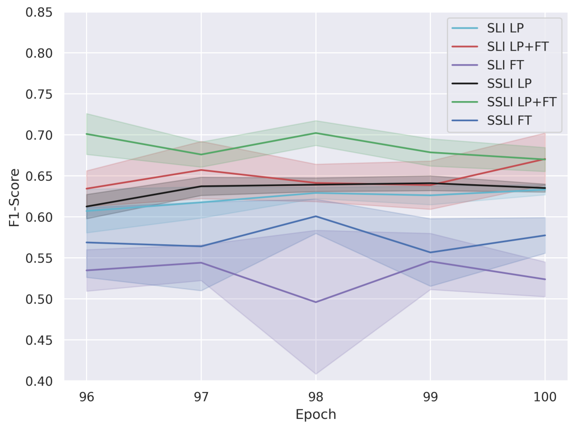
Figure 7.4: F1-score statistics of ResNet-18 models different training setups (FS, LP, LP+FT, FT) and with different initial pretrained weights on ImageNet, by supervised (SLI) and self-supervised learning (SSLI), on test partitions at the end of training across classes

is performing much better than the other two setups across all classes. It shows the importance of training the changed layer before fully training the whole model with a new dataset when transfer learning. Performances of the model with initial SSL and SL ImageNet pretrained weights are similar on all partitions considering standard deviation. SSLI LP models have the smallest standard deviation for all classes across runs showing the robustness of SSL-learned representations for different runs.

Figure 7.5 shows the mean macro average F1-score of models on test data over different runs in the last five epochs of training. The left graph clearly shows the superiority of LP+FT on test data with the same distribution as training data, while the difference gets smaller on out-of-distribution test other data on the right. Besides each setup individually performs better on the test same dataset than the test other data. The results on test data show no advantage of self-supervised learned representation over supervised learned representations on ImageNet.



(a)



(b)

Figure 7.5: Macro-average F1-score of ResNet-50 models with different training setups (FS, LP, LP+FT, FT) and with different initial pretrained weights on ImageNet, by supervised (SLI) and self-supervised learning (SSLI), on a) test same b) test other data in the last 5 epochs over five runs. The solid line is the mean over different runs for the setup. The shaded region around the lines is the standard deviation.

In the second part of the experiment, we train a self-supervised model on our BenthicNet dataset and further fine-tune learned representations with our whole frame depth 3 substrate-labeled dataset. We use MoCo v2 method for SSL on BenthicNet with same hyperparatmeters for ImageNet training. As the dataset contains 4k resolution images, one of the bottlenecks of training a model is reading images from a disk for inputting into the model. As we are resizing images to a smaller size for training, we are first resizing all images to 512 in smaller dimensional saving them to the disk, so that we can load them faster from the disk. We are using resources from The Digital Research Alliance of Canada for our experiments that require a job submission with required resources for the needed node. When training a model, it needs to read all images through the network from its original location to the training node's memory which is very time-consuming. To address this issue we are caching images to the local memory of the node in the first epoch of training so that the data loading speed is faster for preceding epochs. We trained the model for 100 epochs (same with ImageNet SSL pretrained model) and the loss graph during training is given in Figure 7.6. The training took little more than 8 days with 4 NVIDIA Tesla V100 (32 GB variant) GPUs.

When transfer learning from BenthicNet pretrained weights we used the same

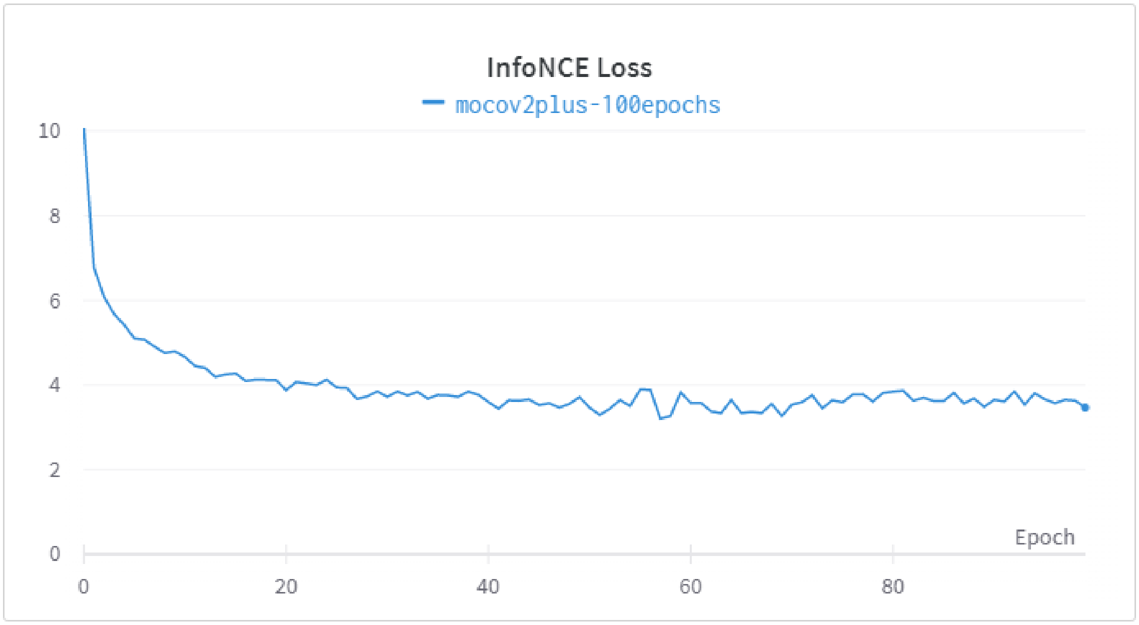


Figure 7.6: InfoNCE Loss of MoCo v2 self-supervised method on BenthicNet

three setups as in the previous part: LP, LP+FT, and FT. We refer to models with initial BenthicNet learned representations as SSLB. When training a model with LP+FT we used a learning rate of $2e-5$ which enabled a smooth transition in the training loss curve when unfreezing all layers halfway through training. We retrained a model using transfer learning from ImageNet pretrained model with this learning rate. This helps a model not to deteriorate learned representation when we start fine-tuning after initial linear probe training. Figure 7.7 shows the mean and standard deviation of the cross entropy loss over different runs for each setup during training. It is clear from the figure that the transition from LP to FT for LP+FT is smooth for both SSLI and SSLB for the chosen learning rate. Besides cross-entropy loss is very similar for both models, with ImageNet and BenthicNet pretrained initial weights, in LP setup during training. Solely looking at the loss curve, SSLI representations are more useful for our classification task as the corresponding models have the smallest loss.

The figures 7.8 and 7.9 show models' performances on training and validation data. The graphs support the superiority of LP+FT and learned representations from ImageNet are working better for our classification task as this setup has a higher F1-score than others considering standard deviation. SSLI LP has smoothest

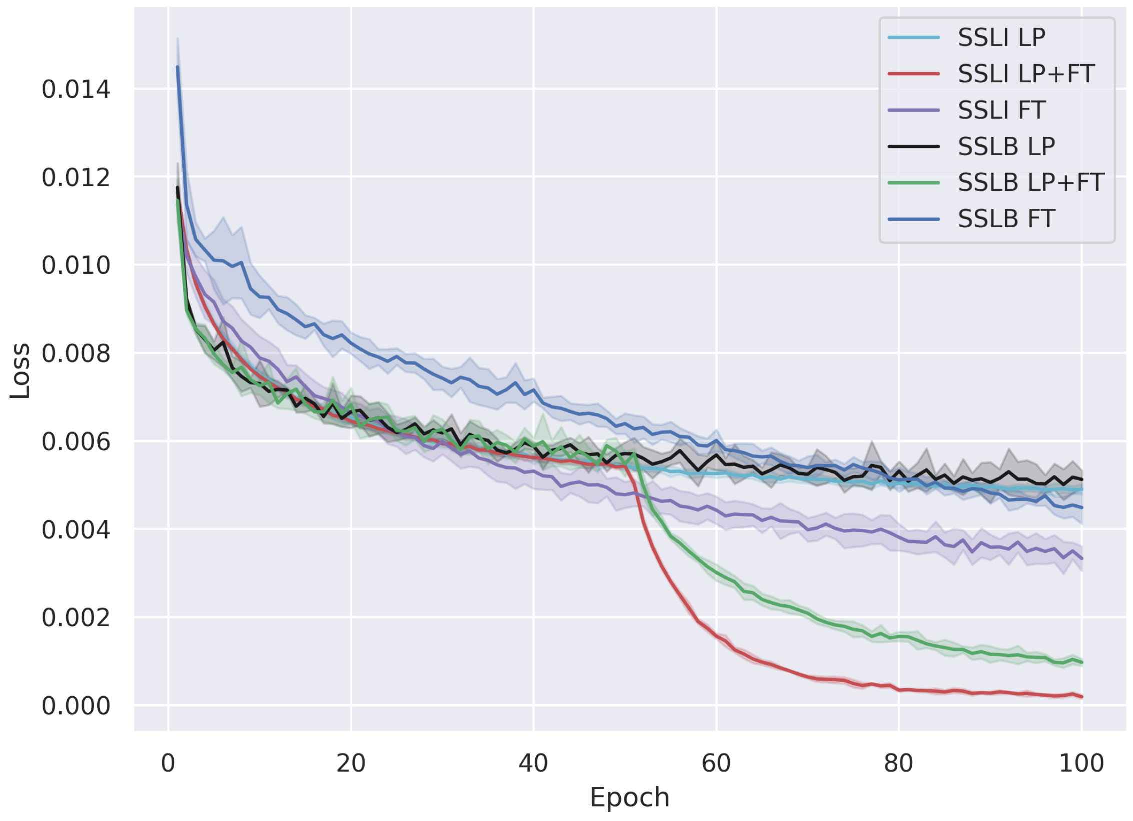


Figure 7.7: Weighted Cross-Entropy loss of ResNet-50 models with different training setups (FS, LP, LP+FT, FT) and with different initial SSL pretrained weights on ImageNet (SSLI) and BenthicNet (SSLB) on training data over five runs. The solid line is the mean over different runs for the setup. The shaded region around the lines is the standard deviation.

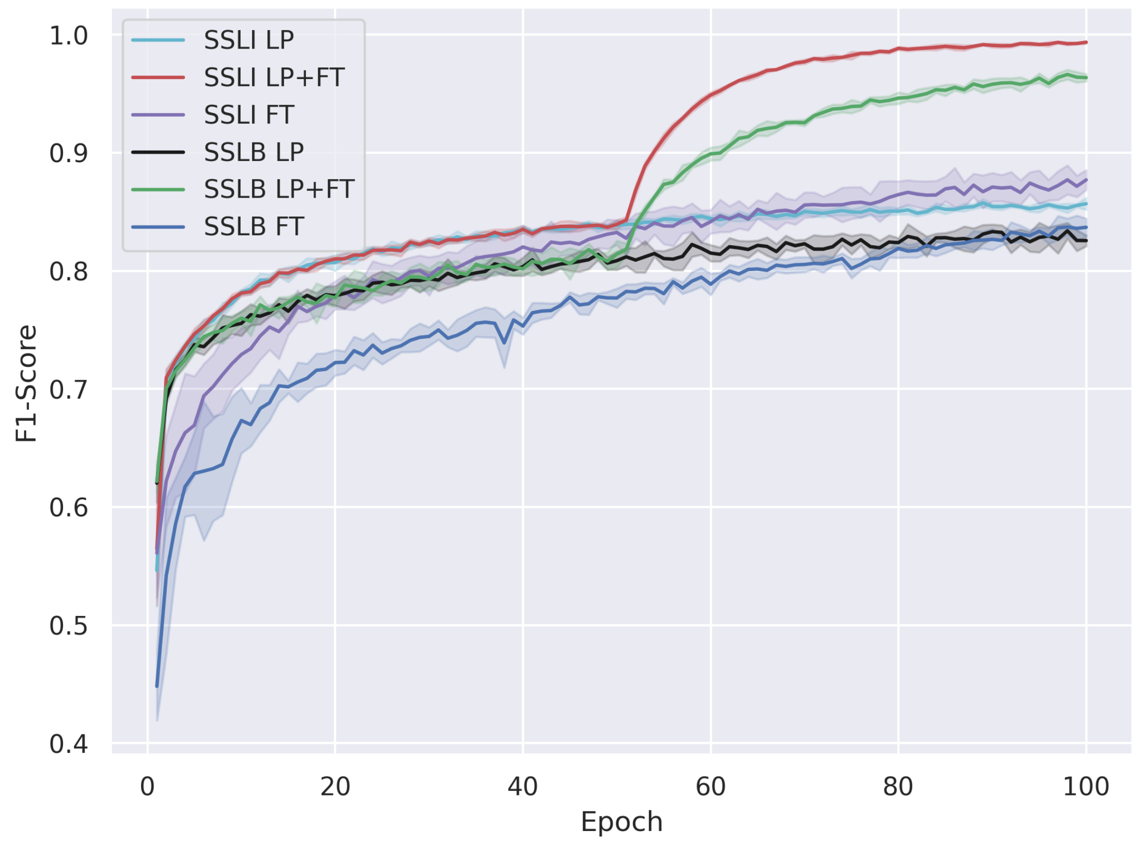


Figure 7.8: Macro-average F1-score of ResNet-50 models with different training setups (FS, LP, LP+FT, FT) and with different initial SSL pretrained weights on ImageNet (SSLI) and BenthicNet (SSLB) on training data over five runs. The solid line is the mean over different runs for the setup. The shaded region around the lines is the standard deviation.

F1 curve in both training and validation partitions. Furthermore, SSLI is performing better than SSLB for all three setups on training and validation data.

Figure 7.10 shows the performance of different setups on test partitions. SSLI LP+FT is performing best in both partitions. SSLB LP+FT is performing better than LP and FT setups on the test same partition, however, it performs similarly to SSLI LP setup in the test other partition. The graph for the test other partition also shows SSL learned representations from ImageNet is more useful without fine-tuning as SSLI LP performs better than SSLB LP.

Figure 7.11 shows mean and standard deviation of F1-scores for different classes at the end of training. SSLI LP+FT setup is performing best in both partitions in all classes. It is achieving a mean F1-score of 0.78 and 0.73 on test same and other

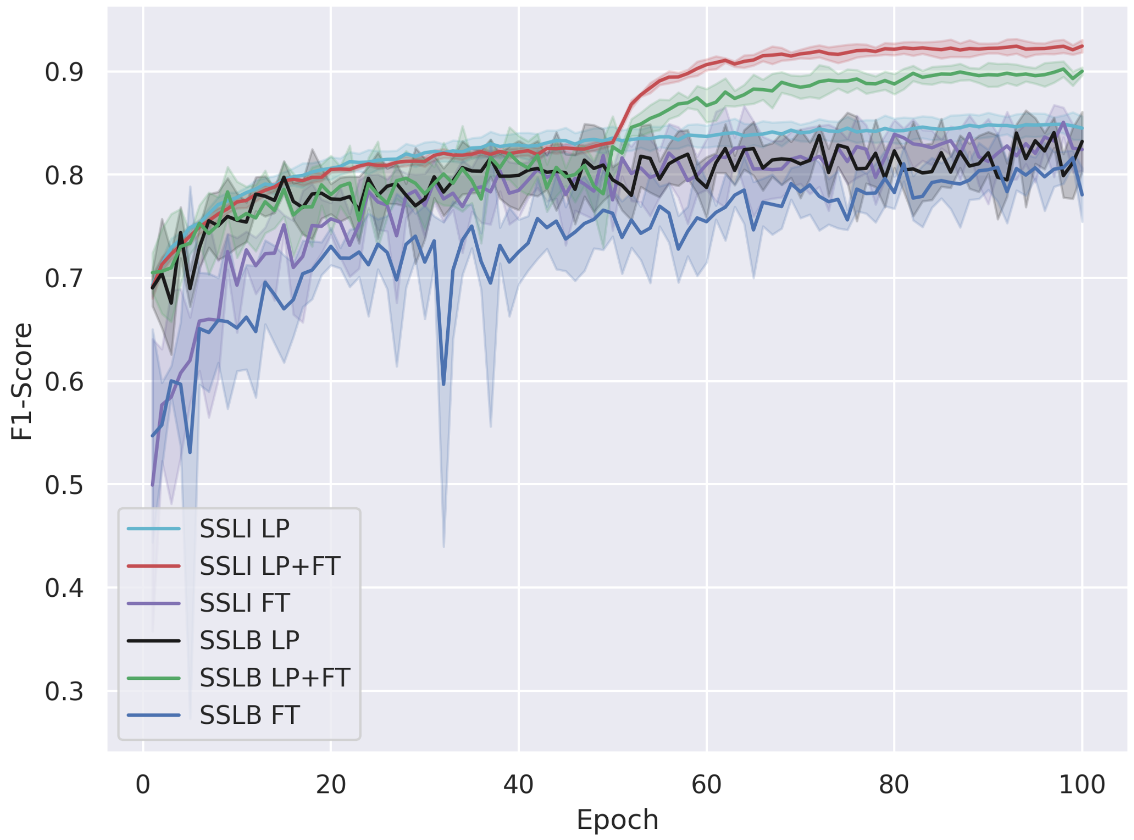
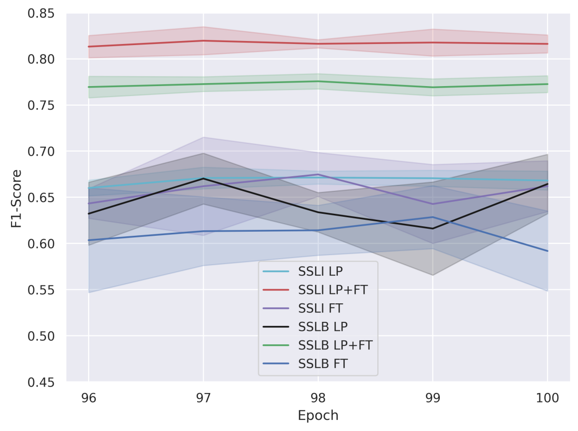
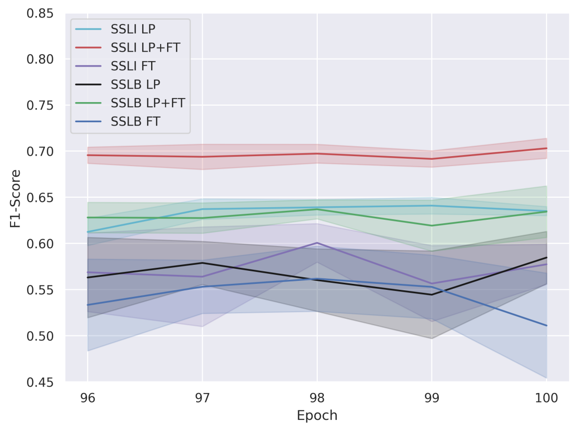


Figure 7.9: Macro-average F1-score of ResNet-50 models with different training setups (FS, LP, LP+FT, FT) and with different initial SSL pretrained weights on ImageNet (SSLI) and BenthicNet (SSLB) on validation data over five runs. The solid line is the mean over different runs for the setup. The shaded region around the lines is the standard deviation.



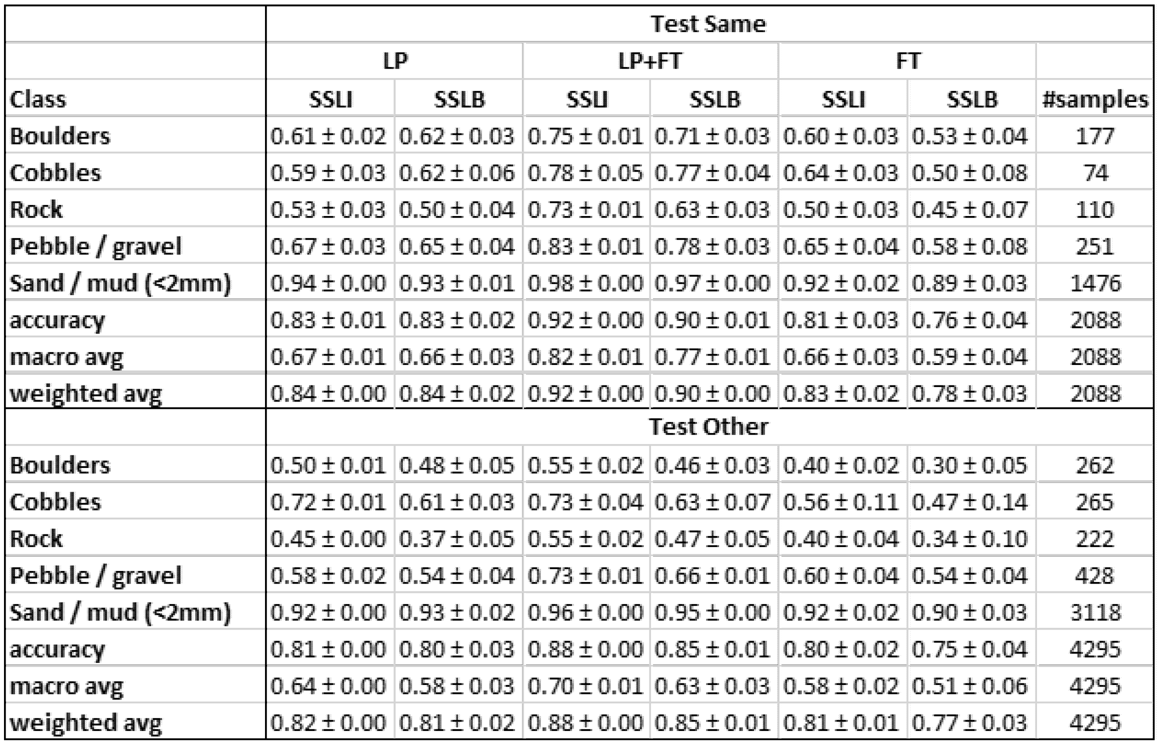
(a)



(b)

Figure 7.10: Macro-average F1-score of ResNet-50 models with different training setups (FS, LP, LP+FT, FT) and with different initial SSL pretrained weights on ImageNet (SSLI) and BenthicNet (SSLB) on a) test same b) test other data in the last 5 epochs over five runs. The solid line is the mean over different runs for the setup. The shaded region around the lines is the standard deviation.

partitions respectively on minority class ”Cobbles”. It is also achieving 0.98 and 0.96 on the test same and other partitions respectively on the majority class ”Sand/mud”. SSLB LP+FT is achieving the second-best F1 score across all classes on the test same data. SSLB FT setup is showing the lowest F1-scores in both test partitions across all classes. Furthermore, SSLI models have smaller standard deviations than SSLB models in all setups across all classes in both test partitions. This shows ImageNet learned representations are more robust to different runs for our classification task.



	Test Same						
	LP		LP+FT		FT		
Class	SSLI	SSLB	SSLI	SSLB	SSLI	SSLB	#samples
Boulders	0.61 ± 0.02	0.62 ± 0.03	0.75 ± 0.01	0.71 ± 0.03	0.60 ± 0.03	0.53 ± 0.04	177
Cobbles	0.59 ± 0.03	0.62 ± 0.06	0.78 ± 0.05	0.77 ± 0.04	0.64 ± 0.03	0.50 ± 0.08	74
Rock	0.53 ± 0.03	0.50 ± 0.04	0.73 ± 0.01	0.63 ± 0.03	0.50 ± 0.03	0.45 ± 0.07	110
Pebble / gravel	0.67 ± 0.03	0.65 ± 0.04	0.83 ± 0.01	0.78 ± 0.03	0.65 ± 0.04	0.58 ± 0.08	251
Sand / mud (<2mm)	0.94 ± 0.00	0.93 ± 0.01	0.98 ± 0.00	0.97 ± 0.00	0.92 ± 0.02	0.89 ± 0.03	1476
accuracy	0.83 ± 0.01	0.83 ± 0.02	0.92 ± 0.00	0.90 ± 0.01	0.81 ± 0.03	0.76 ± 0.04	2088
macro avg	0.67 ± 0.01	0.66 ± 0.03	0.82 ± 0.01	0.77 ± 0.01	0.66 ± 0.03	0.59 ± 0.04	2088
weighted avg	0.84 ± 0.00	0.84 ± 0.02	0.92 ± 0.00	0.90 ± 0.00	0.83 ± 0.02	0.78 ± 0.03	2088
	Test Other						
Boulders	0.50 ± 0.01	0.48 ± 0.05	0.55 ± 0.02	0.46 ± 0.03	0.40 ± 0.02	0.30 ± 0.05	262
Cobbles	0.72 ± 0.01	0.61 ± 0.03	0.73 ± 0.04	0.63 ± 0.07	0.56 ± 0.11	0.47 ± 0.14	265
Rock	0.45 ± 0.00	0.37 ± 0.05	0.55 ± 0.02	0.47 ± 0.05	0.40 ± 0.04	0.34 ± 0.10	222
Pebble / gravel	0.58 ± 0.02	0.54 ± 0.04	0.73 ± 0.01	0.66 ± 0.01	0.60 ± 0.04	0.54 ± 0.04	428
Sand / mud (<2mm)	0.92 ± 0.00	0.93 ± 0.02	0.96 ± 0.00	0.95 ± 0.00	0.92 ± 0.02	0.90 ± 0.03	3118
accuracy	0.81 ± 0.00	0.80 ± 0.03	0.88 ± 0.00	0.85 ± 0.01	0.80 ± 0.02	0.75 ± 0.04	4295
macro avg	0.64 ± 0.00	0.58 ± 0.03	0.70 ± 0.01	0.63 ± 0.03	0.58 ± 0.02	0.51 ± 0.06	4295
weighted avg	0.82 ± 0.00	0.81 ± 0.02	0.88 ± 0.00	0.85 ± 0.01	0.81 ± 0.01	0.77 ± 0.03	4295

Figure 7.11: F1-score statistics of ResNet-18 models different training setups (FS, LP, LP+FT, FT) and with different initial SSL pretrained weights on ImageNet (SSLI) and BenthicNet (SSLB) on test partitions at the end of training across classes

Chapter 8

Conclusion & Future Work

Benthic habitats span large area underwater. The adjacent images or frames in the video can represent very similar benthic habitats causing spatial redundancy in the images. The inclusion of the first image in the training partition and the following image in the test partition can question the generalization of the model. Considering this, it is crucial to partition the benthic dataset either randomly or spatially according to the use case of the model. We found a gap in model performance on different test partitions in our all experiments. In case the model is to be used to label images from a geographical location that is trained with images from, it is acceptable to partition the dataset randomly as we did with the test same partition. However, if the user wants to use the model to classify images from the area which is not used for training the model, the model's generalization on random test data does not infer the model can extrapolate on unseen data we want. In this scenario, it is important to choose a test dataset spatially facilitating that no two images very similar to each other are on the training and test partition.

The results of the experiment in Chapter 6 show that when the labeled data is few to train a model from scratch the learned representations from ImageNet are useful. It also showed the importance of training new layers in the network before fine-tuning the whole network when using transfer learning.

Another finding in our work was getting similar performance for models when transferring learning from ImageNet learned representations by supervised and self-supervised learning. However, the models with initial ImageNet pre-trained weights by SSL were more robust to different runs with different random seeds supporting the advantage of using ImageNet SSL pretrained model for benthic image classification for transfer learning from ImageNet.

Benthic images are different from ImageNet images in terms of lighting conditions and clarity due to the water between the object and the camera. That is why we assumed the SSL model trained on the whole BenthicNet dataset and fine-tuned for the labeled dataset should perform better than transfer learning from ImageNet. However, the last experiment, Chapter 7, showed SSL learned representations on ImageNet are more useful for our classification task than BenthicNet.

Our results also showed that models are performing better to classify sand/mud than other classes (e.g., rock, cobbles) as sand/mud has more distinguishable shape features from other classes than other classes do from each other. One way to make low-performing classes more distinguishable from each other for the model could be integrating camera resolution for each image into training so that model has more information on physical element size which is the main differentiating factor among classes. This hypothesis needs further exploration in future work.

Another forward step in this work would be creating a more complex task to explore the advantage of SSL-learned representations from BenthicNet. As the labeled dataset has a hierarchical structure and we created a simplified task of classifying substrate at a level depth of three, a more complex task can be hierarchical classification on the whole dataset for all four label types: substrate, relief, bedform, and biota. Then one can investigate the effect of different self-supervised methods on the quality of learned representations.

Bibliography

[1] Franziska Althaus, Nicole Hill, Luke Edwards, and Renata Ferrari Legorreta. Catami class pdfguide v4 20141218, 12 2013.

[2] Gail M. Ashley. Classification of large-scale subaqueous bedforms; a new look at an old problem. *Journal of Sedimentary Research*, 60:160–172, 1990.

[3] Asma Bahrani, Babak Majidi, and Mohammad Eshghi. Coral reef management in persian gulf using deep convolutional neural networks. *2019 4th International Conference on Pattern Recognition and Image Analysis (IPRIA)*, pages 200–204, 2019.

[4] Oscar Beijbom, Peter J. Edmunds, David I. Kline, B. Greg Mitchell, and David Kriegman. Automated annotation of coral reef survey images. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1170–1177, 2012.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[6] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

[7] Victor Guilherme Turrisi da Costa, Enrico Fini, Moin Nabi, Nicu Sebe, and Elisa Ricci. solo-learn: A library of self-supervised methods for visual representation learning. *Journal of Machine Learning Research*, 23(56):1–6, 2022.

[8] André Diegues, José Pinto, Pedro Ribeiro, Roberto Frias, and do Campo Alegre. Automatic habitat mapping using convolutional neural networks. In *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, pages 1–6, 2018.

[9] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 184–199, Cham, 2014. Springer International Publishing.

[10] Mohamed Elawady. Sparse coral classification using deep convolutional neural networks. *CoRR*, abs/1511.09067, 2015.

[11] William Falcon et al. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3(6), 2019.

[12] Adam Gauci, Alan Deidun, John Abela, and Kristian Adami. Machine learning for benthic sand and maerl classification and coverage estimation in coastal areas around the maltese islands. *Journal of Applied Research and Technology*, 14, 10 2016.

[13] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[14] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *CoRR*, abs/2006.07733, 2020.

[15] Anabel Gómez-Ríos, Siham Tabik, Julián Luengo, ASM Shihavuddin, Bartosz Krawczyk, and Francisco Herrera. Towards highly accurate coral texture images classification using deep convolutional neural networks and data augmentation. *Expert Systems with Applications*, 118:315–328, 2019.

[16] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv:2111.06377*, 2021.

[17] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, 2020.

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[20] Mi-Young Huh, Pulkit Agrawal, and Alexei A. Efros. What makes imagenet good for transfer learning? *CoRR*, abs/1608.08614, 2016.

[21] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[22] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018.

[23] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[25] Myriam Lacharité, Craig Brown, and Vicki Gazzola. Multisource multibeam backscatter data: developing a strategy for the production of benthic habitat maps using semi-automated seafloor classification methods. *Marine Geophysical Research*, 39, 06 2018.

[26] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[27] Scott C. Lowe, Benjamin Misiuk, Shakhboz Abdulazizov, Amit R. Baroi, Craig J. Brown, and Thomas Trappenberg. Benthicnet: A global dataset of seafloor photography for deep learning applications. Manuscript in preparation.

[28] Subhransu Maji, Alexander C. Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[30] M. Vimal Raj and S. Sakthivel Murugan. Underwater image classification using machine learning technique. In *2019 International Symposium on Ocean Technology (SYMPOL)*, pages 166–173, 2019.

[31] Jimmy S. J. Ren and Li Xu. On vectorization of deep convolutional neural networks for vision tasks. *CoRR*, abs/1501.07338, 2015.

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115, 09 2014.

[34] Edgar Schönfeld, Bernt Schiele, and Anna Khoreva. A u-net based discriminator for generative adversarial networks. *CoRR*, abs/2002.12655, 2020.

[35] A.S.M. Shihavuddin, Nuno Gracias, Rafael Garcia, Arthur C. R. Gleason, and Brooke Gintert. Image-based coral reef classification and thematic mapping. *Remote Sensing*, 5(4):1809–1841, 2013.

[36] K Simonyan and A Zisserman. Very deep convolutional networks for large-scale image recognition. pages 1–14. Computational and Biological Learning Society, 2015.

[37] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017.

[38] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[39] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[40] Alexander Toshev and Christian Szegedy. DeepPose: Human pose estimation via deep neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2014.

[41] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

[42] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.