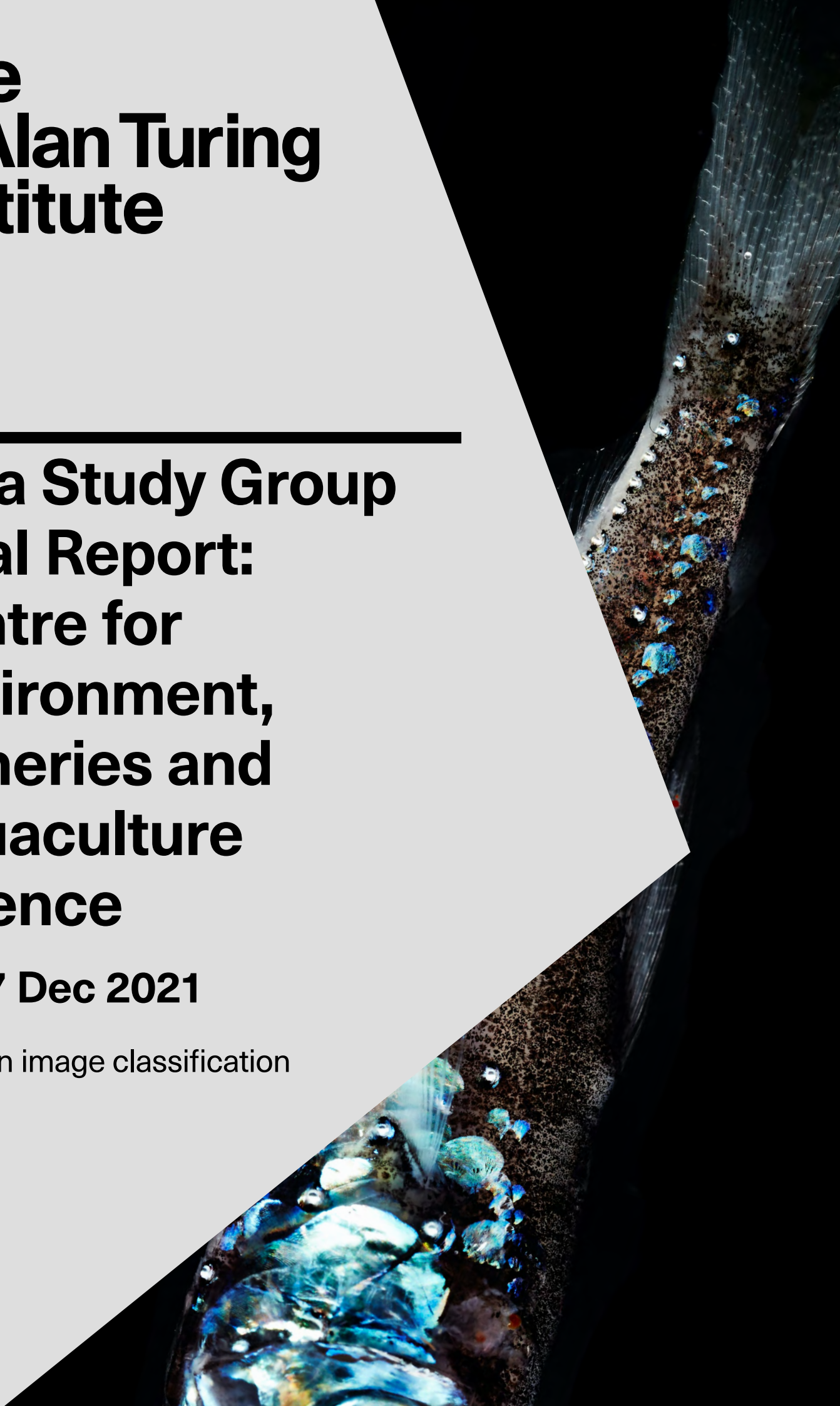
The Alan Turing Institute

Data Study Group Final Report: Centre for Environment, Fisheries and Aquaculture Science

6 – 17 Dec 2021

Plankton image classification



https://doi.org/10.5281/zenodo.6799166

Contents

1 Executive summary

1.1 Background and Data Overview

Cefas (The Centre for Environment, Fisheries, and Aquaculture Science) is an agency of Defra (the Government's Department of Environment, Food and Rural Affairs) and world leading experts in marine and freshwater science. Research at Cefas aims to tackle the serious global problems of climate change, marine litter, overfishing, and pollution to secure a sustainable future for marine ecosystems.

The Cefas Endeavour, a multi-disciplinary research vessel, collects millions of plankton images during its surveys through the Plankton Imager (PI) system: a high-speed imaging instrument which continuously pumps water, takes images of the passing particles, and attempts to identifies the zooplankton organisms present (Figure 1). Images have varying shapes and sizes with a highly-skewed distribution towards smaller particles/images. Of these, over 80 percent can be classified as detritus (e.g., sand, seaweed fragments, microplastics) which are traditionally removed by-eye before any analysis, leaving the remaining plankton images to be manually labelled.

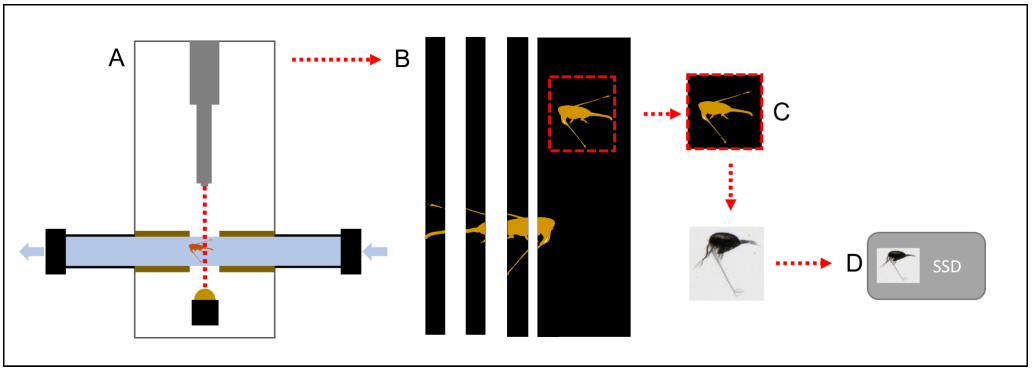


Figure 1: RV Cefas Endeavour PI pipeline. Water flows through perpendicular to a line-scan camera (A). Scan lines are stitched (B) and regions of interest are extracted (C). Images are converted from RGB and stored on external storage (D) (Source: James Scott (CEFAS))

The challenge dataset consisted of 58,791 TIF (Tag Image File Format) images of individual objects detected and segmented in imagery collected on the RV Cefas Endeavour research vessel using the PI system.

Approximately 17,000 of these images are of individual zooplankton. The plankton images had previously been manually classified by experts into two main categories: Copepods, small or microscopic aquatic crustacean of the large taxonomic class Copepoda (see Figures 25 and 26), and Non-Copepods (see Figures 23, 24, 27, 28), for all other plankton not belonging to the Copepoda class. The experts also categorised these images further into 38 species classes. This expert manual classification allowed challenge participants to verify the accuracy of the automated classification methods explored.

The number of images varied greatly between the 38 classes, ranging from 4000 images to 10 images per class. Challenge participants therefore had to decide how to address this imbalance in order to produce a model that could be useful and accurate classifications of plankton.

The remaining 40,000 images consisted of individual pieces of detritus (see Figures 29 and 30). These images were of other objects collected by the RV Cefas Endeavour PI system such as sand, seaweed, or microplastics. Manual removal of these images has been shown to be a significant bottleneck in the analysis of imagery collected using the PI. Therefore as an additional challenge, participants had the opportunity to explore automated sorting of images into plankton and detritus in order to facilitate application of plankton classification models to imagery collected from the PI in real time without pre-processing to remove these erroneous objects.

1.2 Main objectives

1. The primary goal of the challenge was to build a classifier that can distinguish between copepod and non-copepod plankton in images that do not include detritus.
2. Build a classifier that can distinguish between the 38 labelled plankton classes in images that do not include detritus.
3. Stretch Goal: To build a classifier capable of distinguishing between detritus and plankton in addition to accurate classification of plankton in copepods versus non-copepods, and between the 38 classes

1.3 Approach

We approached the task with three key themes:

1. Exploratory data analysis including preliminary investigation of data quality, spatial clustering of data, and principle component analysis (PCA) of low-level features;
2. Use of Morphocut to extract low-level features to train a Random Forest (RF) model which provided a baseline accuracy score;
3. Use of pretrained Convolutional Neural Networks (CNNs) with a ResNet architecture to improve the accuracy of plankton classification at finer taxonomic levels compared to the baseline RF, and Generative Adversarial Networks (GANs) to increase the number of images with synthetic data based on the existing dataset.

1.4 Main conclusions

The plankton dataset collected by CEFAS endeavour is a valuable resource which provides information on a wide range of variables that could be used to improve image classification of plankton. As part of the DSG, we were able to successfully and efficiently explore some of these variables through unsupervised and supervised data-driven methods. For instance, the clustering analysis of geo-location metadata and image-related metrics using PCA gave valuable insights into what phenotypic features of plankton may be most helpful to consider for improving classification over larger geographical areas.

In order to address class imbalance between different plankton species and detritus in the challenge datasets, data augmentation using GANs was also explored. This could allow for a significantly increased in dataset size, in particular for minority classes, and reduce bias in model performance due the relative rarity of certain plankton species.

For the primary goal of classification, superior accuracy results (94-99%) were obtained across all label levels when comparing the proposed ResNet architecture with baseline models (74-92% accuracy). Taking advantage of transfer learning, we optimised a pre-trained ResNet model derived from

ImageNet, an extensive dataset of non-planktonic images, to provide accurate results with reduced training effort for the plankton dataset.

scivision, a Turing-developed tool, aimed to accelerate the experimentation of the challenge participants. While the usage of the tool was minimal for model training, it allowed a quick dive into the dataset without an extensive data preparation. In addition, some preliminary results showed the feasibility of using scivision to visualise the best performing ResNet and RF models.

1.5 Limitations

The first limitation is time available for the project. There were only two weeks for the tuning of the models, a task that could potentially last several months. A second limitation was that there were cases of mislabelling within the detritus class.

MorphoCut only provides a rough thresholding of significant features. The RF method, previously used to analyse the data, was more computationally intensive than the CNNs. The processing of the images for the CNN involved resizing the images to the same size irrespective of their original scale which removed one of the features relevant in plankton classification: their size.

1.6 Recommendations and future work

It is recommended that a more detailed evaluation of the models be carried out on a larger, curated dataset. Data augmentation increased the accuracy when using the CNN, so a test with the RF model is advised. As for feature extraction, it is recommended that other approaches beyond MorphoCut be investigated such as algorithms measuring the colour and hue of an image). It would also be beneficial to increase the dataset by pooling data from several research organisations, collected in other regions. It would then be valuable to apply explainable AI methods to this expanded dataset to further understand the morphological differences in plankton. It is also recommended further exploration of the detritus images be carried out with methods such as clustering. The ResNet pipeline developed could also potentially be applied to other fields, such as micro plastic research, as

the detritus imagery contains similarities to imagery of marine micro plastic particles.

2 Introduction

2.1 Rationale

Plankton play an essential role in the marine ecosystem and the global carbon cycle and carbon sequestration, regulating the exchange of carbon dioxide between the atmosphere, surface ocean and ultimately the seabed. In particular, the role of zooplankton in the food web is critical as they occupy a central position, often controlling the abundance of smaller organisms by grazing and providing food for many larval and adult fish and seabirds. Zooplankton is also used in global monitoring efforts, providing reliable and sensitive indicators to climate change and ecosystem health.

Zooplankton are defined as animals that cannot propel themselves against the current [45]. This group includes a wide range of organisms from small unicellular organisms ('microzooplankton') to giant jellyfish ('macrozooplankton'). Traditional zooplankton research focusses on mesozooplankton, which are animals in the size range of 200 - 20,000 μm, including copepods (oar-footed crustaceans), euphausiids (e.g. krill), medusae (jellyfish), chaetognaths (arrow worms), amphipods, marine gastropds (sea snails and sea slugs), polychaetes (segmented worms) and ostracods (seed shrimps). This diverse group fulfils a wide range of ecological functions and hence play a critical role in the marine ecosystems.

Traditionally, mesozooplankton have been collected with plankton nets or net-based sampling systems, such as the Continuous Plankton Recorder, and identified using light microscopy. These sampling techniques are, however, relatively limited in their potential spatial and temporal extent as they typically require ships for sample collection and extensive labour for sample analysis (especially the enumeration and taxonomic classification). In recent years, the advance of imaging and computing technologies make it now possible to image zooplankton in-situ over larger spatio-temporal scales than with traditional sampling methods.

While technology has progressed rapidly, there is a bottleneck in large amounts of imaging data collected. The taxonomic identification of each object is still largely carried out by humans, taking just as much time as traditional microscopy. To effectively sample, monitor and study mesozooplankton, we require efficient data processing flows that generate reliable taxonomy.

A range of in-situ and ex-situ plankton imaging systems are now available [25]. We are here focusing on the PI developed specifically for continuous horizontal sampling of mesozooplankton (Figure 2).

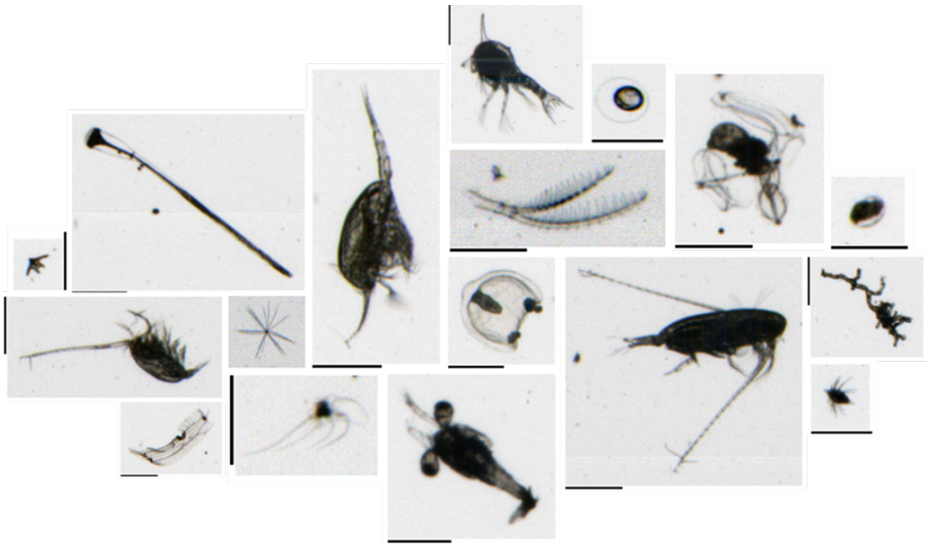


Figure 2: Mesozooplankton imaged by the PI (Source: Pitois)

Our aim here is to develop on-the-fly machine learning methods for automatically classifying plankton classes, that improves the usability of in-situ imaging devices for real-time awareness of the planktonic ecosystem.

2.2 Object Detection and Feature Extraction

2.2.1 Object detection

Object detection in plankton images is typically straightforward as, owing to the low particle density and simple background, objects are typically imaged against a white or, depending on the imaging system, black background. For

the PI, the images are colour images on a white background. For this project, object detection and segmentation had already been carried out.

2.2.2 Low-level features

One of the first plankton identification programmes, Plankton Identifier [1], was based on the open-source software ImageJ and used the simple low-level features that the standard algorithm of this software provided. A purpose-build plug-in for this programme, ZooProcess, is still commonly used to process zooplankton images collected ex situ (e.g. using FlowCam and ZooScan) and in situ (e.g. using the Underwater Vision Profiler). These features are also those used for RF classification prediction in the community plankton sorting software EcoTaxa [2], which hosts, at the time of writing, over 175 million plankton images.

More sophisticated low-level features have been used for plankton classification predictions, for example on images from the PI [4]. Another example is the study by Zheng et al. (2017) [54], who used a comprehensive list of low-level features to describe plankton, including geometric and grayscale features, texture features (Gabor filter, variogram function, local binary pattern, binary gradient contours), granulometric features and local features (histograms of oriented gradients, scale-invariant feature transform, inner-distance shape context).

2.2.3 High-level features

While low-level features are often meaningful to humans (e.g. grey value, image size), high-level features are calculated by neural networks. These features are learnt by the neural network in the hidden layers, and can be thought of as higher level descriptors such as complex shapes and textures or parts of objects. High-level features make CNNs a powerful method for object classification.

[1] https://www.obs-vlfr.fr/~gaspari/Plankton˙Identifier/index.php
[2] https://ecotaxa.obs-vlfr.fr

2.3 Object classification

Convolutional Neural Networks

CNNs are architectures that are suitable for image data and are modelled on biological neurons [7], where the architecture involves processing of units with identical weight vectors and arrangement of local receptive fields in a spatial array. Their hierarchical architecture encompasses alternating subsampling layers, which are analogous to simple and complex cells in the primary visual cortex. CNNs perform mappings between spatially / temporally distributed arrays in arbitrary dimensions and are generally characterised by the following constraints [29]:

- Translation invariance: spatial translation has no effect on the neural weights
- Local connectivity: neural nodes which are located in spatially local regions are connected
- Progressive decrease in spatial resolution: when there is a gradual increase in the number of features

See Appendix A for a timeline of CNN architecture development.

Bello [1] revisited the ResNet architecture [15] and concluded that the strategies used to train and scale the data appear to matter more than architectural changes, and that a well-trained ResNet matches state-of-the-art models.

2.3.1 Architectures used for plankton images

For the classification of plankton, a range of CNN architectures have previously been explored (Table 1).

Reference	Model	Usage
Orenstein et al. 2015 [27]	CNN trained exclusively on plankton data, ImageNet	Starting point for development of plankton-specific classifiers.
Dai et al. 2016 [5]	ZooplanktoNet (inspired by AlexNet and VGGNet)	Deep convolutional network for zooplankton classification
Lee et al. 2016 [21]	CIFAR10	Transfer learning to overcome class imbalances
Li et al. 2016 [22]	ResNet (19, 32 and 50 layers) and VGG-19	Introduce a deep residual network to classify images of plankton
Py et al. 2016 [33]	Custom (inspired by Network in Network and GooLeNet)	Design very deep CNN and developed a inception module for multi-scale input
Pedraza et al. 2017 [30]	AlexNet	Classification of diatoms
Pedraza et al. 2018 [31]	RCNN, YOLO	Classification of diatoms
Cheng et al. 2019 [3]	AlexNet, VGG16, VGG19, GoogLeNet, and ResNet50	Multiple Deep Learning models to collaborate in a single classification system improves classification accuracy for minority classes compared with the best individual model. ResNet50 performed best.
Ellen et al. 2019 [8]	VGG-16	Improving plankton image classification using context metadata
Lumini et al. 2019 [26]	AlexNet, GoogleNet, InceptionV3, VGGNet, ResNet50, ResNet101, DenseNet, MobileNetV2, NasNet	Evaluation of various CNN architectures for plankton identification. Concludes that the best stand-alone model for most targets is DenseNet, though an ensemble improves the performance of the best single model.
Kerr et al. 2020 [18]	VGGNet, GoogLeNet, ResNet, DenseNet	Comparison of different architectures for plankton classification. Best performing CNN was ResNet50. Concluded that combination of CNN and Support Vector Machine improves classification.
Schroder at al. 2020 [42]	ResNet18	Clustering of plankton images based on features for efficient annotation
Li at al. 2021 [23]	CycleGAN (DenseNet YOLO3)	A novel detection and classification strategy for imbalanced distributed plankton

Table 1: CNN Architectures Previously Used to Classify Plankton Images

2.4 Challenges for plankton classification

Plankton classification using deep neural networks suffers from two challenges [23]. First, there is typically a large class imbalance in plankton

datasets as organisms are not distributed equally and have widely varying abundances. This imbalance challenges CNNs owing to overfitting. Second, many subtle features of plankton in images may be lost during the CNN operation.

Other challenges include [5]:

1. Image resolution is often limiting for smaller zooplankton (small number of pixels), obscuring features critical for identification. Classification of these organisms by humans, even by highly trained taxonomists, is easily mistaken, resulting is uncertain training data. As zooplankton abundance follows the typical particle size spectra (log scale), these smaller zooplankton are much more frequent than larger, clearly identifiable organisms.
2. The size of datasets, particularly those with high-quality training data, is often small (compared to other fields).
3. The extracted features may not be discriminative enough for zooplankton, due to both the overlap in common features between many plankton species, and the high level of variation in phenotype within many species.

2.5 Classification

Experts can reliably classify species into categories, e.g. detritus vs zooplankton. However, when they deal with large amounts of data, classification becomes a timely and expensive task, where an automated solution is critical. As a data mining task, classification aims to accurately predict the target class for each case in the data. We can identify two classification tasks relevant to the analysis of plankton imagery (Figure 3):

1. Multi-class classification (e.g. detritus vs zooplankton, where we have to predict if in an image we have detritus or zooplankton)
2. Multi-label classification (e.g. an image can contain multiple species, as well as detritus; this can be done as a further step of this project, as currently, we have just images that contain one class)

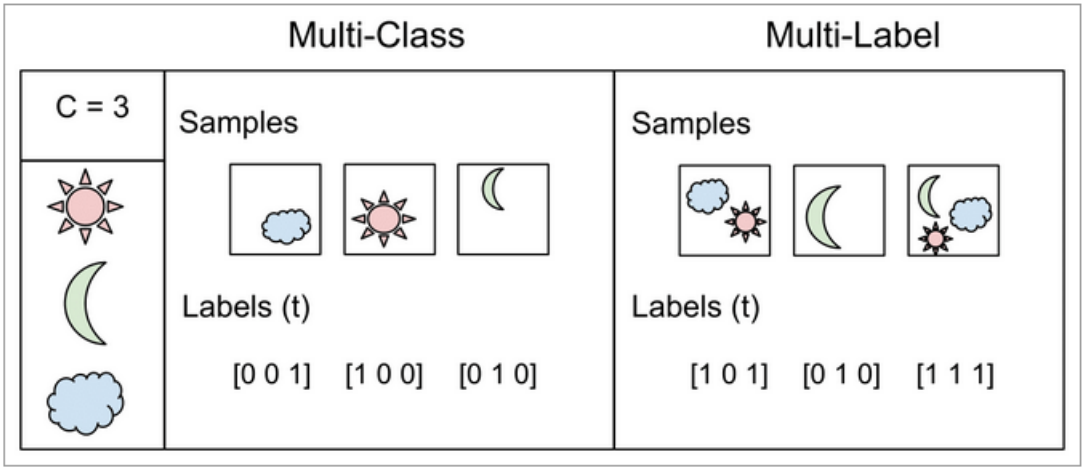


Figure 3: An infographic description of multi-class and multi-label classification [38]

2.6 Challenge summary and objectives

The PI is a high speed imaging and analysis instrument that photographs plankton in the size range 100 micron to 20 mm in water. Each specimen imaged is timestamped, GPS located and recorded to disk for further analysis.

The images contain examples of various plankton species. We also include examples of detritus (sand, seaweed, particulates etc.)

The challenge is to:

1. Build a classifier that can distinguish between copepods and non-copepods in images that do not include detritus.
2. Build a classifier that can distinguish amongst labelled plankton at a finer taxonomic scale in images that do not include detritus.
3. Stretch Goal: Repeat the experiment including detritus images.

Based on a better understanding of Plankton data, we selected a set of machine learning models that can help us find solutions to solve this real-world problem.

Our main task was to classify our data and there are many ML models that could help us to solve these tasks, but given the time constraint we selected

the following: unsupervised models (clustering), supervised models (RF) and deep learning models (CNNs).

3 Data Overview

3.1 Data collection

Plankton images were collected over the UK continental shelf between August 2016 and October 2020 using the PI.

3.1.1 Plankton Imager

The PI is an in situ camera system designed to replicate continuous horizontal sampling of mesozooplankton similar to the Continuous Plankton Recorder, allowing underway sampling of mesozooplankton over wide spatial scales. As the sample passes through the PI, the plankton is imaged using a line scanning camera (Basler 2048-70kc) at a scanning rate of 70,000 lines per second. Each line is 10 µm wide and 20.48 mm long. The images are then assembled, producing colour images of mesozooplankton against a white background. Resulting objects range in size from from 10 µm to 2 cm [4].

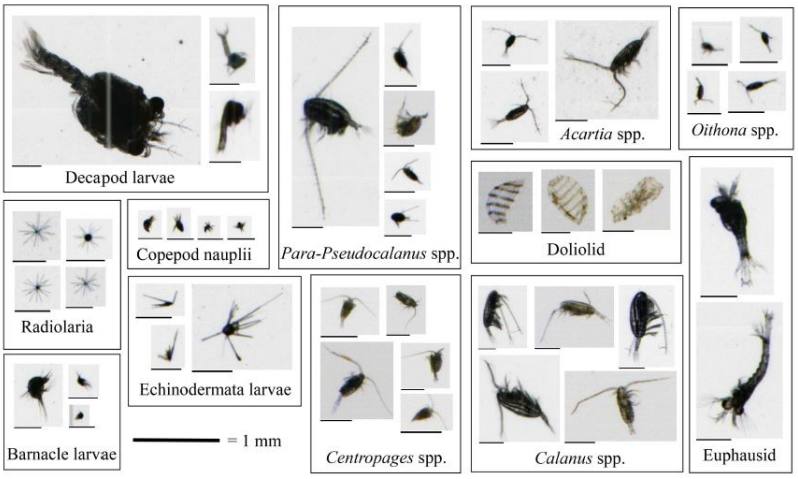


Figure 4: Collage of example mesozooplankton images for 12 of the most abundant categories (Scott et al. 2021 [43])

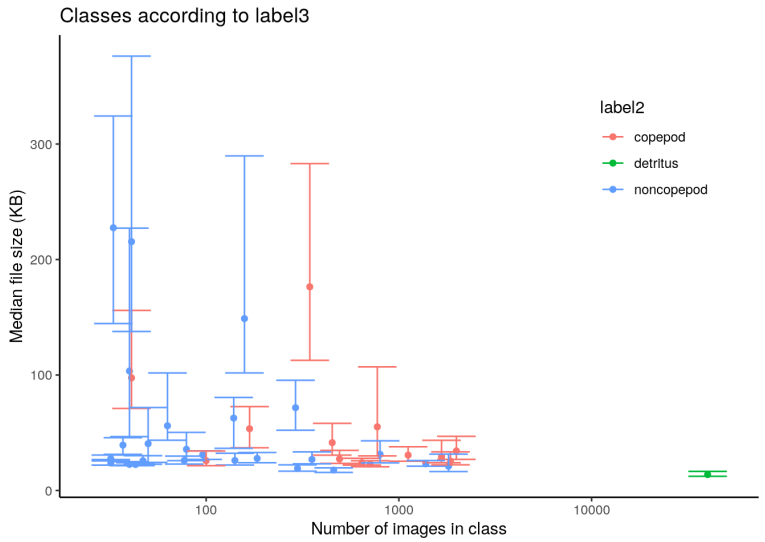


Figure 5: Graphical representation of class imbalances. X-axis shows number if images per label3 class; y-axis shows the median file size in each label3 class, which serves as a proxy of object size. Error bars show upper and lower quartile (i.e. range of 25% to 75% of the data). Colours identify label2 classes (red: copepods, blue: noncopepods, green: detritus). Note log-scale on x-axis.

3.1.2 Data set

A total of 58,791 plankton objects were detected, saved as individual image files (`.tif`), and classified by trained human taxonomists. Plankton were identified, where possible, to species level. Labels were grouped into three levels: (1) zooplankton and detritus, (2) copepod, non-copepod and detritus, (3) species-resolved (38 plankton classes + 1 detritus class).

Label 1

detritus	40000
zooplankton	17069

Label 2

detritus	40000
copepod	10346
noncopepod	6723

Label 3

detritus	40000
copepod_calanoida_para-pseudocalanus-spp	1988
copepod_unknown	1853
radiolaria	1810
copepod_calanoida	1665

copepod_nauplii	1380
copepod_cyclopoida_corycaeus-spp	1117
echniodermata-larvae	799
copepod_calanoida_centropages-spp	773
copepod_cyclopoida_oncaea-spp	710
copepod_harpacticoida	643
copepod_cyclopoida_oithona-spp	492
nt-phyto_ceratium-spp	459
copepod_calanoida_acartia-spp	451
nt-bubbles	354
copepod_calanoida_calanus-spp	345
nt_phyto_chains	298
tunicata_doliolida	291
nt-phyto_rhizosolenia-spp	184
copepod_calanoida_temora-spp	168
chaetognatha	158
annelida_polychaeta	141
euphausiid_nauplii	139
copepod_cyclopoida	100
byrozoa-larvae	96
appendicularia	79
cirripedia_barnacle-nauplii	77
cnidaria	63
fish-eggs	50
ostracoda	47
bivalvia-larvae	43
euphausiid	41
copepod_calanoida_candacia-spp	41
gastropoda-larva	40
decapoda-larvae_brachyura	40
cladocera	37
mysideacea	33
tintinnida	32
cladocera_evadne-spp	32

3.2 Data quality issues

Of the 581791 images, 1722 are augmentations: 574 of the original images were transformed (simple mirroring along x, y and xy axes). These images can be identified by their file extensions ("`˙fx.tif`", "`˙fy.tif`", and "`˙fxy.tif`"). These images were removed and not used further for this project. We also removed images with duplicate labels i.e. with similar filename.

The new data set is given in `plankton-dsg-challenge/data/processed/clean-index/` within the GitHub repository. We then split the data into test and training data sets using stratified random sampling to produce `test.csv` and `train.csv` in `plankton-dsg-challenge/data/processed/test-train/`. The partition sizes were train (42317), validation (10580) and test (5863). Later, during the exploratory data analysis, we discovered that a number of the images labelled as detritus were in fact empty. In the interest of time we chose to ignore this problem.

3.3 Data Augmentation

The amount of training data available often has a major impact on the performance of deep learning neural networks.

Data augmentation is a technique for artificially generating additional training data from existing training data. This is accomplished by using domain-specific approaches to transform examples from the training data into new and unique training examples. The most well-known kind of data augmentation is image data augmentation, which involves transforming images in the training dataset using operations such as shifts, flips, zooms, pixel intensity and brightness changes. The goal is to add similar but distinct examples to the training collection that can improve the model's learning of a class representation through variation of image characteristics like lighting, perspective, noise, and rotation. By including these variations in the training data, this could allow the model to perform better on real-world footage that may be subject to similar variation.

Modern deep learning algorithms, such as CNNs, can learn features independent of where they appear in the image. However, augmentation can help with this transform invariant method to learning by assisting the model in learning features that are also transform invariant, such as left-to-right to top-to-bottom ordering, light levels in images, etc. Typically, image data augmentation is only used on the training dataset, not the validation or test datasets. A similar method of data preparation, such as image resizing or pixel scaling, should be applied consistently across all training, validation, and testing datasets.

3.3.1 Basic techniques

Deep learning frameworks such as Tensorflow, Pytorch, and Keras offer tools within their libraries to apply a range of data transforms for augmentation when training a model. For example, Keras offers the ImageDataGenerator class. First, the class may be instantiated, and arguments to the class constructor specify the configuration for the types of data augmentation.

- Image shifts via the width_shift_range and height_shift_range arguments.
- Image flips via the horizontal_flip and vertical_flip arguments.
- Image rotations via the rotation_range argument.
- Image brightness via the brightness_range argument.
- Image zoom via the zoom_range argument.

A range of image transform techniques are supported, as well as pixel scaling methods (Figure 10; Figure 11). We will focus on five main types of data augmentation techniques for image data; specifically: transforms such as rotation, width/height shift, shearing, zooming, and flipping. Our techniques are parameters inspired by prior work on plankton classification for a Kaggle challenge as well as TensorFlow's ImageDataGenerator.

We focused on augmenting classes that had the poorest performance, which roughly matched the classes that had the fewest image samples. To account for this class imbalance, a scaling factor was applied so that smaller classes had more image augmentations created and larger classes had fewer.

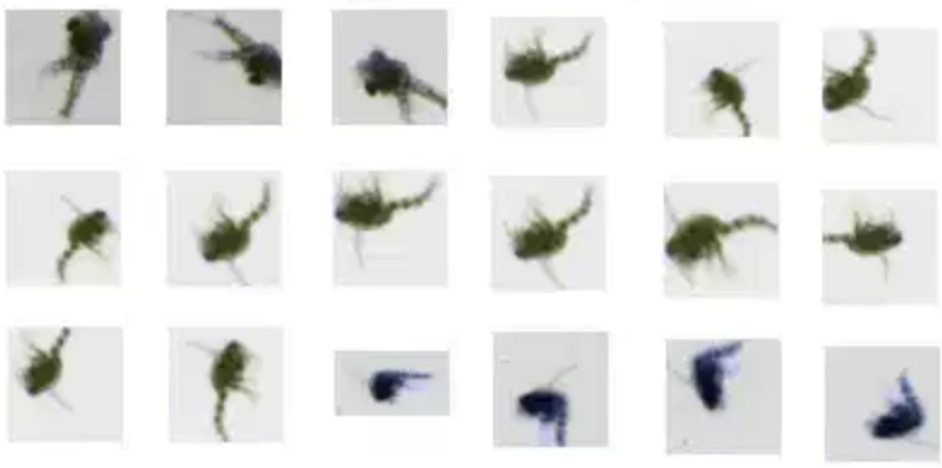


Figure 6: Transformations performed on ecapoda larvae brachyura

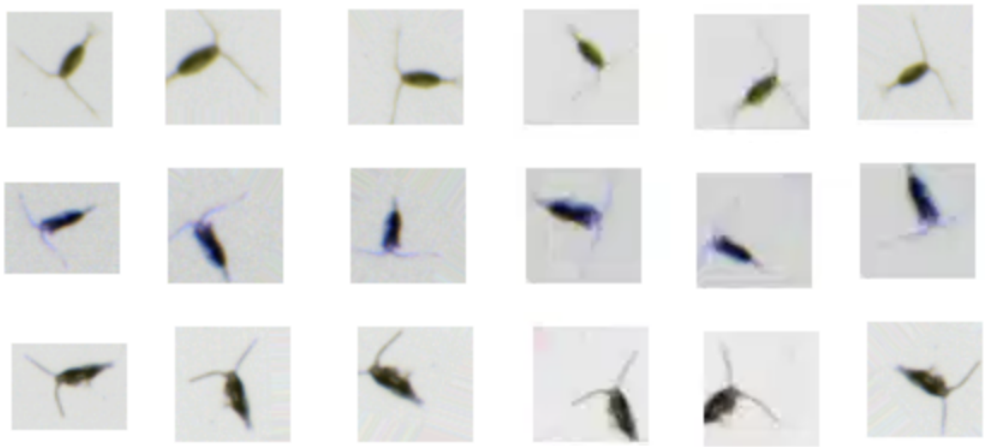


Figure 7: Transformations performed on copepod calanoida acartia spp

3.3.2 Synthetic Data Generation

Another technique we explored is synthetic data generation through Generative Adversarial Networks (GANs). Introduced by Ian Goodfellow in 2014 [13], GANs use two neural networks—a generator and a discriminator—which learn through a minimax game how to generate synthetic data and distinguish between real and synthetic data, respectively[13]. GAN's central concept is derived from the Nash equilibrium in game theory. It presupposes two players in the game: a generator and a

discriminator. The generator's goal is to learn the composition of accurate data, whereas the discriminator's goal is to identify whether the input data is actual or synthetic correctly. To succeed, the two players must constantly increase their abilities. This process aims to arrive at a Nash equilibrium between the two parties.

Deep Convolutional Generative Adversarial Network (DCGAN)

For synthetic data generation in our dataset, we used a Keras implementation of DCGANs, a class of CNNs used for unsupervised learning of image representations.

DCGAN uses convolutional and convolutional-transpose layers in the generator and discriminator, respectively. It was proposed by Radford et al. in the paper Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks. Here the discriminator consists of stridden convolution layers, batch normalization layers, and LeakyRelu as activation function. It takes a 3x64x64 input image. The generator consists of convolutional-transpose layers, batch normalization layers, and ReLU activations. The output will be a 3x64x64 RGB image [34].

Architecture for Deep Convolutional GANs

The architecture applied for DCGANs augmentation is as follows:

- replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batch norm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in the generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Figure 8: Copepods generated using DCGANs

Figure 9: Copepod Calanoida Acartia spp generated using DCGAN

3.4 Exploratory data analysis

3.4.1 Image classes

Examples of images in each class The images show a high diversity even within each class (see Figures 23 to 28). The copepod classes look very similar, particularly towards the smaller size of images.

For some classes, colour appears to be a good indicator. For example, Euphausiids appeared to generally have a more yellow tint (Figure 27). However, some images are dominated by blue shades, suggesting that there were quality issues with the colour channels for some deployments. If colour should appear important for classification, quality control of the colours after each deployment might help to improve classification.

3.4.2 Density distribution of classes

Spatial distribution of classes using basic visual analysis of point patterns

Our initial visual analysis of the spatial patterns of our data primarily illustrates the sampling locations. When plotting the location of all objects according to label 1 (plankton vs debris), plankton appears to be spread over a much wider area, across the Celtic Sea, North Atlantic and English Channel (Figure 10-top). Debris appears to be much less widely spread

across the study area, primarily being focused in the English channel to the east of the study site. For label 2 (three classes), both copepods and non-copepods appeared fairly evenly distributed across the study site, with small clusters of non-copepods in the area just west of Guernsey (Figure 10-middle). Copepods appeared to be less clustered across the study site, with a larger proportion being present in the sea between Cornwall and Ireland. For the third label (39 classes), no clear spatial patterns was apparent (Figure 10-bottom).

This visualisation method is not ideal as there is substantial of data points at each location, so that the true distributions are not obvious. We therefore used kernel density estimations to investigate the true distribution.

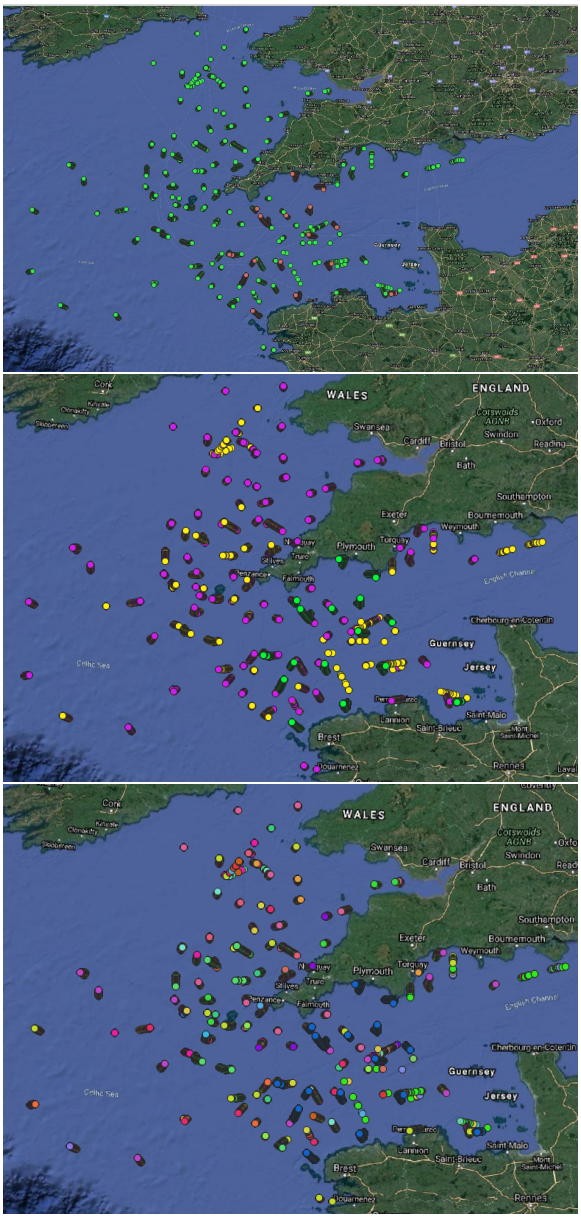


Figure 10: (Top) Spatial distribution of copepods (pink), non-copepods (yellow), and detritus (green). **(Middle)** Spatial distribution of plankton (green) and detritus (brown). **(Bottom)** Spatial distribution of the dataset at the third scale of classification.

Spatial distribution of classes based on kernel density estimation Spatial density plots (using kernel density estimation) were created to enable the distinction between plankton and detritus. This method was chosen to overcome the limitations of the point plots. These plot does not show true abundances and partly reflects sampling effort. Nonetheless, if object

classes were equally distributed, we expect to see the same distribution patterns.

The distribution of plankton was heterogeneous across the study area (Figure 11-top). Highest densities appear to occur on the south coast of Cornwall and in the central Celtic Sea and the Irish Sea. Generally, plankton was observed across the entire study region.

For detritus, highest densities appeared to occur on the North West French coast, near Plousecat (Figure 11-bottom). Different to the plankton classes, detritus did not occur at every sampling station. We suspect that this unusual distribution is due to inconsistencies with the treatment of 'detritus' images. It appears that for many deployments, detritus categories were deleted all together, whereas for some deployment, any images that was not clearly plankton was labelled 'detritus'.

This analysis suggests distinct spatial patterns of the different classes, where particles of similar type appear to be geographically linked. This interpretation makes sense from a biological perspective as plankton are typically patchy [6]. Though we did not have time to explore this avenue, we hypothesise that the incorporation of geolocation information of objects might improve classification accuracy [51].

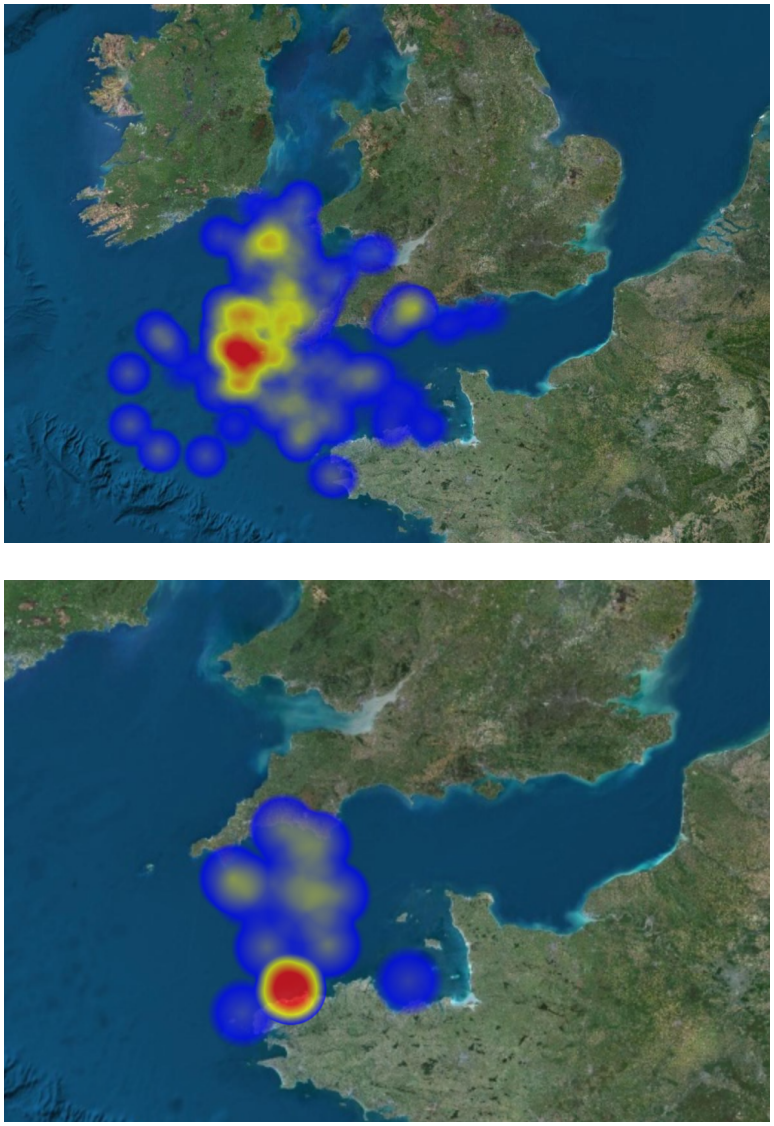


Figure 11: Distribution of plankton (top) and detritus (bottom) visualised using density plots. Colours show low to high densities as a colour gradient: blue $>$ yellow $>$ red.

3.5 Clustering

Clustering or clustering analysis is a machine learning technique, which groups unlabelled datasets so that we can have a better understanding of similarity between groups can be seen as a form of classification.

Most of the images collected by the PI are of detritus. It is difficult to categorise detritus manually, therefore in this dataset no detritus classification was done. We explore whether we can use unsupervised

clustering to find groups of detritus that are similar, and could be used for environmental mapping. We explore this technique on low-level features using MorphoCut pipeline. MorphoCut [41] is an image processing Python library designed to handle large volumes of oceanographic data. MorphoCut was developed with plankton images in mind, particularly the processing of plankton images for uploading to EcoTaxa (see section 2.2.2)which allows fast extraction of low-level features. Therefore, MorphoCut is particularly well-suited for low-level feature extraction of our image database.

For the low-level features, most parameters were not normally distributed. As distributions varied, we explored possible transformation techniques and opted for the square-root transformation. It seemed to improve the distribution of all parameters (visual inspection) and can also be used when the data include zeros.

All data were scaled using *scale()* function in R. For the PCA calculation, however, these values have different weights. We therefore scale each parameter so that the mean is 0 and the SD is 1. This way, all variables - regardless of the *pca()* function from R's FactoMineR package.

The optimal number of clusters was determined visually using the elbow method, plotted using the *fviz_nbclust()* function from R's factoextra package. For low-level features, the optimal cluster number was chosen as 7.

The data was clustered using k-means clustering (using R's *kmeans()* function). To visually explore each cluster, representative images for each cluster were picked at random and compiled as a collage.

3.5.1 Clustering for detritus

The PCA of the low-level features shows that a large number of variables contribute to the first principal component (Figure 12, Dimension 1). Closer inspection of these variables shows that the majority of these are related to size, specifically overall image size and length (e.g. perimeter, area, diameter, major axis, height) (Figure 13). The second dimension, or the second principal component, on the other hand, is influenced strongly by parameters relating to object width and grey value. There also seems to be some information in where the object is located within each image frame.

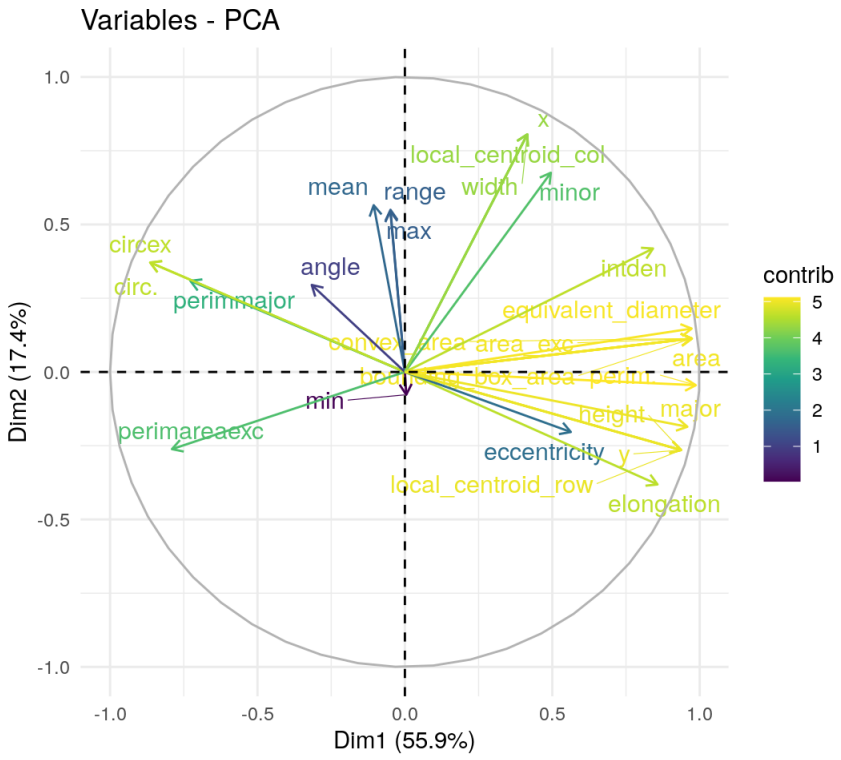


Figure 12: Correlation circle showing the relationship between all variables and how they influence the components. Level of contribution is indicated in colour as shown in the colour legend.

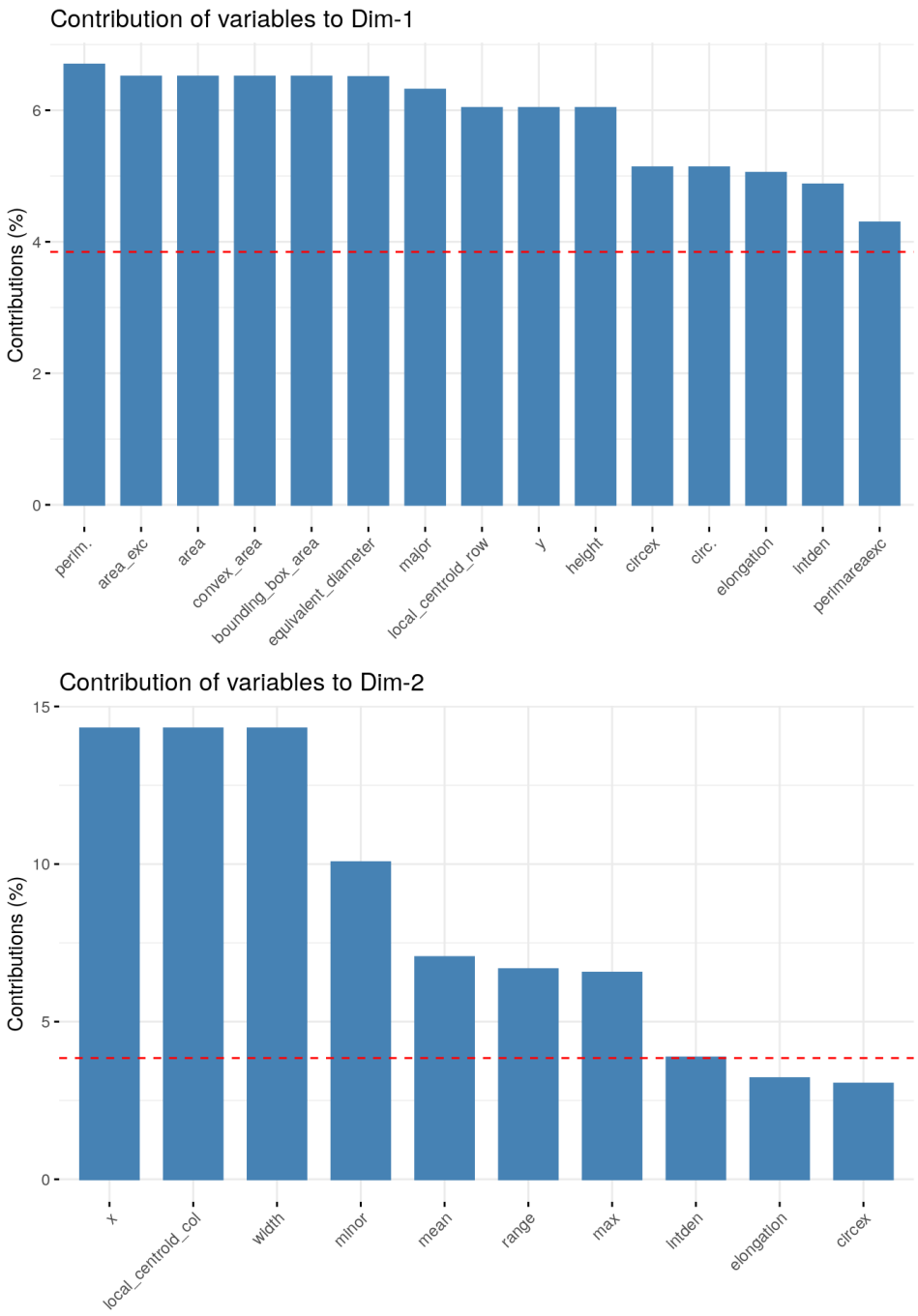


Figure 13: Contribution of variables to principal components. Variables significantly contribute to the component if their contribution is above the threshold (red dashed line). Top: First dimension. Bottom: Second dimension.

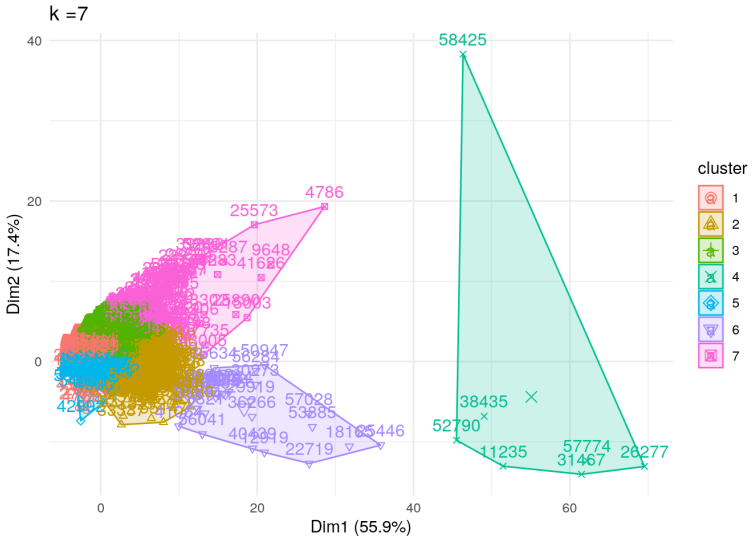


Figure 14: Cluster analysis of detritus features. Low-level features for all detritus images were evaluated using PCA. The first axis approximates size metrics; the second axis approximates object width and grey level. Points closer together have higher similarity. Images were grouped using kmeans clustering with 7 clusters. Colours show cluster identity as indicated in the legend. Numbers identify objects.

The k-means clustering with 7 clusters identified three clusters (cluster 4, 6 and 7) that were dominated by large objects. Clusters 6 and 7 appeared to be separated based on the second dimension, indicating differences in image width and grey values. When we explored sample images from these clusters, this interpretation was confirmed. Cluster 4 contained mostly what appeared to be empty strips of the sampling volume, with the exception of one large organic object. The latter could be considered an outlier based on the PCA plot (object 58425 in Figure 14). Cluster 6 was comprised of long and thin images, showing a mix of empty images and organic material (likely faecal pellets and fibre-like structures). Cluster 7 was comprised of larger, wider objects. The objects appear to be mostly organic.

The majority of detritus images were assigned to clusters 1, 2, 3 and 5. The largest cluster, cluster 1, contained primarily small circular objects of insufficient resolution to allow positive identification. Cluster 2 contained slightly larger images and darker images. Again, many of the images appear to be empty. Cluster 3 appears similar to cluster 1 with primarily bright images, albeit containing larger and more elongated objects. Cluster 5 was

dominated by dark images. Again many of these images appeared to be empty.

Overall, the clustering of detritus appeared to be a powerful method to retrieve more information on the ecosystem. However, the detritus class currently appears to be used as a bucket for any unwanted images, with a lot of the images not containing any object, which hinders any meaningful use of this class for ecosystem description.

4 Methods

4.1 Spatial exploration

The goals of visualising and performing geospatial analysis on the data as part of the initial data exploration are as follows:

- to look at the spatial clustering of the classified data at different scales of classes
- to compare the distributions of the classified images to the distributions of the images we classify
- to identify whether the longitude and latitude data is useful in the project
- to provide visual context of the spatial considerations of the data i.e. clustering of certain classes in a given region.

4.1.1 Extraction of metadata and geolocation

The metadata, including longitude and latitude data, for the images were extracted using the ExifTool, a platform-independent Perl library and command-line application used for reading, writing, and editing metadata. For more information, see here.[3]

[3] See also `https://github.com/alan-turing-institute/plankton-dsg-challenge/blob/main/notebooks/python/scivision/3_explore_metadata.ipynb` for a way to do this with Scivision.

4.1.2 Geospatial mapping of data

The geolocation of the data was explored using QGIS, an OpenSource software[4]. The data in csv format was imported into QGIS as a delimited text file using 'GPSLongitude' as the 'X field' and 'GPSLatitude' as the 'Y field'. The Geometry Coordinate Reference System (CRS) was set as EPSG:4326-WGS84 and the attribute was renamed to 'planktonpoints'. The csv containing the class types was also imported as csv and the table was renamed 'classifications'. In the properties of the 'planktonpoints' attribute, the 'classifications' file was joined to the 'planktonpoints' attributes, using 'FileName' as the common join feature. Once joined, the symbology of the 'planktonpoints' attribute was changed to display different elements of the dataset.

4.2 Low-level feature extraction with MorphoCut

Within the MorphoCut pipeline, images were converted to greyscale and objects were detected using a threshold of 128 (default). In future, we recommend using a more sophisticated thresholding algorithm as thresholding strongly affects particle metrics (e.g. [10]). A total of 33 features such as surface area of the object, the average grey value within the object, among others (Table 8), were extracted for each image. There were used for further data exploration (see Subsection 3.5) and feature matrix for the baseline RF model.

4.3 Baseline model: RF on low-level features

RF Classification is a simple classification algorithm that builds multiple decision trees and merges them together to get a more accurate and stable prediction.

RFs or random decision forests are an ensemble learning method, taking the collective decision of many trees thus improving the performance of a single random tree. They are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance [5].

[4] https://download.qgis.org/

[5] Variance is a statistical measurement of the spread between numbers in a data set and measures how far each number in the set is from the mean and thus from every other number in the set.

A limitation of RFs is the computational complexity ($O(vn\log(n))$) where n is the number of records and v is the number of variables/attributes), as RFs is slow in generating predictions because it has multiple decision trees. More specifically, when it makes a prediction, all the trees in the forest have to make a prediction for the same given input and then perform voting on it (Figure 15).

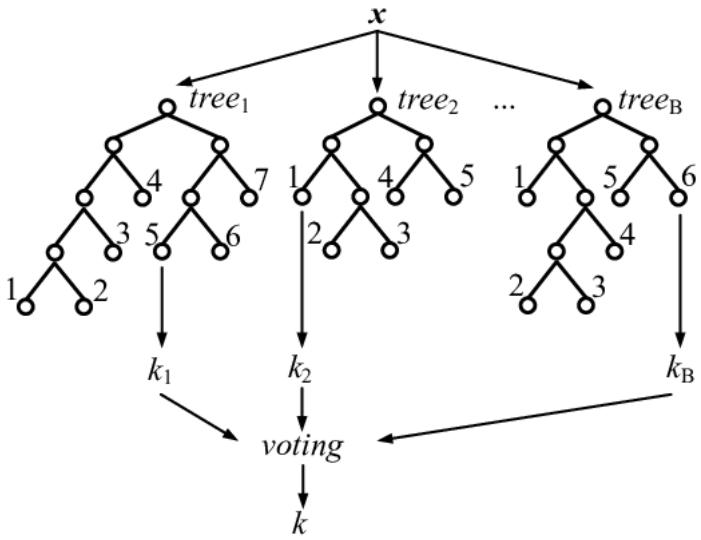


Figure 15: A general RF architecture for classification tasks [9]

Implementation: For implementing a RF classifier we used Scikit-learn library [32], integrating in Python the sklearn.ensemble module that includes the RF Classifier that should be fitted with two arrays: a sparse or dense array *X* of shape (n_samples, n_features) holding the training samples, and an array *y* of shape (n_samples,) holding the target values (class labels) for the training samples. By default, the number of trees in the forest is set to 100 (n_estimators=100).

Evaluation Methods for Random Forest During the decision-making process of RFs, multiple features are taken into consideration and each feature can influence the prediction. In order to achieve a better performance, RFs models can be evaluated using splitting measures, such as entropy and Gini index. We focused on these two measures as we will use them for the optimisation task (Figure 18).

Entropy is a measurement of the randomness in data points and varies between 0 and 1, where 1 indicates more randomness or uncertainty. The value depends on the number of groups or classes present in the data set.

Gini Index is a measurement of the purity of classification and ranges between 0 and 1, where 1 indicates the random distribution of elements across various classes and a value of 0.5 shows an equal distribution of elements over some classes [49].

4.4 Implementation of advanced models (CNNs)

A classic CNN encompasses alternating layers of convolution and pooling. The convolutional layers are tasked to extract patterns in the images that are located in a particular region. This task is achieved by computing the inner product of an arbitrary convolving filter and every region of the image in order to obtain a feature map. The feature map is passed through a non-linear function generating activations that are further processed in the pooling layer. The most commonly used pooling functions are average- and max-pooling, which, respectively, select the arithmetic mean and maximum of the elements in a particular pooling region. The alternating convolution and pooling layers extract features at each step. Succeeding this extraction is the non-linear function that can be chosen among tanh, logistic, softmax or ReLu activations. The final layer is the fully connected layer that outputs class in a recognition task.

Convolutional Layer: The function of a convolutional layer is to transform the input data using a group of connected neurons from the previous layer. It computes a dot product between the region of neurons in the input layer and the locally connected weights in the output layer. This provides the final output volume for the layer. The feat is achieved by a concept known as convolution.

1. Convolution: It is a mathematical operation which specifies the nature in which two sets of information are combined together. The operation is also known as the feature detector of a CNN where it applies a convolution kernel to the input and returns a feature map as output. This is achieved by sliding the kernel across the input data and multiplying kernel with the segment of data within its bounds to create

a single entry in the feature map. Finally, the activation map of each filter is stacked together along the depth dimension to construct the 3D output volume. [7]. Like any other neural network model, the parameter optimisation is performed using gradient descent. The major components of convolutional layer are as follows:

- Filters: These are one of the CNN architecture parameters which learn to produce the strongest activation to spatially local input patterns i.e. they will be activated only when the pattern occurs in the training data. With increasing depth of CNN, it is observed that the filters are able to identify nonlinear combination of features.
- Activation Maps: These are computed by sliding each filter across the spatial dimensions of the input volume during forward pass of information through CNN. A numerical value is obtained if a neuron decides to pass the information through.

2. Hyperparameters: These dictate the spatial arrangement and size of the output volume from a convolutional layer. Following are some of the most important hyperparameters:
   - Filter size: It is generally spatially small and possess three dimensions width, height and colour channels.
   - Output depth: This controls the number of neurons in the convolutional layer which are connected to the same region in the input volume.
   - Stride: This defines the sliding pace of filter per application. The depth of output volume is inversely proportional to the stride value.
   - Zero-padding: It determines the spatial size of output volume and is quite useful when maintenance of input volume spatial size is preferred in the output volume.

Pooling Layer: The layer helps to progressively reduce the spatial size of the data representation and thus prevent overfitting on the training data. These are generally incorporated between successive convolutional layers and use the maximun operation to resize the input data spatially. Pooling layers do not have any learnable parameters and generally have zero-padding.

Fully Connected Layer: This layer acts as the output layer for the network and has the output volume dimension as [1 x 1 x N] where N is the number of output classes to be evaluated. Fully connected layers have general neural network layer parameters and hyperparameters.

Our approach to an effective development for a plankton species recognition model includes understanding the core modules of the system. In order to effectively come up with a solution in the given time frame, this project concentrates on only improving the accuracy of proven models on our plankton dataset. Therefore, a pre-trained model was used for the generation of image features.

VGG16

VGG16 is a CNN model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" [46]. The model achieves 92.7 top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous models submitted to ILSVRC-2014 [46].

The plankton data is trained on the VGG-16 model as the baseline CNN model. Then, this model is applied to classify 39 classes of species within plankton and is trained for 25 epochs. Since training the mentioned models is a demanding computational task, we used transfer learning, which consists of taking models already trained for image recognition on ImageNet, an extensive dataset of non-planktonic images (Russakovsky et al., 2015 [39]).

Network Architecture: During training, the input to our ConvNets is a fixed-size 224 × 224 RGB image. The only preprocessing done is subtraction of the mean RGB value, computed on the training set, from each pixel. The image is passed through a stack of convolutional layers, where filters with a very small receptive field: 3 × 3 (which is the smallest size to capture the notion of left/right, up/down, center) have been used. The convolution stride has been fixed to 1 pixel; the spatial padding of convolutional layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1 pixel for 3 × 3 convolutional layers. Spatial pooling was carried out by five max-pooling layers, which follow some of the convolutional

layers (not all the convolutional layers are followed by max-pooling). Max-pooling was performed over a 2 × 2 pixel window, with stride 2. A stack of convolutional layers (which has a different depth in different architectures) has been followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks. All hidden layers are equipped with the rectification (ReLU (Krizhevsky et al., 2012)) non-linearity [46].

ResNet50

ResNet50 provides a residual learning framework to ease the training of networks that are substantially deeper than VGG16. These are proven to optimise easily and gain higher accuracy. On the ImageNet dataset, residual nets with a depth of up to 152 layers—8 times deeper than VGG nets exhibit lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task [15].

ResNets incorporate an underlying mapping which could be fitted on a few stacked layers which can asymptotically approximate complicated functions. This addresses the degradation problem which suggests that the solvers might have difficulties in approximating identity mappings by multiple nonlinear layers. With the residual learning reformulation, if identity mappings are optimal, the solvers may simply drive the weights of the multiple nonlinear layers toward zero to approach identity mappings. The residual learning has been adopted every few stacked layers and can be expressed as:

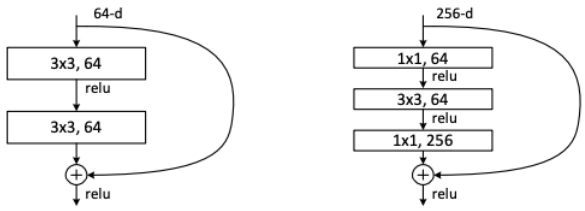


Figure 16: A deeper residual function *F* for ImageNet: a building block [15]

$$y = F(x, W_i) + x \tag{1}$$

Where x and y are the input and output vectors of the layers considered and $F(x, W_i)$ represents the residual mapping to be learned.

Network Architecture: The baseline ResNet is mainly inspired by the philosophy of VGG nets. The convolutional layers mostly have 3×3 filters and follow two simple design rules: (i) for the same output feature map size, the layers have the same number of filters; and (ii) if the feature map size is halved, the number of filters is doubled so as to preserve the time complexity per layer. Additionally, downsampling has been directly performed on convolutional layers that have a stride of 2, the network ends with a global average pooling layer and a 1000-way fully-connected layer with softmax. The total number of weighted layers is 34. It is worth noticing that the ResNet model has fewer filters and lower complexity than VGG nets i.e. 34- layer baseline has 3.6 billion floating point operations per second (FLOPs), which is only 18% of VGG-19 (19.6 billion FLOPs). Finally, the residual network is created by inserting the shortcut connections (i.e. mappings discussed earlier). The 50-layer ResNet has been created by replacing the 2-layer block in the 34-layer net with this 3-layer bottleneck block which consists of 3.8 billion FLOPs [15].

Image Features from Pre-trained Models: ResNet50 and VGG-16

In literature, many models use VGG-16/ResNet50 model which works best for large-scale image recognition. The VGG-16/ResNet50 model used the ImageNet dataset with images split into three sets training (1.3M), validation (50K) and testing (100K) divided within 1000 object categories. It achieves a very high score of 92.7% top-5 test accuracy for ImageNet [46]

The training phase consists of inputs as fixed-size 224 x 224 RGB image. The pre-processing included cropping images randomly from the rescaled training images. Furthermore, random horizontal flipping and RGB colour shift was performed to augment the training set. The stacks of convolutional layers used on the image include filters with small receptive field of 3 x 3, capturing the notion of left/right, up/down and centre. The convolution stride is fixed to 1 pixel with a padding of 1 pixel. The use of stacked small

convolutional layer has multiple benefits as compared to a single large receptive field. It incorporates multiple non-linear rectification layers which makes the decision function more discriminate. It also decreases the number of learnable parameters [46].

The process of spatial pooling is performed on five max-pooling layers whereas max-pooling is carried out over a 2 x 2 pixel window with a stride of 2. These layers are followed by three Fully-Connected layers with 4096 channels in the first two layers and 1000 channels in the last layer (corresponding to each object class). These are followed by the final softmax layer. Rectification non-linearity (ReLU) has been included in each of the hidden layers [46].

Reasons for choosing VGG-16 and ResNet50 models for the object recognition module

The requirement for the model that could work efficiently was to have high accuracy for identifying specific objects with minor differences as well as be lightweight. An intensive literature review was performed in order to choose an appropriate object recognition module as its performance is crucial to the performance of the model holistically. The VGG-16/ResNet50 [46] model achieves 92.7'% top-5 test accuracy in the ImageNet dataset. This information has been crucial in deciding which pre-trained CNN architecture to choose for the image classification model. Furthermore, use of ImageNet dataset was another reason for choosing the models as it consists of over 15 million images divided into 22,000 distinct categories which provides a good estimation of the low level features for our plankton class identifier. The VGG-16/ResNet50 architecture is the driving force for choosing it. It made a major leap from the previous work AlexNet by replacing large kernel-sized filters (11 and 5) with multiple 3 × 3 kernel-sized filters stacked one after another. This made the training and computation time to significantly less. Though, multiple better performing networks have been published after VGG-16 and ResNet50, we chose it for the simplicity and lightweight implementation it offers. However, due to the models containing depth and multiple fully-connected nodes, thus, making it a tiresome task to deploy them. Therefore we use transfer learning instead of training from scratch.

Transfer Learning from pre-trained VGG-16 and ResNet50 models

Transfer learning is a technique where a model that has been trained for one task is re-iterated to be used on another task. Such domain adaptation help exploit information that has been learned in one situation to be applied in another situation. It also helps to better learn generalisation, allows rapid progress and improves performance in the second task [40].

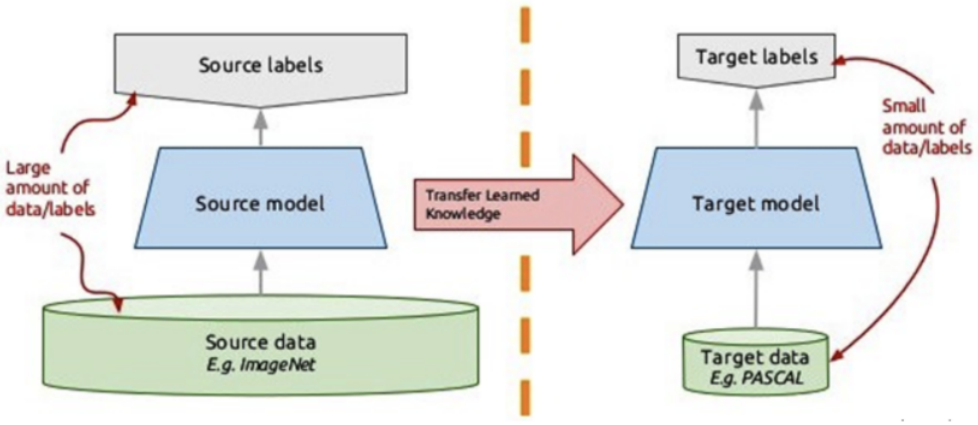


Figure 17: Transfer learning [40]

The transfer learning approach was undertaken as it is a good way to optimise our model and save time. However, the output from the final fully connected layer of the models were not useful as it is just a categorical label (for instance cat, book, etc.), therefore, it was decided to remove the final fully connected layer and extract the 4096 features from fully connected layer before it. In addition, for simplicity and faster computation, downsizing the features from 4096 to 256 using a dense layer was attempted. This however, reduced the learning capacity of our model and made it less sensitive to subtle differences in object which could lead to misidentifying objects that fall under the same category (for instance identifying woman as man). Therefore, it was finally decided to use all the 4096 features.

4.5 Model implementation using scivision

4.5.1 What is Scivision?

scivision is a new tool currently being developed at the Alan Turing Institute which aims to facilitate application of computer vision models to a wide range of images from different scientific domains. To achieve this, the tool will provide a generalised python framework capable of handling and loading image datasets in a variety of formats. scivision will provide a wrapper allowing users to efficiently run a catalogue of models and easily experiment by switching between them.

4.5.2 Implementation

The scivision team designed and provided eight notebooks showing how to access, explore and play with the challenge data. These notebooks are hosted in the challenge GitHub repository[6]. Two existing scivision functions for the version used in the challenge are described below.

The *load_dataset* function allows reading the challenge input data from a YAML file provided in repository. The file follows the structure of an *Intake* catalogue instance. The scivision team defined five datasets for the challenge: two related to input images, and the rest to labels (see the complete YAML file in the A). Whilst images are fetched as *xarray.Dataset* objects, labels are read as a *pandas.DataFrame object*. The set of challenge notebooks show how to handle both object types after being imported through scivision.

The *load_pretrained_model* function imports a given pretrained model for predicting over images loaded through scivision. The pretrained model can be defined into a local or remote source code which should contain a configuration YAML file defining the model. An example of the configuration file describing one of the models is provided in the listing1 below.

Listing 1: Configuration file defining a model.

```
1 name: plankton-models
2 url: https://github.com/acocac/scivision-plankton-models
3 import: scivision.plankton.models
```

[6] https://github.com/alan-turing-institute/plankton-dsg-challenge/tree/main/notebooks/python/scivision

```
model: resnet50_label1
args: None
prediction_fn:
    call: predict
    args:
        X: image
    kwargs: None
```

Following the above functions, two separate notebooks were designed throughout the challenge to highlight end-to-end predictions from one of the CNN models and one of the RF models. These notebooks are a work in progress and the scivision team expects to refine them in order to apply them to new plankton data from the PI system as well as other sources of images.

5 Experiments

5.1 RF

Optimisation: For better performance of our RFs model we optimise the hyperparameters. We first used the default parameters from the scikit-learn module. For hyperparameter tuning we created a parameter grid to sample from during fitting, using RandomizedSearchCV. The parameters of the estimator used to apply these methods are optimized by a cross-validated search over parameter settings. As observed in Figure 18, there is no significant improvement (accuracy between 0.698 to 0.688) from hyper-parameter tuning.

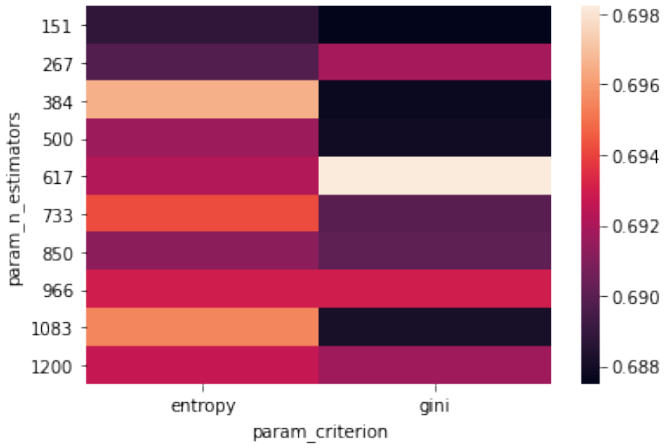
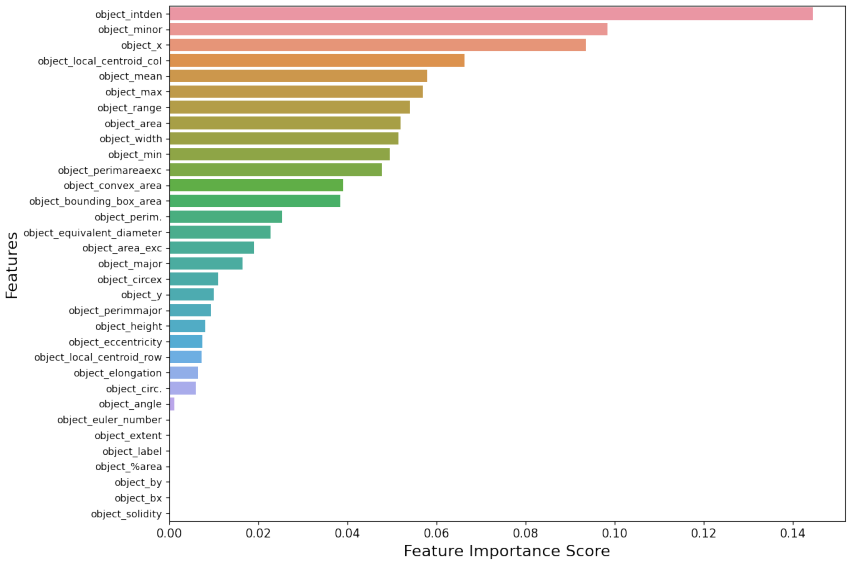
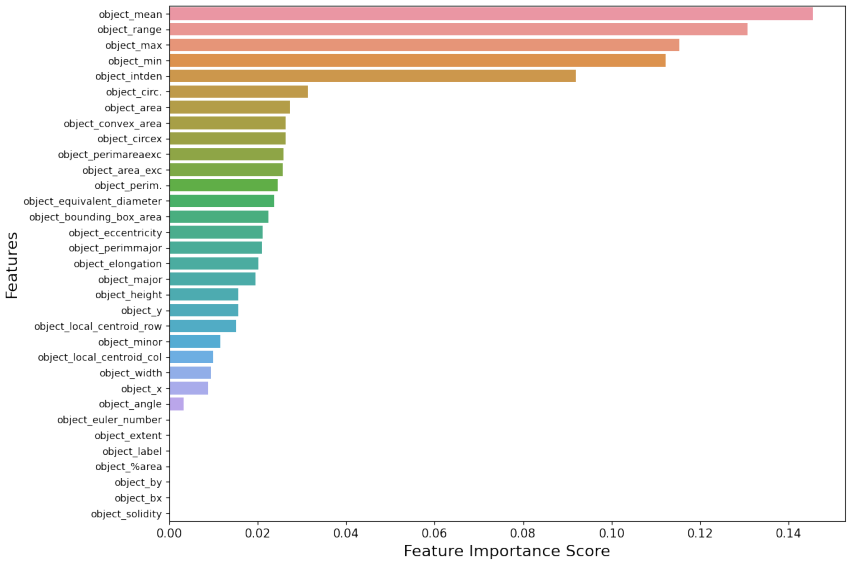


Figure 18: Parameter estimation using Entropy and Gini index.

A RF was employed as a baseline model to establish a lower bound on the classification metrics and gauge the performance of CNNs. RFs have been employed in previous investigations of plankton monitoring [4], with limited success in discriminating between plankton classes. Therefore it was expected that our CNN model would considerably outperform the RF model in the third level of classification (Label 3 - see Subsection 3.1.2). Our RF models performed well in in Label 1 and Label 2 classification, with accuracy values of 92% and 74%, respectively. However, it performed poorly in Label 3 classification, with an accuracy of only 30%. A more detailed discussion of model performance can be found in Section 5.3. The feature matrix for the RF models across all three labels was built using the 33 low-level features extracted with MorphoCut (see Subsection 4.2). Figure 19 shows the feature importance, also known as Gini importance, across Label 1 and Label 2 (with detritus removed) classification. Feature importance has been computed within the RF classifier of the scikit-learn package [32]. As illustrated, integrated density is the most dominant feature in discriminating between zooplankton and detritus, whereas the average grey value and the grey-value range are important in the discrimination between copepods and non-copepods. A detailed explanation of each low-level feature shown in Figure 19 is given in Table 8.



(a) Label 1 classification (zooplankton vs. detritus)



(b) Label 2 classification (copepod vs. non-copepod, with detritus removed)

Figure 19: Feature importance in the RF models in Label 1 and Label 2 classifications

5.2 CNNs

Setup: In deep learning literature use of transfer learning has always shown better results specially when there is shortage of training data for specific domain, in our case plankton species identification/classification with highly imbalanced dataset with 39 classes, having as low as around 35 samples

for a few classes and 40000 samples for 1 class. So we tried Resnet50 and VGG16 pretrained on ImageNet. Experimenting with Resnet50 and VGG16 we attached 1 fully connected layer after the CNN feature extractor, with number of neurons depending upon number of classes, in our case 2, 3 and 39 classes for 3 levels of plankton classification.

We used Cross Entropy Loss [53], SGD (Stochastic gradient decent) [37] with initial learning rate of 0.001 and 0.9 momentum. We also used a learning rate schedular StepLR [19] with step size 7 and gamma value 0.1. We did train test split using k-fold stratification with 10%. For all of our experiments with CNNs above settings remained the same.

Training: We started experimenting with pretrained ResNet50 + 1 Fully Connected layer (1FC), specifically we used pytorch for all of our experimentations so we used Pytorch [28] implementation and publicly available learned parameters of ResNet50 from TorchVision. We first trained the model with 256x256 input image size, scaled image pixel values to range [0.0, 1.0] as float tensor(see To Tensor from torchvision transforms). Images in dataset were of different sizes from less than 100 pixels to greater than 1000 pixels so we used bilinear interpolation to resize all images to 256x256 and trained the model for 25 epochs. During the first few epochs the model converged reaching accuracy 91% on test set, and in remaining epochs, accuracy and loss stopped improving. As our next experiment we repeated previous experiment with image size 40x40 as we thought it would be excessive with such large image size but it dropped the accuracy to 89%. The we added some augmentations like Random Gray Scale with p=0.5; Random Horizontal Flip with p=0.5; Random Vertical Flip with p=0.5 as our next experiment keeping the image size 40x40. It improved the accuracy by 1% and in subsequent experiments we increased image size to 128x128 and then again to 256x256 keeping the previous set of augmentations and it improved the accuracy [20].

Model	Image Resolution	Epochs (1 Epoch run: 1.5 mins)	Augmentation	Test Accuracy(%)
1	256x256	30	None	91
2	40x40	25	Resize 40x40;	89
3	40x40	25	Resize 40x40; Random Gray Scale 0.5; Random Horizontal Flip 0.5; Random Vertical Flip 0.5	90
4	128x128	35	Resize 128x128; Random Gray Scale 0.5; Random Horizontal Flip 0.5; Random Vertical Flip 0.5	92.2
5	256x256	25	Random Gray Scale 0.5; Random Horizontal Flip 0.5; Random Vertical Flip 0.5	93
6	256x256	25	Random Gray Scale 0.5; Random Horizontal Flip 0.5; Random Vertical Flip 0.5; Weighted Sampling with replacement	92.1
7	256x256	25	Random Gray Scale 0.5;Random Horizontal Flip 0.5; Random Vertical Flip 0.5; Weighted Sampling w/o replacement	92.3

Figure 20: ResNet50 Experiments

So far the model was not improving beyond 92.8% so we tried to handle the class imbalance using Weighted Random sampler. We assigned weights to each class relative to its contribution in dataset as the low the number of samples the higher the weight. But, it did not improve the results further therefore we did not perform any experiments on it.

Because of time constraints of the DSG, we could not repeat the same experiments with VGG16 so only trained the VGG16 with configuration with which resnet50 preformed highest. However, it could not improve results beyond 91.4%.

Evaluation metrics we used are Precision, Recall, Accuracy and F1 score for the best performed model which are available in Table 6.

5.3 Model performance

5.3.1 Classification Evaluation Metrics

The evaluation and accuracy assessment of machine learning models is essential for comparing their relative performance. There is a large literature on the accuracy assessment techniques, but given the time constraints we adopted a simple strategy that we could quickly and easily apply to

our models, namely the calculation of accuracy, precision, recall and F1-score.

Accuracy is the most intuitive performance metric as it simply measures the ratio of correctly predicted observations to the total observations. However, in the presence of imbalanced data sets, other metrics such as precision and recall must be used to asses the classifier's performance. Mathematically, accuracy is defined as:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2}$$

Where TP, FP, TN and FN in Equation 2 are the number of true positives, false positives, true negatives and false negatives, respectively.

Precision, also known as positive predictive power, indicates the fraction of relevant positive instances among all retrieved positive instances. Mathematically, it is defined as:

$$precision = \frac{TP}{TP + FP} \tag{3}$$

Recall, also known as sensitivity, is the fraction of relevant positive instances that were retrieved by the classifier, as defined by Equation 4:

$$recall = \frac{TP}{TP + FN} \tag{4}$$

The F1-score is a weighted-average of precision and recall:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{5}$$

Label 1: Zooplankton vs. Detritus		
Metric	RF	ResNet CNN
Accuracy	92%	99%
Precision (average)	90%	98%
Recall (average)	90%	99%
F1 (average)	90%	98%

Table 3: Model performance in Label 1 classification

Label 2: Copepod vs. Non-Copepod vs. Detritus		
Metric	RF	ResNet CNN
Accuracy	84%	97%
Precision (average)	71%	95%
Recall (average)	70%	95%
F1 (average)	70%	95%

Table 4: Model performance in Label 2 classification (with detritus)

Label 2: Copepod vs. Non-Copepod		
Metric	RF	ResNet CNN
Accuracy	74%	94%
Precision (average)	73%	94%
Recall (average)	72%	94%
F1 (average)	72%	94%

Table 5: Model performance in Label 2 classification (without detritus)

Label 3 with detritus		
Metric	RF	ResNet CNN
Accuracy	73%	92%
Precision (average)	18%	73%
Recall (average)	17%	71%
F1 (average)	17%	72%

Table 6: Model performance in Label 3 classification (with detritus)

Label 3 without detritus		
Metric	RF	ResNet CNN
Accuracy	30%	76%
Precision (average)	24%	78%
Recall (average)	22%	75%
F1 (average)	22%	76%

Table 7: Model performance in Label 3 classification (without detritus)

Confusion Matrix Confusion Matrix is a technique that summarises the prediction results of the classification task, by displaying the number of correct and incorrect predictions broken down by each class, Figure 21.

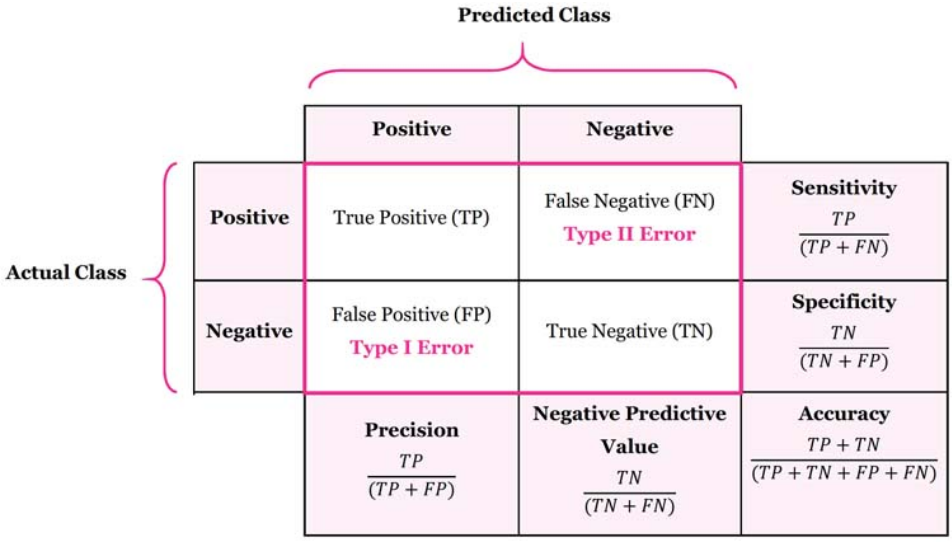


Figure 21: Confusion matrix with advanced classification metrics [50]

AUC- ROC Curve The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

- **AUC = 1**, the classifier is able to perfectly distinguish between all the Positive and the Negative class points correctly.
- **AUC = 0**, the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.
- **0.5** $<$ **AUC** $<$ **1**, there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values. This is so because the classifier is able to detect more numbers of True positives and True negatives than False negatives and False positives.
- **AUC = 0.5**, then the classifier is not able to distinguish between Positive and Negative class points. Meaning either the classifier is predicting random class or constant class for all the data points. [2]

The Receiver Operator Characteristic (ROC) curve is an evaluation metric for **binary** classification problems. It is a probability curve that plots the TPR against FPR at various threshold values and essentially separates the

'signal' from the 'noise'. In a ROC curve, a higher X-axis value indicates a higher number of False positives than True negatives. While a higher Y-axis value indicates a higher number of True positives than False negatives. So, the choice of the threshold depends on the ability to balance between False positives and False negatives.

5.3.2 Conclusions

The ResNet model performed better than the baseline RF model in terms of accuracy, precision, recall and F1 value at all three label levels. The ResNet model also maintained a high level of accuracy when more specific labels were used in classification, with an average accuracy of over 90% for all three label levels. Conversely, the accuracy, precision, recall and F1 score of the RF model decreased as the specificity of labels increased, widening the gap in performance between the two approaches.

Both models performed less accurately at classifying images at Label 3 level without including detritus than when classifying images at Label 3 level classification with detritus included, though the performance of the ResNet model was still better than the RF. This is noteworthy as it suggests the ResNet performs very accurately at distinguishing detritus from zooplankton but less accurately at distinguishing plankton species from each other. This has the potential to introduce bias in the accuracy score of the model when detritus images are included as they make up the vast majority of imagery collected. However, overall the accuracy and precision of the ResNet model presents an improvement on both the RF and previously developed models, as well as allowing the imagery to be analysed more efficiently than manual classification by domain experts.

6 Future work and research avenues

There are several avenues for future work that we have identified.

1. While our results are very promising for on-the-fly automatic classification of plankton, the machine learning models should be evaluated more carefully. There are a number of biases in the data owing to unequal class representation and a majority of detritus. We recommend a more detailed, wide-ranging evaluation including the

classification of more unseen data after filtering out empty images. We are particularly keen to explore classification performance of under-represented classes and marginal cases. It would be interesting to benchmark classification performance against human annotator performance.

2. We further recommend repeating the RF and CNN experiments with augmented images. The image augmentation experimentation demonstrated an increase in CNN accuracy, but there was too little time to make extensive use of these new images during the project.

3. The selection of low-level features used in the RF model was very simple. Future studies may want to consider using other image feature extraction approaches beyond MorphoCut. For example, [14] discuss algorithms for measuring the colourfulness of an image.

4. The implementation of the MorphoCut pipeline allowed us to prepare the data in a format that allows direct upload to EcoTaxa, should this be of interest for future manual validation. If so, we recommend exploring whether the predicted classifications from our models can be uploaded to EcoTaxa in order to use the EcoTaxa interface for fast and efficient human validation.

5. The principal component analysis and RF models both suggested that size is an important information for plankton classification. Yet, the CNN algorithms herein resized the images (which originally have different dimensions, i.e., the same resolution yields different sized images) to fit the CNN aperture. We hence recommend exploring how the size of the images, and therefore of the organisms, can be preserved within the CNN architecture and whether this will yield classification improvements.

6. We noticed some duplication in the supplied labelled data. We suspect that this is because multiple annotators have classified particular images differently, and all of these labels have been included. This is perhaps a useful indicator of uncertainty, but makes the classification task more difficult, so we removed any images with more than one label. A de-duplicated version of the dataset would be most useful for similar tasks.

7. A separate ‘unknown’ category (distinct from ‘detritus’) would be helpful for training the classifiers.
8. The current work raises important questions about transferability of the models; would the models work with other instruments operating other ocean regions? One way to test this would be a collaborative effort to pool data from several research organisations, covering a wider spatial and temporal scale.
9. A useful future avenue is the application of explainable AI techniques to the models to better understand the discriminatory variables describing plankton. Understanding how the classifiers are deciding between classes could lead to new insights in plankton morphology.
10. Consider clustering the detritus images to explore micro plastic pollution. Although the PI is primarily designed to sample plankton, images of detritus are a by-product which may prove useful for micro plastics research.

We also recommend three avenues for immediate impact:

- We suggest that a version of this dataset (bearing in mind point 6 above) be made publicly available under a liberal license to allow the scientific community to compare and benchmark algorithms. For instance, further approaches such multi-label and hierarchical classification could be tested to provide a more end-to-end multi-level classification.
- Given the unexpectedly high performance of this classifier, it has potential as a real-time index of estimated copepod abundance in the North Sea. We therefore recommend to explore why the model has a superior performance and, subsequently, whether the label2 (copepod, non-copepod and detritus) CNN model can be put into production on the Cefas vessel.
- We recommend, with further validation of the methods, preparing a submission to a peer-reviewed scientific journal.

One of the steps of the challenge included the use of a Turing-developed tool, scivision. The access to the dataset aimed to be seamless in order to allow a quick dive into the problem rather than extensive data preparation. From the

participants experiences, the scivision team gathered important feedback, which will be taken into account in the next iterations of the tool:

- More extensive documentation is needed about the libraries used for data handling (e.g. intake catalogues).
- User guide to be provided for data preparation and handling with scivision.

7 Team members

Meghna Asthana is a PhD candidate at University of York working on Computer Vision, Deep Learning and 3D graphics pipelines. She contributed towards selecting, building, improving the CNN models.

Robert Blackwell is a data scientist at Cefas. He contributed to the data preparation and accuracy assessment of the RF and CNN algorithms.

Miruna Clinciu is a postgraduate research candidate in Robotics and Autonomous Systems (CDT-RAS) at the Edinburgh Centre of Robotics. She contributed to the data preprocessing, RF modelling and optimisation.

Jennifer Ding is a Research Application Manager at the Alan Turing Institute. Her background is in Computer Vision and Natural Language Processing for the domains of urban mobility, social media analysis, and digital privacy. She contributed to image processing and augmentation, including DCGANs.

Sari Giering is a researcher in the Ocean Biogeosciences Group at the National Oceanography Centre, Southampton (UK). She produced the plankton and detritus collages, carried out the clustering for detritus, and reviewed use of CNNs in plankton research.

Charlie Hewitt is a PhD student in Geographical Information Science (GIS) at the University of Leicester, supported by Ordnance Survey. His current research interests are GeoAI and spatial data science. He contributed to the data exploration and clustering elements of this project.

Attiq ur Rehman is a software engineer with bachelor's degree in Software Engineering from Punjab University College of Information Technology, Lahore Pakistan. His current research interests are Computer Vision and

Deep Learning. He contributed to this project by conducting experiments with CNN models.

Rohit Sahoo is an engineer with a bachelor's degree in Computer Engineering from the University of Mumbai, India. His current research interests are in Time Series Forecasting, Generative Modelling and Deep Learning. He contributed to this project by working on Data Augmentation Techniques.

David Salvado Jasin is finishing a PhD at the National Centre for Combustion and Aerothermal technology (Loughborough University),in collaboration with Rolls-Royce. In August 2022 he will start a new position as Research Data Scientist at the Alan Turing Institute. David was one of the two facilitators of the team during the December 2021 DSG, worked on low-level feature extraction and helped develop the baseline Random Forest models.

Aida Mehonic is a Senior Researcher in the Tools, Practices and Systems Programme. She is leading on Research Applications at The Alan Turing Institute. Her focus is making sure that research outputs are not just openly available but that they meet stakeholder needs. Aida is leading a growing team of Research Application Managers (RAMs) whose goal is to increase the reach and maximise the positive impact of research outputs. The role of a RAM has been inspired by that of a product manager in a tech firm, but adapted for research purposes. Aida was a facilitator during this DSG project.

All team members contributed to designing the project, data analysis and interpretation, presentation preparations and report writing.

The supervisory PI role was shared between **Alejandro Coca-Castro**, **Evangeline Corcoran** and **Beatriz Costa-Gomes**, post-doctoral research associates at the Alan Turing Institute.

8

On behalf of Cefas, **Sophie Pitois** was the challenge owner.

Acknowledgements

Our thanks to the officers and crew on the Cefas R.V. Endeavour for their assistance in collecting the data.

Thanks to **Phil Culverhouse** and Plankton Analytics Ltd. for supplying the Plankton Imaging instrument and expertise.

We thank **James Scott** (University of East Anglia), **Oliver Strickson** and **Ed Chalstrey** (Alan Turing Institute) for their assistance in preparing for the challenge.

References

[1] Irwan Bello et al. “Revisiting ResNets: Improved Training and Scaling Strategies”. In: *arXiv:2103.07579 [cs]* (Mar. 2021). arXiv: 2103.07579. URL: http://arxiv.org/abs/2103.07579 (visited on 12/09/2021).

[2] Aniruddha Bhandari. *AUC-ROC Curve in Machine Learning Clearly Explained*. https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning. Accessed: 2021-12-17.

[3] Kaichang Cheng et al. “Enhanced convolutional neural network for plankton identification and enumeration”. en. In: *PLOS ONE* 14.7 (July 2019). Publisher: Public Library of Science, e0219570. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0219570. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0219570 (visited on 12/09/2021).

[4] Phil F. Culverhouse et al. “An Instrument for Rapid Mesozooplankton Monitoring at Ocean Basin Scale”. en. In: *Journal of Marine Biology and Aquaculture* 1.1 (Aug. 2015). Publisher: Ommega Internationals. URL: https://www.ommegaonline.org/article-details/An-Instrument-for-Rapid-Mesozooplankton-Monitoring--at-Ocean-Basin-Scale/423 (visited on 12/09/2021).

[5] Jialun Dai et al. “ZooplanktoNet: Deep convolutional network for zooplankton classification”. In: *OCEANS 2016 - Shanghai*. Apr. 2016, pp. 1–6. DOI: 10.1109/OCEANSAP.2016.7485680.

[6] K. L. Denman and J. F. Dower. “Patch Dynamics*”. en. In: *Encyclopedia of Ocean Sciences (Second Edition)*. Ed. by John H. Steele. Oxford: Academic Press, Jan. 2001, pp. 348–355. ISBN: 978-0-12-374473-9. DOI: 10.1016/B978-012374473-9.00290-3. URL:

https://www.sciencedirect.com/science/article/pii/B9780123744739002903 (visited on 12/17/2021).

[7] Michael Eickenberg et al. “Convolutional network layers map the function of the human visual cortex”. In: *ERCIM NEWS* 108 (2017), pp. 12–13.

[8] Jeffrey S. Ellen, Casey A. Graff, and Mark D. Ohman. “Improving plankton image classification using context metadata”. en. In: *Limnology and Oceanography: Methods* 17.8 (2019). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/lom3.10324, pp. 439–461. ISSN: 1541-5856. DOI: 10.1002/lom3.10324. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/lom3.10324 (visited on 12/09/2021).

[9] Adas Gelzinis et al. “Exploring sustained phonation recorded with acoustic and contact microphones to screen for laryngeal disorders”. In: *2014 IEEE Symposium on Computational Intelligence in Healthcare and e-health (CICARE)*. 2014, pp. 125–132. DOI: 10.1109/CICARE.2014.7007844.

[10] Sarah L. C. Giering et al. “The Interpretation of Particle Size, Shape, and Carbon Flux of Marine Particle Images Is Strongly Affected by the Choice of Particle Detection Algorithm”. In: *Frontiers in Marine Science* 7 (2020), p. 564. ISSN: 2296-7745. DOI: 10.3389/fmars.2020.00564. URL: https://www.frontiersin.org/article/10.3389/fmars.2020.00564 (visited on 12/15/2021).

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

[12] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[13] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[14] David Hasler and Sabine E Suesstrunk. “Measuring colorfulness in natural images”. In: *Human vision and electronic imaging VIII*. Vol. 5007. International Society for Optics and Photonics. 2003, pp. 87–95.

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

[16] Kaiming He et al. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[17] Kaiming He et al. “Spatial pyramid pooling in deep convolutional networks for visual recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916.

[18] Thomas Kerr et al. “Collaborative Deep Learning Models to Handle Class Imbalance in FlowCam Plankton Imagery”. In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 170013–170032. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3022242.

[19] Chiheon Kim et al. *Automated Learning Rate Scheduler for Large-batch Training*. 2021. arXiv: 2107.05855 [cs.LG].

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[21] Hansang Lee, Minseok Park, and Junmo Kim. “Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning”. In: *2016 IEEE International Conference on Image Processing (ICIP)*. ISSN: 2381-8549. Sept. 2016, pp. 3713–3717. DOI: 10.1109/ICIP.2016.7533053.

[22] Xiu Li and Zuoying Cui. “Deep residual networks for plankton classification”. In: *OCEANS 2016 MTS/IEEE Monterey*. Sept. 2016, pp. 1–4. DOI: 10.1109/OCEANS.2016.7761223.

[23] Yan Li et al. “Plankton Detection with Adversarial Learning and a Densely Connected Deep Learning Model for Class Imbalanced Distribution”. en. In: *Journal of Marine Science and Engineering* 9.6 (June 2021). Number: 6 Publisher: Multidisciplinary Digital Publishing Institute, p. 636. DOI: 10.3390/jmse9060636. URL: https://www.mdpi.com/2077-1312/9/6/636 (visited on 12/09/2021).

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

[25] Fabien Lombard et al. “Globally Consistent Quantitative Observations of Planktonic Ecosystems”. In: *Frontiers in Marine Science* 6 (Apr. 2019). DOI: 10.3389/fmars.2019.00196. URL: https://www.frontiersin.org/article/10.3389/fmars.2019.00196/full.

[26] Alessandra Lumini, Loris Nanni, and Gianluca Maguolo. “Deep learning for Plankton and Coral Classification”. In: *arXiv:1908.05489 [cs]* (Dec. 2019). arXiv: 1908.05489. URL: http://arxiv.org/abs/1908.05489 (visited on 12/09/2021).

[27] Eric C. Orenstein et al. “WHOI-Plankton- A Large Scale Fine Grained Visual Recognition Benchmark Dataset for Plankton Classification”. In: *arXiv:1510.00745 [cs]* (Oct. 2015). arXiv: 1510.00745. URL: http://arxiv.org/abs/1510.00745 (visited on 12/09/2021).

[28] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[29] Josh Patterson and Adam Gibson. *Deep learning: A practitioner’s approach*. ” O’Reilly Media, Inc.”, 2017.

[30] Anibal Pedraza et al. “Automated Diatom Classification (Part B): A Deep Learning Approach”. en. In: *Applied Sciences* 7.5 (May 2017). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, p. 460. DOI: 10.3390/app7050460. URL: https://www.mdpi.com/2076-3417/7/5/460 (visited on 12/09/2021).

[31] Anibal Pedraza et al. “Lights and pitfalls of convolutional neural networks for diatom identification”. In: *Optics, Photonics, and Digital Technologies for Imaging Applications V*. Vol. 10679. SPIE, May 2018, pp. 88–96. DOI: 10.1117/12.2309488. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10679/106790G/Lights-and-pitfalls-of-convolutional-neural-networks-for-diatom-identification/10.1117/12.2309488.full (visited on 12/09/2021).

[32] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[33] Ouyang Py, Hu Hong, and Shi Zhongzhi. “Plankton classification with deep convolutional neural networks”. In: *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*. May 2016, pp. 132–136. DOI: 10.1109/ITNEC.2016.7560334.

[34] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG].

[35] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[36] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015), pp. 91–99.

[37] Herbert E. Robbins. “A Stochastic Approximation Method”. In: *Annals of Mathematical Statistics* 22 (2007), pp. 400–407.

[38] Rrohan.Arrora. *Multi-label classification using fastai — a shallow dive into fastai data-block API*. https://medium.com/analytics-vidhya/multi-label-classification-using-fastai-a-shallow-dive-into-fastai-data-block-api-54ea57b2c78b. Accessed: 2021-12-17.

[39] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *nternational Journal of Computer Vision* 115 (2015). ISSN: 0142-7873. DOI: 10.1007/s11263-015-0816-y. URL: https://doi.org/10.1007/s11263-015-0816-y.

[40] Dipanjan (DJ) Sarkar. *A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning*. https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a. Accessed: 2021-12-17.

[41] Simon-Martin Schroder. *MorphoCut: the MorphoCut image processing pipeline library*. URL: %5Curl % 7Bhttps : / / github . com / morphocut/morphocut%7D.

[42] Simon-Martin Schröder, Rainer Kiko, and Reinhard Koch. "MorphoCluster: Efficient Annotation of Plankton images by Clustering". In: *arXiv:2005.01595 [cs]* (May 2020). arXiv: 2005.01595. URL: http : / / arxiv . org / abs / 2005 . 01595 (visited on 12/09/2021).

[43] James Scott et al. "In situ automated imaging, using the Plankton Imager, captures temporal variations in mesozooplankton using the Celtic Sea as a case study". In: *Journal of Plankton Research* 43.2 (Mar. 2021), pp. 300–313. ISSN: 0142-7873. DOI: 10.1093/plankt/fbab018. URL: https://doi.org/10.1093/plankt/fbab018 (visited on 12/16/2021).

[44] Pierre Sermanet et al. "Overfeat: Integrated recognition, localization and detection using convolutional networks". In: *arXiv preprint arXiv:1312.6229* (2013).

[45] J. McN. Sieburth, Victor Smetacek, and J Lenz. "Pelagic ecosystem structure: heterotrophic compartments and their relationship to plankton size fractions". In: *Limnology and Oceanography* 23 (1978), pp. 1256–1263.

[46] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409 . 1556 [cs.CV].

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

[48] Christian Szegedy et al. "Scalable, high-quality object detection". In: *arXiv preprint arXiv:1412.1441* (2014).

[49] Neelam Tyagi. *Understanding the Gini Index and Information Gain in Decision Trees*. https://medium.com/analytics-steps/understanding-the-gini-index-and-information-gain-in-decision-trees-ab4720518ba8. Accessed: 2021-12-17.

[50] *What is Confusion Matrix and Advanced Classification Metrics?* https://manisha-sirsat.blogspot.com/2019/04/confusion-matrix.html. Accessed: 2021-12-17.

[51] Takaki Yamada, Adam Prügel-Bennett, and Blair Thornton. "Learning features from georeferenced seafloor imagery with location guided autoencoders". en. In: *Journal of Field Robotics* 38.1 (2021). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21961, pp. 52–67. ISSN: 1556-4967. DOI: 10.1002/rob.21961. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21961 (visited on 12/17/2021).

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

[53] Zhilu Zhang and Mert R. Sabuncu. *Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels*. 2018. arXiv: 1805.07836 [cs.LG].

[54] Haiyong Zheng et al. "Automatic plankton image classification combining multiple view features via multiple kernel learning". In: *BMC Bioinformatics* 18.16 (Dec. 2017), p. 570. ISSN: 1471-2105. DOI: 10.1186/s12859-017-1954-8. URL: https://doi.org/10.1186/s12859-017-1954-8 (visited on 12/09/2021).

A Appendix

A.1 CNN architectures

Variety of Architectures

There is a large variety of network architectures, with the first architecture having been released in 2012. The flow on the progression of CNN is as follows.

- **AlexNet (2012):** The short object recognition history began in 2012, when Krizhevsky et al. [20] trained a deep CNN to classify images into 1000 different classes during the ImageNet LSVRC-2010 contest. They incorporated ReLU, dropouts and GPU implementation. The object recognition was performed by an additional SVM model. The

model achieved record-breaking results for purely supervised learning [20].

- **RCNN (2012):** Around the same time, RCNN was proposed by Girshich et al. claiming localisation and segmentation of objects by bottom-up region proposals, supervised pre-training and domain-specific fine-tuning is the secret to its performance boost[12].
- **OverFeat (2013):** OverFeat [44] implemented a multi-scale, sliding window approach which could efficiently perform classification, localisation and detection tasks. The model possessed some flaws which could be remedied, first, the localisation task was not performed by back-propagating though the entire network and second, it utilised l2 loss as opposed to IOU criterion optimisation.
- **ZFNet (2013):** In 2013, ZFNet [52] won the ILSVRC. Their model implemented information retention by using a 7x7 kernel for the AlexNet model.
- **SPPNets (2014):** Spatial Pyramid Pooling [17] had been introduced to combat the issue of fixed-size input image. The advantage of SSP-net was that it was able to generate fixed-length representation regardless of the image size/scale. Due to arbitrary sub-image pooling, it is comparatively faster than R-CNN with similar accuracy score.
- **MultiBox (2014):** The drawback of previous agnostic proposal generation approach was that it had no or an extremely weak proposal ranking system which had adverse effects on the runtime. This led to development of MSC-Multibox [48] which provided learning based proposal methods that could be paired upwith hard-engineered methods efficiently and deliver decent quality-runtime trade-offs.
- **VGGNet (2014):** In 2015, the VGGNet [46] provided a very deep network architecture with small (3 x 3) convolutional filters and layers up to 19. These proved to perform better as compared to their priors in localisation and classifications tasks.
- **InceptionNet (2015):** A well-crafted design, based on the Hebbian principle and multi-scale processing intuition, led to the foundation of InceptionNet [47] which experimented on the depth and width of

the network while maintaining the computation expenditure constant. The most efficient architecture consisted stacks of Inception modules instead of multiple parallel convolutional and maxpooling layers.

- **Fast RCNN (2015):** Following the drawbacks of R-CNN [12] and SPPNet [17] training being expensive, slow and consisting of multi-stage pipeline, a remedial approach was required. The Fast R-CNN [11] was built on the previous works R-CNN for object classification. It incorporated very deep VGG16 network which provided results nine and three times faster than R-CNN and SPPnet respectively.
- **YOLO (2016):** The YOLO [35] (You Only Look Once) approach has been based on the Multibox [48] regional proposal where it is considered as a regression problem to spatially separated bounding boxes with associated class probabilities. The architecture was fast with processing speed of 45 frames per second in real-time. The only drawback the model possessed was higher occurrences of localisation error, however, it outperformed R-CNN.
- **ResNet (2015):** Deep residual learning, ResNet [15] provided a framework of re-designed layers including residual functions with reference to the input layers. The approach proved to be optimised easier as compared to prior unreferenced functions. The ResNet ensemble achieved 3.57% error on ILSVRC in 2015 beating the average human performance (of 5-10%).
- **Faster RCNN (2015):** In contrast to the SPPNet and Fast R-CNN [11] utilising heuristic region proposal, the Faster R-CNN [36] incorporated region proposed network (RPN) for its architecture. Further, it incorporated the attention mechanism which specifically directed the network towards the right region to find the object.
- **SSD (2016):** The Single Shot Multibox Detector, SSD [24] applied a comparatively simpler approach to the prior models by completely eliminating additional object proposal step which makes it faster. This was achieved by discretizing the output space into default boxes for which the network generates a prediction score for each object category. The main giveaway from this approach was faster and more accurate results for images with varied sizes, scales and aspect ratios.

- **Mask RCNN (2017):** The Mask R-CNN [16] developed in 2018 accomplished multiple object recognition tasks with a simple and flexible framework by extending the work of Faster R-CNN [36] and implementing an addition object masking branch with minimally small overhead. It has been proven useful for tasks like estimation of human poses.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 22: Tables with network configurations for VGG [46] (top) and ResNet [15] (bottom), for different number of layers. Adapted from respective papers.

A.2 Plankton and Detritus Classes

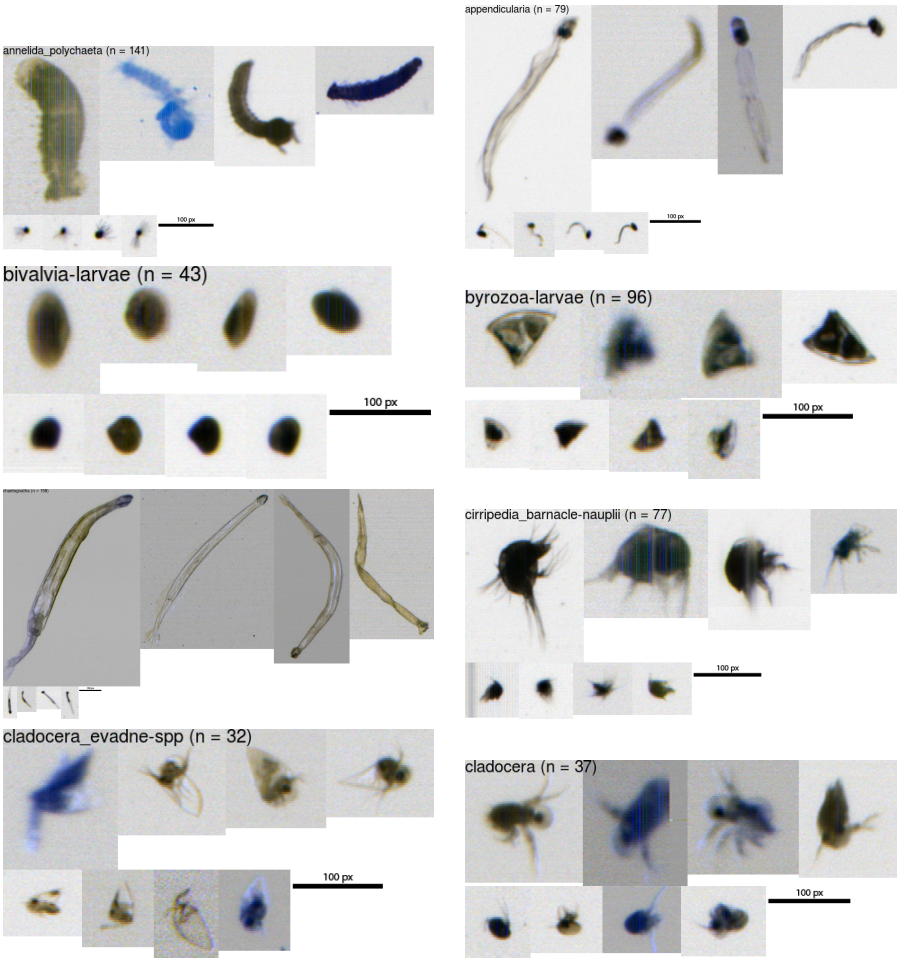


Figure 23: Plankton classes 1 to 8. Name and number of objects in class (n) are indicated in each collage.

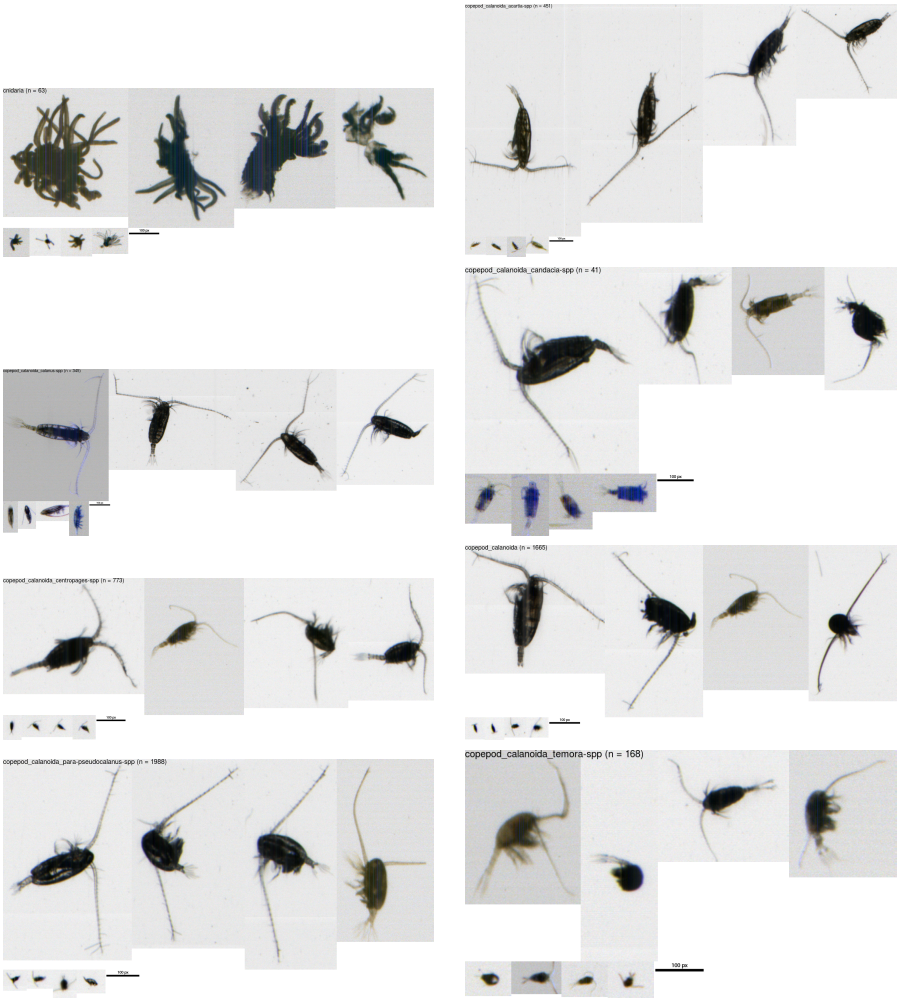


Figure 24: Plankton classes 9 to 16. Name and number of objects in class (n) are indicated in each collage.

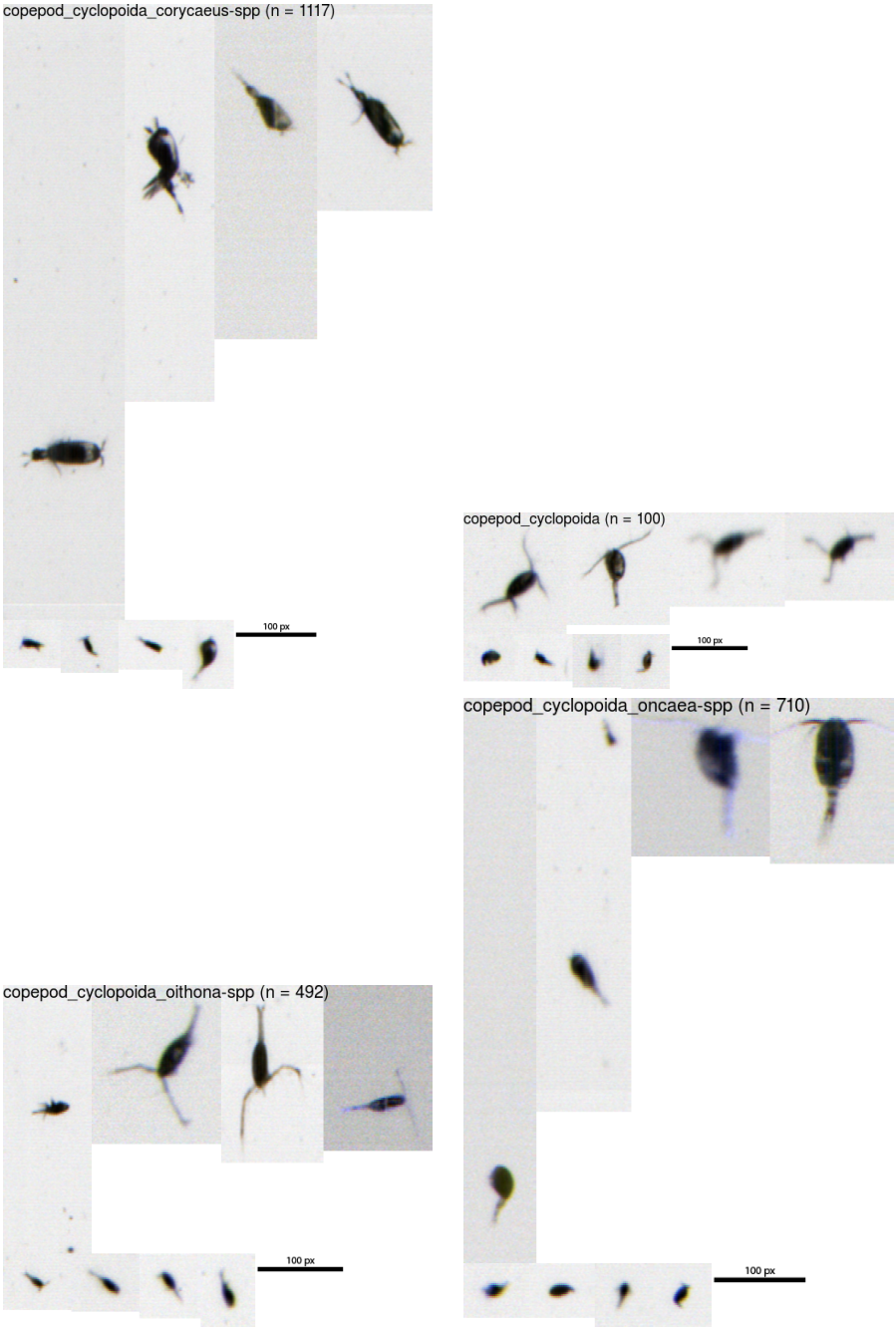


Figure 25: Plankton classes 17 to 20. Name and number of objects in class (n) are indicated in each collage.

Figure 26: Plankton classes 21 to 24. Name and number of objects in class (n) are indicated in each collage.

Figure 27: Plankton classes 25 to 32. Name and number of objects in class (n) are indicated in each collage.

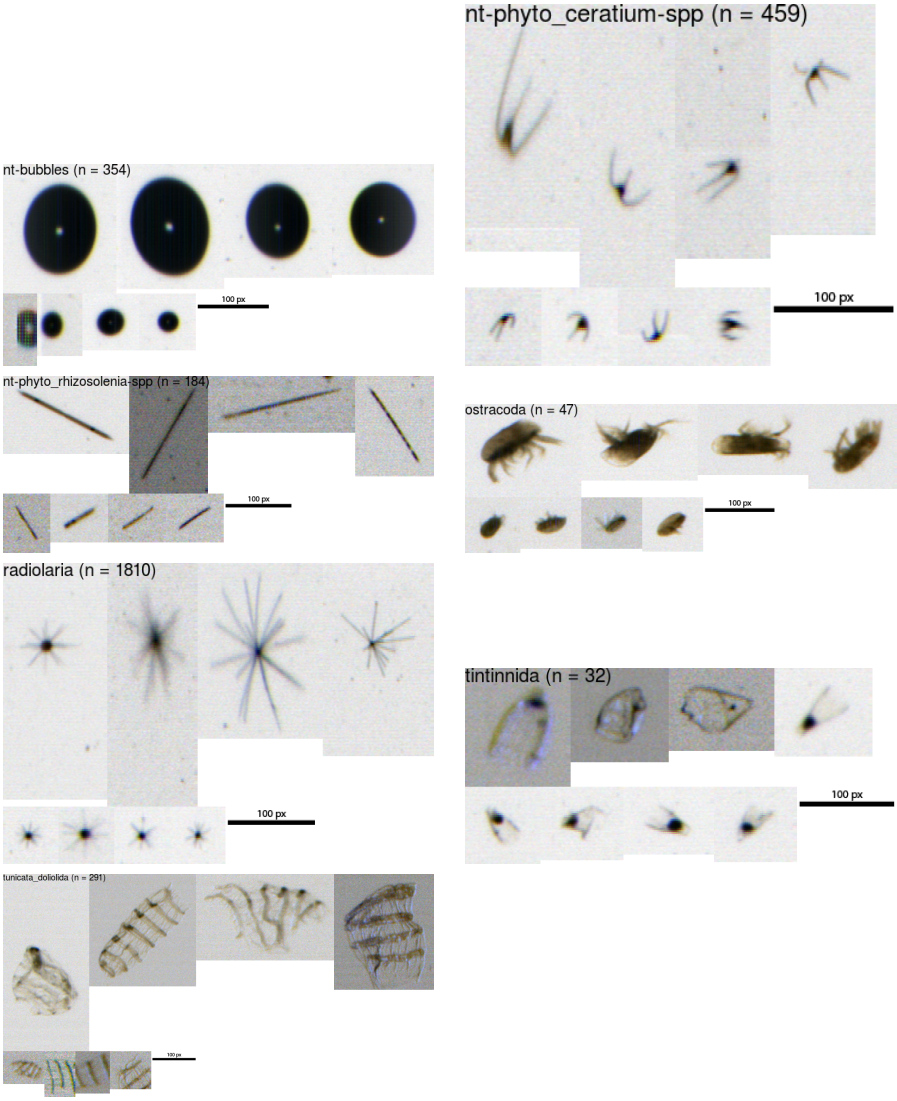


Figure 28: Plankton classes 33 to 39. Name and number of objects in class (n) are indicated in each collage.

Figure 29: Detritus clusters 1 to 4. Name and number of objects in cluster (n) are indicated in each collage.

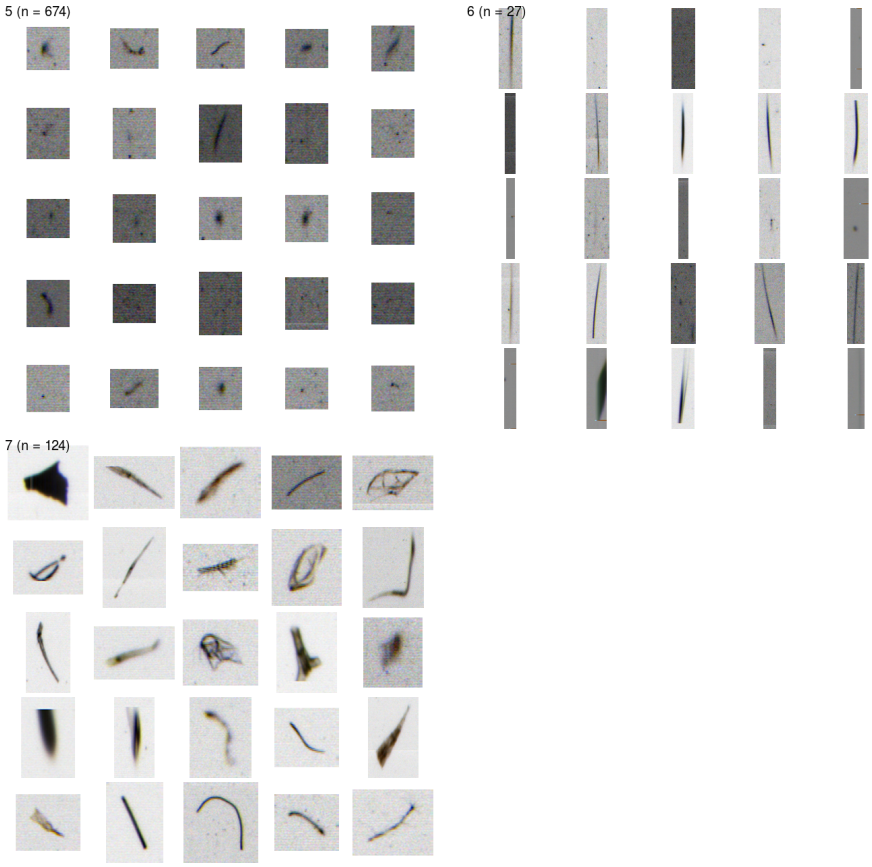


Figure 30: Detritus clusters 5 to 7. Name and number of objects in cluster (n) are indicated in each collage.

A.3 MorphoCut features

Table 8: List and definition of low-level features extracted using MorphoCut

	Feature	Name	Description
[1]	”object_label”	Label	Label of region of interest in image. Should be 1 as only one particle is detected per image.
[2]	”object_width”	Width	Width of the smallest rectangle enclosing the object (used to extract thumbnails, not really a measurement)
[3]	”object_height”	Height	Height of the smallest rectangle enclosing the object (used to extract thumbnails, not really a measurement)
[4]	”object_bx”	BX	X coordinate of the top left point of the smallest rectangle enclosing the object (used to extract thumbnails, not really a measurement)

[5]	”object_by”	BY	Y coordinate of the top left point of the smallest rectangle enclosing the object (used to extract thumbnails, not really a measurement)
[6]	”object_circ.”	Circularity	Circularity = (4 * Pi * Area) / Perimˆ2 ; a value of 1 indicates a perfect circle, a value approaching 0 indicates an increasingly elongated polygon. (it is the reverse of compactness)
[7]	”object_area_exc”	Area excluding	Surface of the object excluding holes in square pixel (=Area*(1-(%area/100))
[8]	”object_area”	Area	Surface of the object in square pixel.
[9]	”object_.area”	%area	Surface of holes in percentage.
[10]	”object_major”	Major	Primary axis of the best fitting ellipse to the object.
[11]	”object_minor”	Minor	Secondary axis of the best fitting ellipse to the object.
[12]	”object_y”	Y	Y position of the centre of gravity of the object (can be used in customized variables, do not use it directly as a measurement)
[13]	”object_x”	X	X position of the centre of gravity of the object (can be used in customized variables, do not use it directly as a measurement)
[14]	”object_convex_area”		Area of convex hull. The convex hull can be thought of as a rubber band wrapped tightly around the points that define the selection.
[15]	”object_min”	Min	Minimum grey value within the object (0 = black)
[16]	”object_max”	Max	Maximum grey value within the object (255 = white)
[17]	”object_mean”	Mean	Average grey value within the object; this is the sum of the grey values of all the pixels in the object divided by the number of pixels
[18]	”object_intden”	IntDen	Integrated density. This is the sum of the grey values of the pixels in the object (i.e. = Area*Mean)
[19]	”object_perim.”	Perimeter	The length of the outside boundary of the object
[20]	”object_elongation”	Elongation	Major / Minor (‘ellipse’ elongation)
[21]	”object_range”	Range	Max – Min
[22]	”object_perimareaexc”		Perim / Area_exc
[23]	”object_perimmajor”		Perim / Major
[24]	”object_circex”		(4 * Pi * Area_exc) / Perimˆ2
[25]	”object_angle”	Angle	Angle between the primary axis and a line parallel to the x-axis of the image (used to get object positioning, not really a measurement)

[26]	”object_bounding_box_area”	Area of bounding box	Are of the smallest rectangle enclosing the selection. Also see headings BX, BY, Width and Height.
[27]	”object_eccentricity”	Eccentricity	Eccentricity of ellipse
[28]	”object_equivalent_diameter”	ESD	Angle between the primary axis and a line parallel to the x-axis of the image (used to get object positioning, not really a measurement)
[29]	”object_euler_number”	Euler number	Here 1.
[30]	”object_extent”	Extent	Net area of feature / bounding rectangle
[31]	”object_local_centroid_col”	Centroid X	X position of the center point of the selection. This is the average of the x and y coordinates of all of the pixels in the selection.
[32]	”object_local_centroid_row”	Centroid Y	Y position of the center point of the selection. This is the average of the x and y coordinates of all of the pixels in the selection.
[33]	”object_solidity”	Solidity	Area/Convex Area.

A.4 Random Forest results

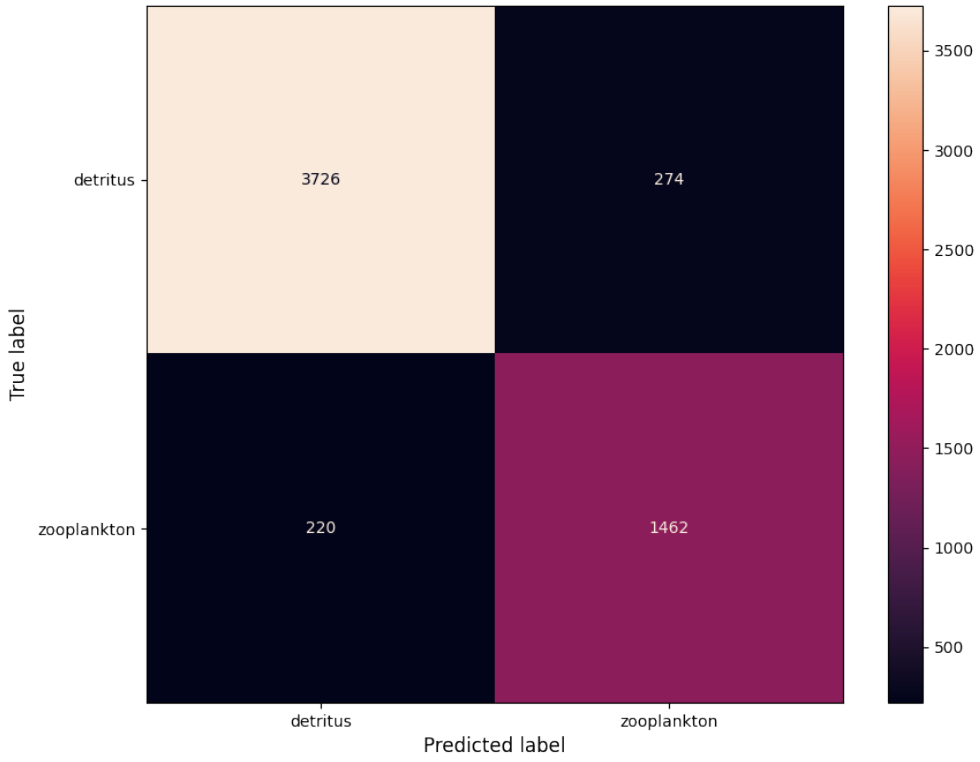


Figure 31: Label1 Confusion Matrix for RF

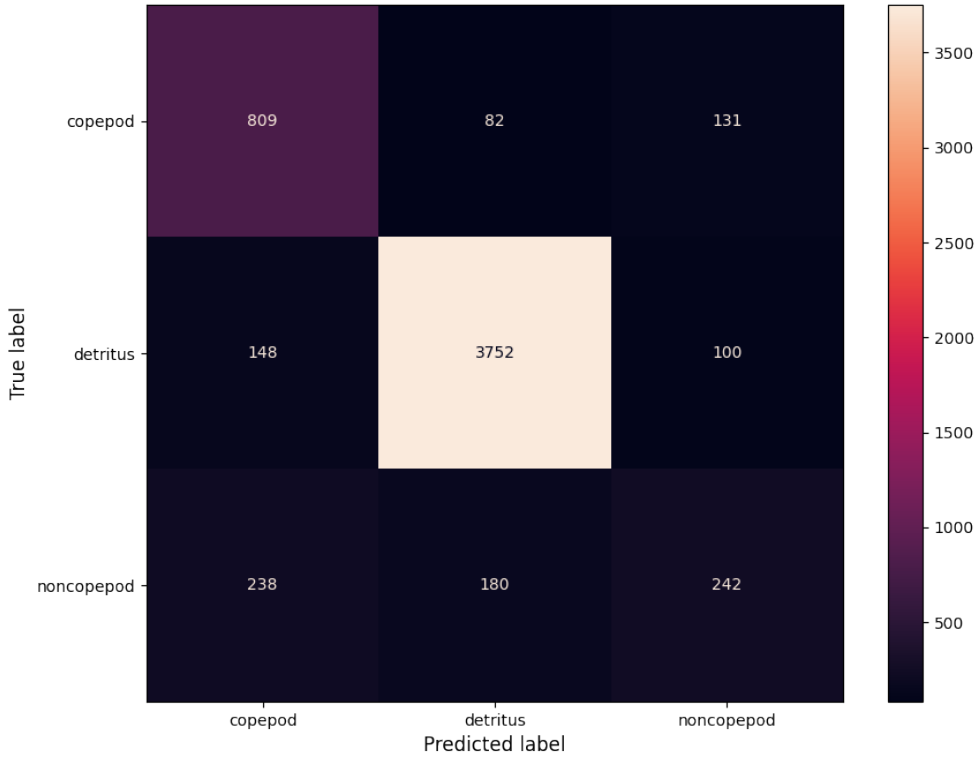


Figure 32: Label2 Confusion Matrix for RF

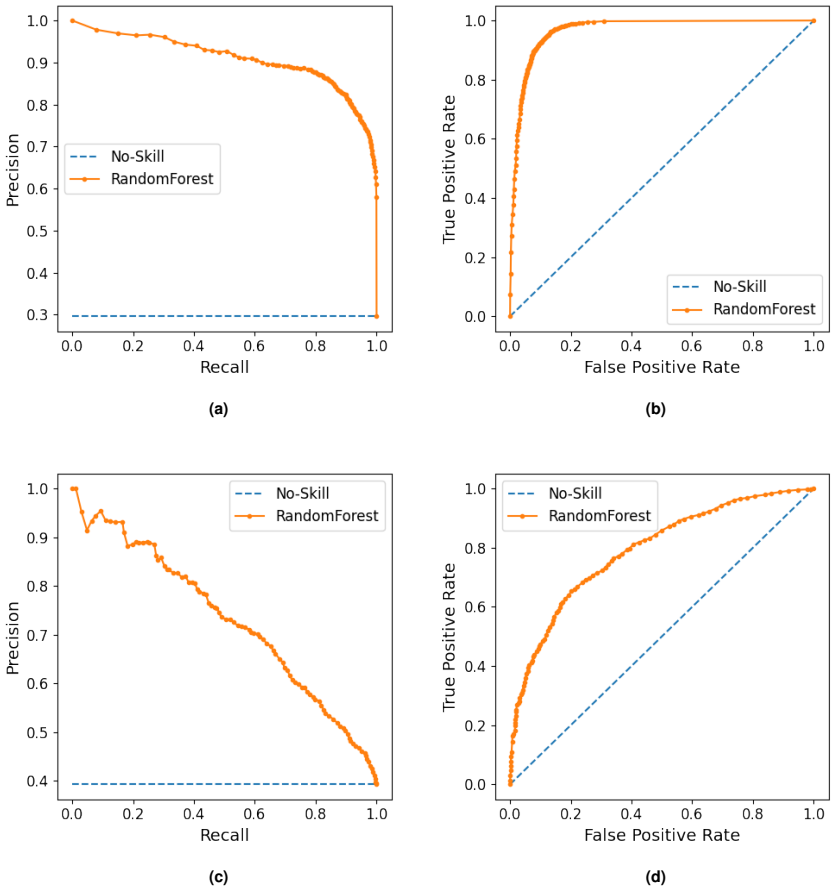


Figure 33: (a) Precision-Recall for Label1 RF; (b) ROC curve for Label1 RF; (c) Precision-Recall for Label2 RF; (d) ROC curve for Label2 RF.

A.5 ResNet50 results

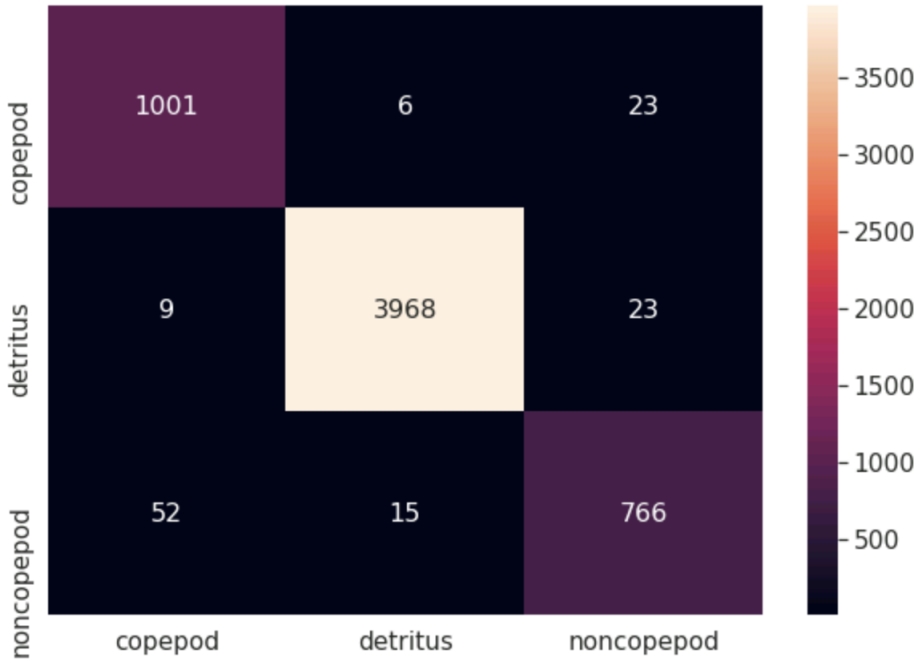


Figure 34: Label2 Confusion Matrix for ResNet50

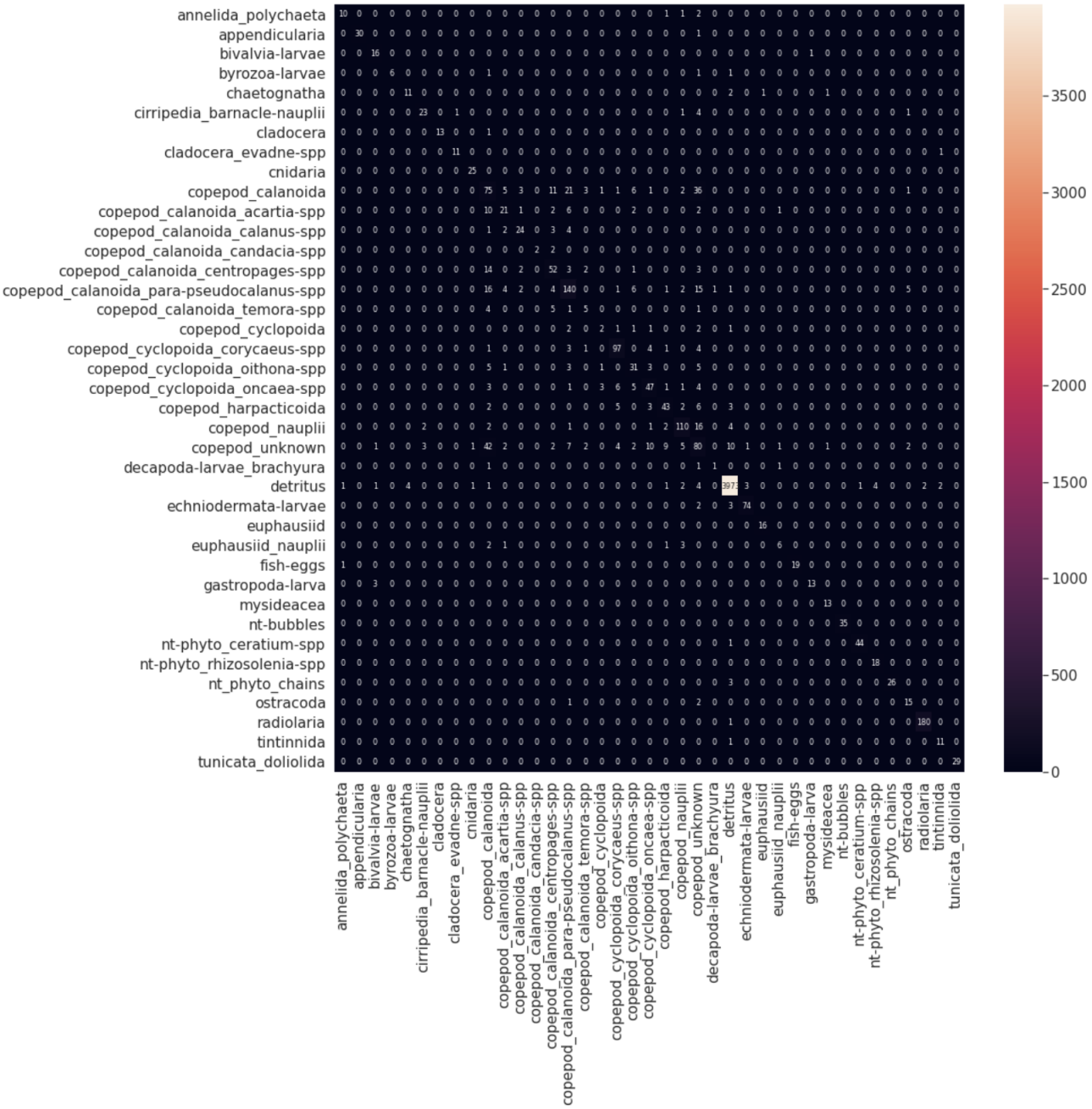
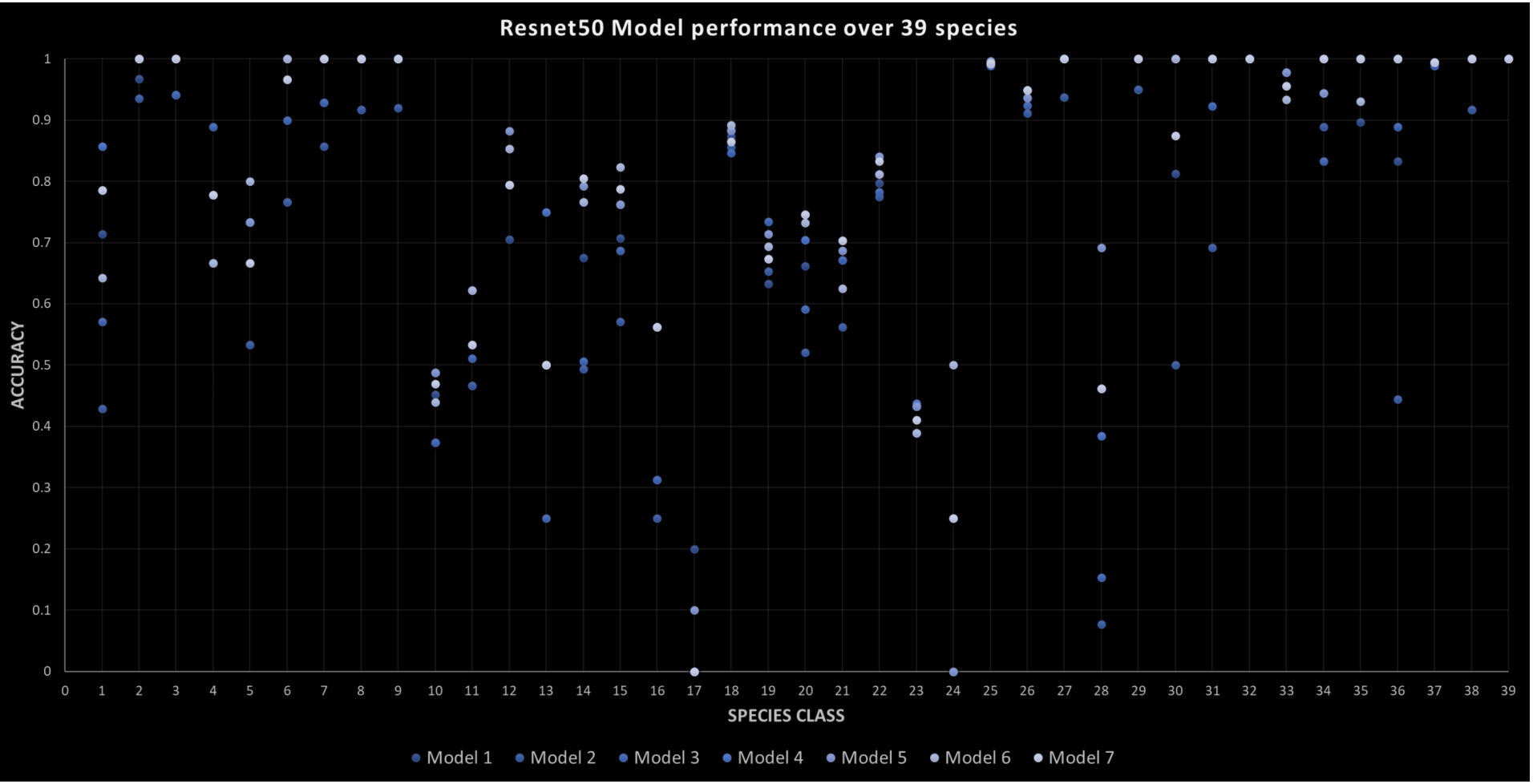


Figure 35: Label3 Confusion Matrix for ResNet50


Resnet50 Model performance over 39 species
ACCURACY
SPECIES CLASS
Model 1
Model 2
Model 3
Model 4
Model 5
Model 6
Model 7

A.6 scivision catalogue

Listing 2: Configuration file for the data catalogue to be loaded with scivision.

```yaml
sources:
  plankton_single:
      description: Load a single labeled images from CEFAS zooplankton dataset
      origin:
      driver: intake_xarray.image.ImageSource
      parameters:
        id:
          description: which filename
          type: str
          default: Pia1.2017-10-03.1726+N0029678_hc
      args:
        # Update with the correct path to the data in the Data Safe Haven
        urlpath: '/scratch/data/images/--id"".tif'
        storage_options: -'anon': True"
        exif_tags: True
  plankton_multiple:
      description: Labeled images from CEFAS zooplankton dataset
      origin:
      driver: intake_xarray.image.ImageSource
      args:
        # Update with the correct path to the data in the Data Safe Haven
        urlpath: '/scratch/data/images/-filename"'
        chunks: -"
        storage_options: -'anon': True"
        coerce_shape: [832, 1040]
        exif_tags: True
  labels_raw:
      description: ¿
```

```
29            Contains the classification labels for
               ↪ all images.
30            IMPORTANT NOTE: only use this data source
               ↪ if you intend to fetch the
31            labels for ALL of the images (both the
               ↪ test and training set)
32        origin:
33        driver: csv
34        args:
35          # Update with the correct path to the data
             ↪ in the Data Safe Haven
36          urlpath: '/scratch/data/index.csv'
37    labels:
38        description: Contains a subset of filenames
           ↪ to use as the primary working dataset for
           ↪ the challenge
39        origin:
40        driver: csv
41        args:
42          # Update with the correct path to the data
             ↪ in the Data Safe Haven
43          urlpath: '/output/data/partition/train.csv'
44    labels_holdout:
45        description: Contains a subset of filenames
           ↪ to be used as the final holdout set, for
           ↪ model assessment
46        origin:
47        driver: csv
48        args:
49          # Update with the correct path to the data
             ↪ in the Data Safe Haven
50          urlpath: '/output/data/partition/test.csv'
```

turing.ac.uk
@turinginst