IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 5, 2020, accepted August 19, 2020, date of publication September 7, 2020, date of current version September 25, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3022242*

# Collaborative Deep Learning Models to Handle Class Imbalance in FlowCam Plankton Imagery

**THOMAS KERR[1], JAMES R. CLARK[2], ELAINE S. FILEMAN[2], CLAIRE E. WIDDICOMBE[2], AND NICOLAS PUGEAULT[3], (Member, IEEE)**

[1]Department of Computer Science, University of Exeter, Exeter EX4 4QF, U.K.
[2]Plymouth Marine Laboratory, Plymouth PL1 3DH, U.K.
[3]Sir Alwyn Williams Building, University of Glasgow, Glasgow G12 8RZ, U.K.

Corresponding author: James R. Clark (jcl@pml.ac.uk)

The work of James R. Clark, Elaine S. Fileman, and Claire E. Widdicombe was supported in part by the UK Natural Environment Research Council's National Capability Long-term Single Centre Science Programme, Climate Linked Atlantic Sector Science, under Grant NE/R015953/1, and is a contribution to Theme 1.3 - Biological Dynamics. The work of Nicolas Pugeault was supported by the Alan Turing Institute under Grant EP/N510129/1.

**ABSTRACT** Using automated imaging technologies, it is now possible to generate previously unprecedented volumes of plankton image data which can be used to study the composition of plankton assemblages. However, the current need to manually classify individual images introduces a bottleneck into processing chains. Although Machine Learning techniques have been used to try and address this issue, past efforts have suffered from accuracy limitations, especially in minority classes. Here we use state-of-the-art methods in Deep Learning to investigate suitable architectures for training an automated plankton classification system which achieves high efficacy for both abundant and rare taxa. We collected live plankton from Station L4 in the Western English Channel and imaged 11,371 particles covering 104 taxonomic groups using the automated plankton imaging system FlowCam. The image set contained a severe class imbalance, with some taxa represented by > 600 images while other, rarer taxa were represented by just 14. We demonstrate that by allowing multiple Deep Learning models to collaborate in a single classification system, classification accuracy improves for minority classes when compared with the best individual model. The top collaborative model achieved a 6 % improvement in F1 accuracy over the best individual model, while overall accuracy improved by 3.2 %. This resulted in a 97.4 % overall accuracy score and a 96.2 % F1 macro score on a separate holdout test set containing 104 taxonomic groups. Based on a survey of similar studies in the literature, we believe collaborative deep learning models can significantly improve the accuracy of existing automated plankton classification systems.

**INDEX TERMS** Automated plankton identification, convolutional neural networks, deep learning, FlowCam, multi-layer perceptron model, Station L4.

## I. INTRODUCTION

Marine and freshwater plankton are a taxonomically and morphologically diverse group of organisms that span many phyla and tens of thousands of species [1], [2]. Traditionally, microscopic analysis has been used to identify and enumerate different types of plankton present within a given environment at a given point in time. However, this approach is both time-consuming and labour intensive, as it relies on the manual processing and classification of plankton present within water samples. Furthermore, it requires highly-skilled specialists who are able to discern the often subtle morphological differences that distinguish one taxa from another [3]. Given these restrictions, over recent years a considerable amount of effort has gone into automating different steps in the sampling and classification process.

In the late 1970s, the introduction of silhouette photography allowed plankton to be recorded electronically [4]. This led to the first computing systems capable of automatically enumerating and measuring planktonic particles within images [5], [6]. The following years saw the rapid development of plankton image recorders, including both in situ devices such as the Video Plankton Recorder [7], SIPPER [8], and FlowCytobot [9]; and laboratory devices such as

The associate editor coordinating the review of this manuscript and approving it for publication was Mostafa M. Fouda.

FlowCam [10] and ZooProcess [11]. Such systems can generate unprecedented volumes of plankton image data. However, without a technique to automatically and reliably classify objects within the images, their identification remains reliant upon manual classification, which introduces a bottle neck into processing chains.

Over the last few decades a considerable amount of effort has gone into developing software to automatically process plankton image data; however, success in accurately classifying plankton in natural conditions has often been mixed [12], with the ability of different systems to correctly identify minority or rare taxa being particularly poor [13], [14]. Until recently, plankton classification systems tended to be based upon traditional computer vision techniques. These involved extracting and calculating features from plankton images such as moments (applied to extract certain geometric information of interest from an image), texture and edges using manually constructed algorithms. Following this, some form of pattern recognition was implemented (i.e. a machine learning model) such that a system can learn to map a set of input features to a taxonomic group.

In one of the earliest studies, 41 morphological features were automatically extracted from plankton images and analysed using discriminant analysis [15]. The work demonstrated a 90 % classification accuracy on 8 zooplankton taxa. Later, Tang *et al.* developed the first automated, real-time approach with a dataset consisting of 2000 images belonging to 6 taxonomic groups [16]. The approach extracted four unique features by applying custom convolution filters to images. The resulting system, trained using a standard neural network, achieved a 95 % classification accuracy. Meanwhile, a classification system presented by Blaschko *et al.* used FlowCam to generate a dataset containing 982 images consisting of 13 taxonomic groups [17]. Various shape, moment and geometric features were extracted and presented to a neural network, resulting in a 71 % accuracy score. In addition, a study using ZooScan collected 1135 particles across 34 taxonomic groups from the Icelandic sea [18]. Using 26 features automatically extracted by the ZooScan system the authors achieved 75 % accuracy using a random forest machine learning model.

More recently, automatic plankton classification systems have utilised Convolutional Neural Networks (ConvNet / ConvNets) [19]. The main difference between traditional computer vision approaches and ConvNets is that ConvNets combine automatic feature extraction and pattern recognition into a single model. ConvNets represent an extension of a more basic neural network, also referred to as Multi-layer Perceptron (MLP), Fig. 1B. The neural network loosely represents a biological brain in which groups of neurons are organised in a hierarchy of layers. As an input signal flows through the network, certain neurons are stimulated which pass the signal on to all connected neurons in the subsequent layer. Using a biological brain as an example, a ConvNet can be thought of as adding a visual cortex to this model. Key to ConvNets are the numerous convolutional filters within each layer (Fig. 1Ai). These filters glide over an image like a sliding window, extracting new feature spaces from an input image. These new feature spaces are also images, meaning that ConvNets may contain any number of convolutional layers. The overarching goal of the ConvNet is to learn the filter values for each convolution function, such that the network will learn to extract important features that will allow it to correctly predict a class label given an input image.

In the field of visual object recognition research, the ImageNet dataset and competition [20] have highlighted the performance benefits of ConvNets over traditional computer vision methods. Although ConvNets are more computationally expensive to run, AlexNet [19] demonstrated that by using graphic processing units (GPU), it is now feasible to train a ConvNet within a reasonable time period (e.g. less than 24 hours). Their entry into the 2012 ImageNet competition showcased significant improvements over competitors using traditional computer vision techniques. With recent studies related to ImageNet all using ConvNets [21]–[24], research into network designs is rapidly evolving leading to further improvements in the performance of ConvNets.

There are several open source plankton image sets that have been used to test the efficacy of different ConvNets for automated plankton classification, although none are currently as large as the ImageNet dataset. These include the WHOI dataset [25], consisting of 3.4 million images covering 103 taxonomic groups; and the Kaggle dataset [26], consisting of 30,000 images covering 121 taxonomic groups. As is often the case in the real world, the WHOI and Kaggle datasets both contain many infrequently observed minority taxa. Within the field of Machine Learning research, such ''class imbalances'' are a long-standing issue [27], [28], that result in the training phase being excessively influenced by majority classes which are observed more frequently. Several studies which use ConvNets to automatically identify plankton have attempted to address the class imbalance issue. Lee *et al.* applied dual training phases, the first using a class-normalised dataset with a reduced number of images in majority classes to match the minority classes, and the second using the entire dataset [13]. This was shown to increase accuracy, yet prediction quality in minority classes remained a significant problem. Meanwhile, Dai *et al.* explored the use of Image Augmentation to address class imbalance, in which the size of the dataset was synthetically increased through the controlled modification of images within the original dataset, which boosted the number of training images for minority classes [29]. They demonstrated an approximate ten-point improvement with this technique, ultimately reaching a 93.7 % accuracy score on a dataset containing 9460 images covering 13 taxonomic groups.

A study by Wang *et al.* also addressed the class imbalance issue by training a ConvNet only on minority classes, with the entire dataset introduced after convergence [14]. Eventually, the authors proposed using the two models from this process in parallel. Upon doing this, state-of-the-art performance was achieved on the WHOI dataset, with a 95 % accuracy score.

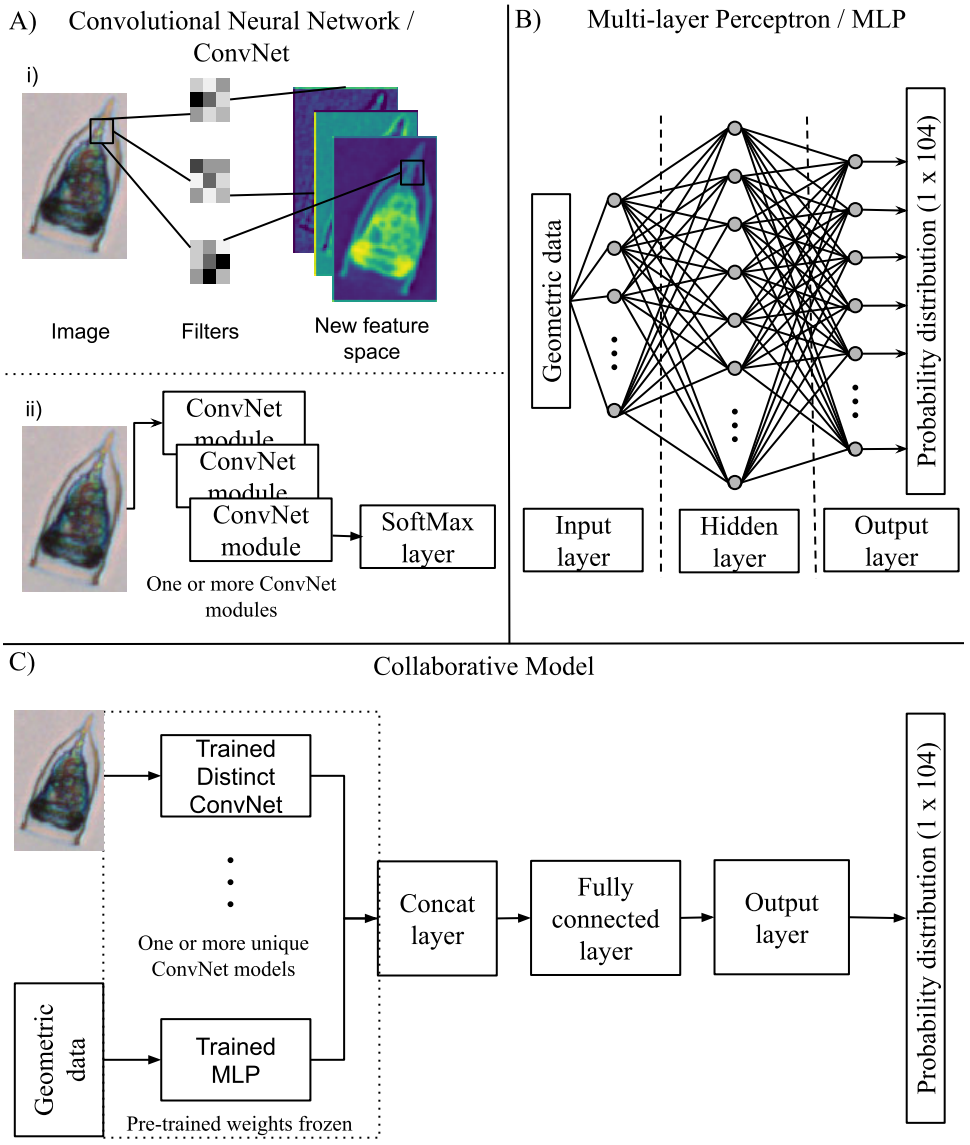


**FIGURE 1.** **Visual schematics of model architectures implemented. A) ConvNet where (i) demonstrates the operation performed in a typical convolution layer using 3 × 3 filters to produce a new feature space for a given input image. (ii) Abstract ConvNet model where the model can be made up of any of the modules shown in Fig. 5 and described in Table 1. B) Multi-layer Perceptron architecture which was trained on the geometric features of each particle. C) Collaborative model schematic whereby each model contains one or more unique trained ConvNet models and a trained MLP model. If two or more ConvNets are provided the MLP model could optionally be omitted.**

However, while there was some improvement in the classification accuracy of minority classes, this remained a problem. In more recent work, it was found that the addition of contextual data can also provide improved classification accuracy [30]. Using a small scale ConvNet architecture known as VGGNet [21], the authors experimented fusing geometric, geo-temporal and hydrographic data with associated images into the final two layers (non-convolutional) of the ConvNet. Using a dataset containing 350,000 images the authors obtained a 92.3 % accuracy score across 27 taxonomic groups with a notable 1.3-point improvement observed when using metadata compared to runs without.

To the best of our knowledge, all existing attempts to use ConvNets to automatically identify plankton within image datasets have relied on using single ConvNet architectures. However, as noted by [30], recent winners of the ImageNet competition (e.g. [19]) have tended to exploit multiple ConvNets by forming an ensemble. This is where multiple neural networks, trained with different initializations or architectures, are combined and the output predictions are either averaged or a majority voting system is implemented [31]. Within the literature, there are now multiple distinct ConvNet architectures each harbouring unique innovations and properties. While training different types of ConvNet on an

identical dataset, it is possible that each ConvNet will learn different representations and patterns, which when combined, could yield higher classification accuracies. This raises the question: can similar methods be used to improve the efficacy of current automated plankton classification systems?

Rather than constructing a typical ensemble, here we investigate the possibility of developing a classification system which allows multiple unique deep learning models to collaborate when classifying plankton image data (Fig 1C). The classification system is designed in such a way as to allow multiple pre-trained individual deep learning models to be accepted as inputs, which are then combined in an additional learning phase, to help the individual models learn how to effectively work together before forming a collective prediction. At the same time, we include many of the innovations that have previously demonstrated an improvement in classification accuracy when working with ConvNets, including image augmentation and the inclusion of contextual metadata (c.f. [30]). We test this system on an image set containing 11,371 images and covering 104 taxonomic groups, which were collected using the automated plankton imaging system FlowCam. The image set contained a severe class imbalance, and particular attention was given to the performance of the collaborative system in correctly predicting minority classes when compared with non-collaborative architectures.

In this paper, we describe the image dataset, the four unique ConvNet models configured within this study and an MLP model for additional contextual data, along with the approach that allows the individual models to work collaboratively when classifying images. A description of the approach used to prepare images for the classification system, including image augmentation, is also included. Details of the performance of individual learners compared against the new collaborative models is provided. Finally, we discuss our results, including the prediction quality of the collaborative models compared to individual learners, with a focus on minority taxonomic groups, and suggest new areas of research that may further improve the use of machine learning to accurately and efficiently identify and quantify natural plankton communities.

## II. METHODS

In this section we detail the full work flow required to construct a plankton classification system, including the collection of live plankton, the processing of samples through a FlowCam imaging system, constructing a data processing pipeline and developing suitable methods to construct novel deep learning architectures suitable for use with the subsequent data.

### A. DATA ACQUISITION

Live plankton samples have been collected using vertical hauls of a standard WP2 net fitted with a 63 $\mu$m mesh net from station L4 in the western English Channel (www.westernchannelobservatory.org.uk [32]; Fig. 2) on a

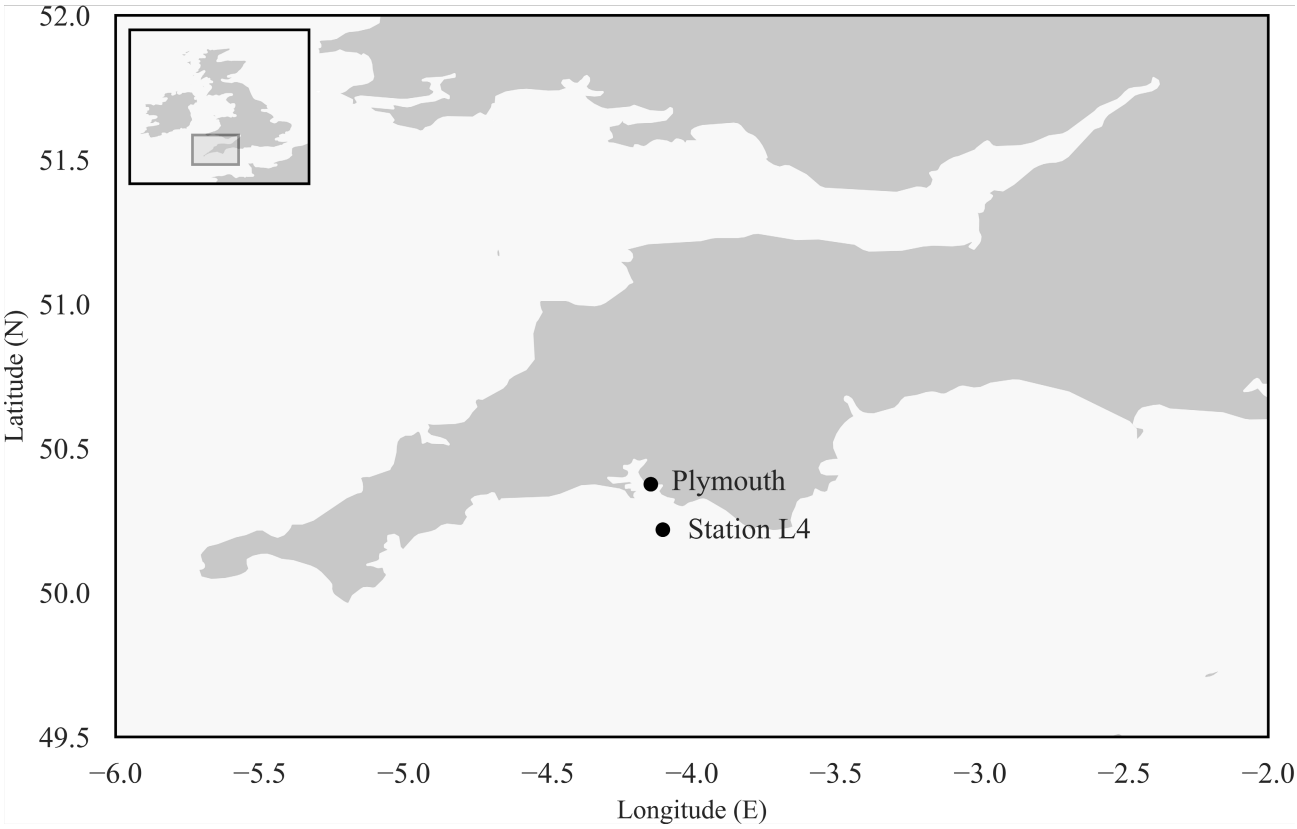


**FIGURE 2.** Map showing the location of the long-term monitoring site Station L4 which forms part of the Western Channel Observatory (WCO). The WCO is located in coastal waters near to the city of Plymouth, UK.

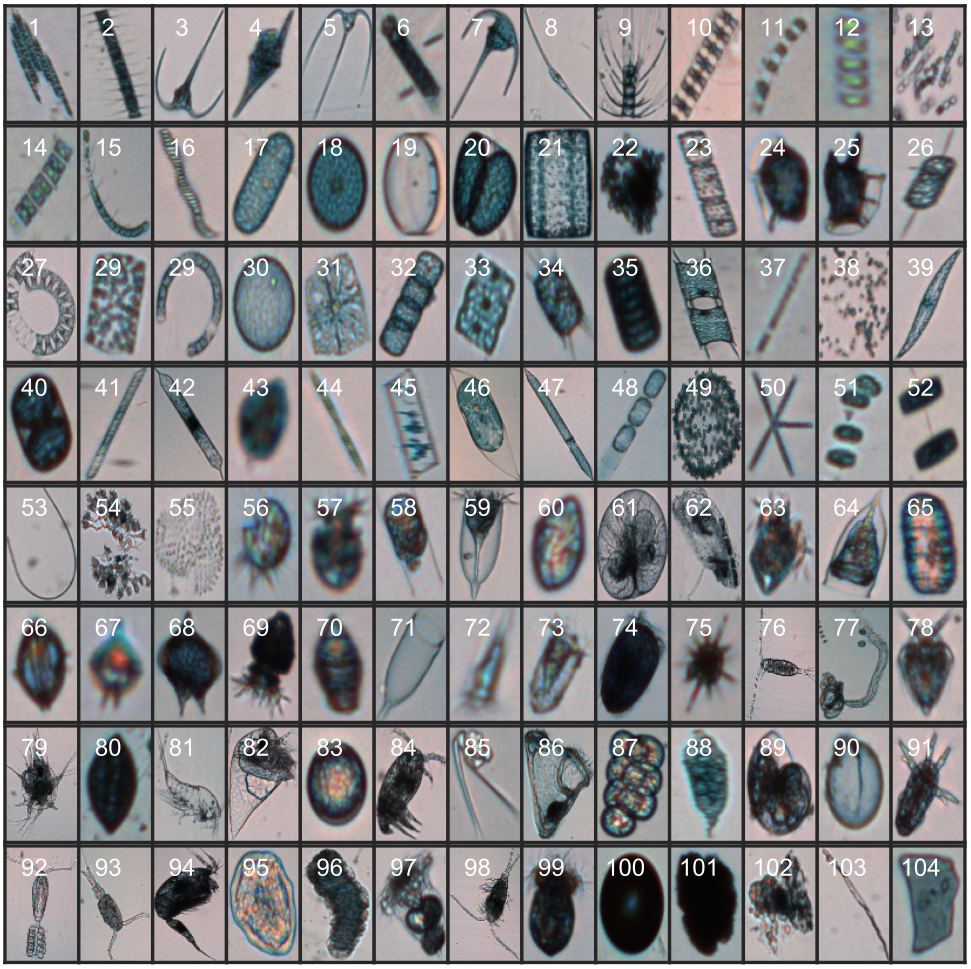


**FIGURE 3.** Sample images, resized for publication purposes, showing the 99 plankton and 5 non-plankton classes used in this study. Details of scientific names and sizes ($\mu$m) for each class are provided in Table S1.

weekly basis since 2012. For this study we focussed on data generated from the live analysis of samples collected between 2016 and 2017. Sub-samples of the net material were analysed using a FlowCam VS-IVc (Yokogawa Fluid Imaging Technologies Inc.) fitted with a 300 $\mu$m path length flow cell, and a 4$\times$ microscope objective. Images were collected using auto-image mode at a rate of 6-12 frames per second. Image files were manually classified to determine the abundance of protists and metazoa using VisualSpreadsheet® software (Version 4.1.95). Image libraries were created for each taxon or cell type with the number of images in each library ranging between 14 and 637. The final dataset consisted of 11,371 sample images belonging to 104 taxonomic groups (Fig. 3). Each image library contained single image collages with a corresponding file consisting of geometric and time data. The dataset contained a natural class imbalance (Fig. 4), with several of the minority classes containing just 14 unique particles while majority classes each contained more than 500 particles.

### B. ConvNet ARCHITECTURES

The structure of ConvNets is an active area of research, with new architectures being continuously developed. Here, we investigate the use of four distinct architectures that are commonly used within the literature (Table 1). These form the basis for the four individual ConvNets used within the current study. The first is VGGNet [21], which introduced applying 3 $\times$ 3 convolution filters to shrink the number of parameters required while still identifying similar patterns within images compared to applying larger filters which were common practise previously, e.g. AlexNet [33]. Various best practices were also introduced in VGGNet which are currently widely followed when setting up a new ConvNet classification system. The second is based on the Inception module, which formed the backbone of GoogLeNet [22]. The Inception module contains numerous functions, whereby all are operated in parallel on an input image. These include various sized convolutional filters, a pooling function to aggregate information to a smaller image dimension and 1 $\times$ 1 convolution filters to reduce the number of parameters used. The third is the Residual Network [23], which introduced a method to construct deeper networks without reducing accuracy. By introducing an identity function between layers, the learning algorithm can skip through the network when updating weights between neuron connections. The fourth is DenseNet [24], a network in which the output of each layer is

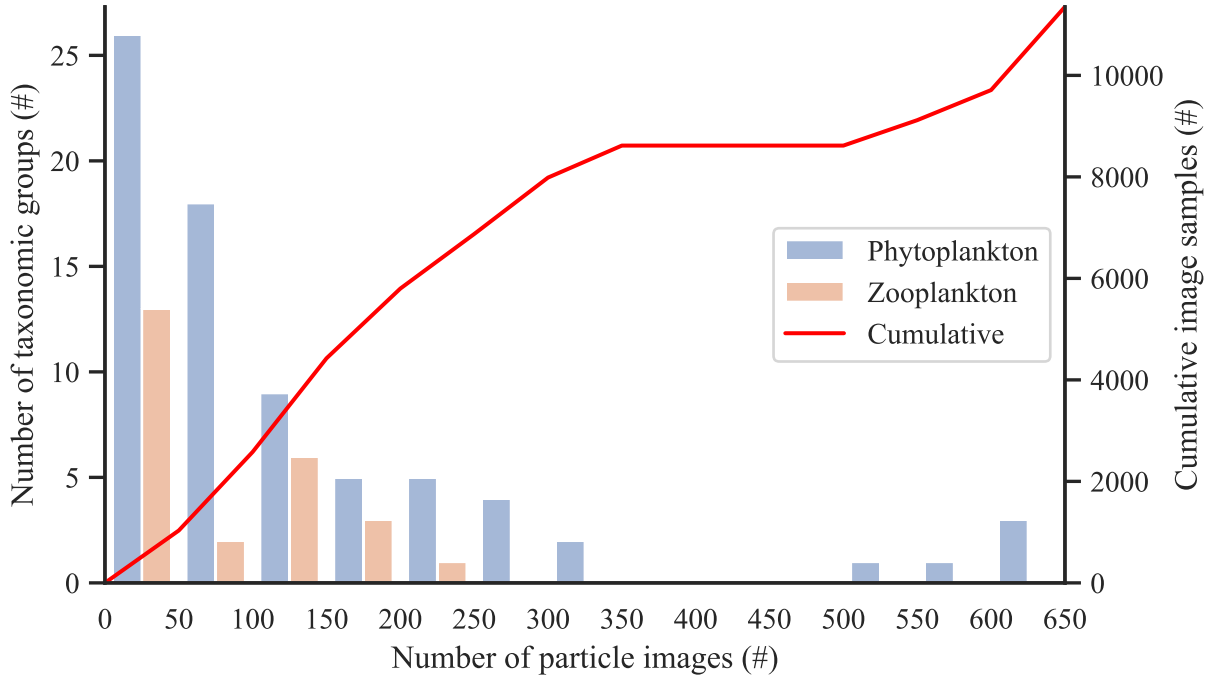


**FIGURE 4.** Variation in the number of particle objects across distinct phytoplankton and zooplankton taxonomic groups. A particle corresponds to a single image and its accompanying geometric data, as generated by FlowCam. The cumulative number of particles is shown in red. In total, the dataset contained 11,371 particles covering 104 taxonomic groups.

**TABLE 1.** Brief summary of four recognised ConvNet architectures that inspired the fundamental building blocks used in this study. References: Simonyan and Zizzerman (2015) [21], Szegedy et al. (2015) [22], He et al. (2016) [23] and Huang et al. (2017) [24].

| ConvNet model | Key features | Reference |
|---|---|---|
| VGGNet | VGGNet introduced the repeating convolution, relu activation and batch normalization pattern within ConvNets that form the basis of the most widely-recognised architectures. | Simonyan and Zisserman (2015) |
| GoogLeNet | The GoogLeNet model contains numerous Inception modules which allows multiple tasks to be carried out in parallel. These include 1x1 convolutional and max pooling layers to reduce parameters and aggregate information, standard 3x3 convolutional filters to extract fine grain details and a 5x5 convolutional filter to cover larger receptive fields of input. | Szegedy et al. (2015) |
| ResidualNet | ResidualNet is distinguished by its shortcut connections between an input layer and a convolutional layer deeper in the network. This allows layers deeper in the network to learn from input and assist in dealing with the problem of a vanishing gradient. | He et al. (2016) |
| DenseNet | DenseNet connects the output of each convolutional layer to every subsequent layer in the network. This allows layers in the model to learn from any previous layer in the network. | Huang et al. (2017) |

connected to every subsequent layer in the network, allowing layers deep in the model to learn from an input image.

A straightforward approach would have been to implement these models in full as described in the original papers. However, initial exploration found the models to be too complex for the FlowCam image data. This led to unnecessary computation and models that tended to over fit, meaning they learnt the training data almost perfectly, but had poor classification accuracy when presented with novel data. The four selected ConvNet models were originally tested against the ImageNet dataset, containing more than 1 million high resolution images spanning 1000 categories. Consequently,

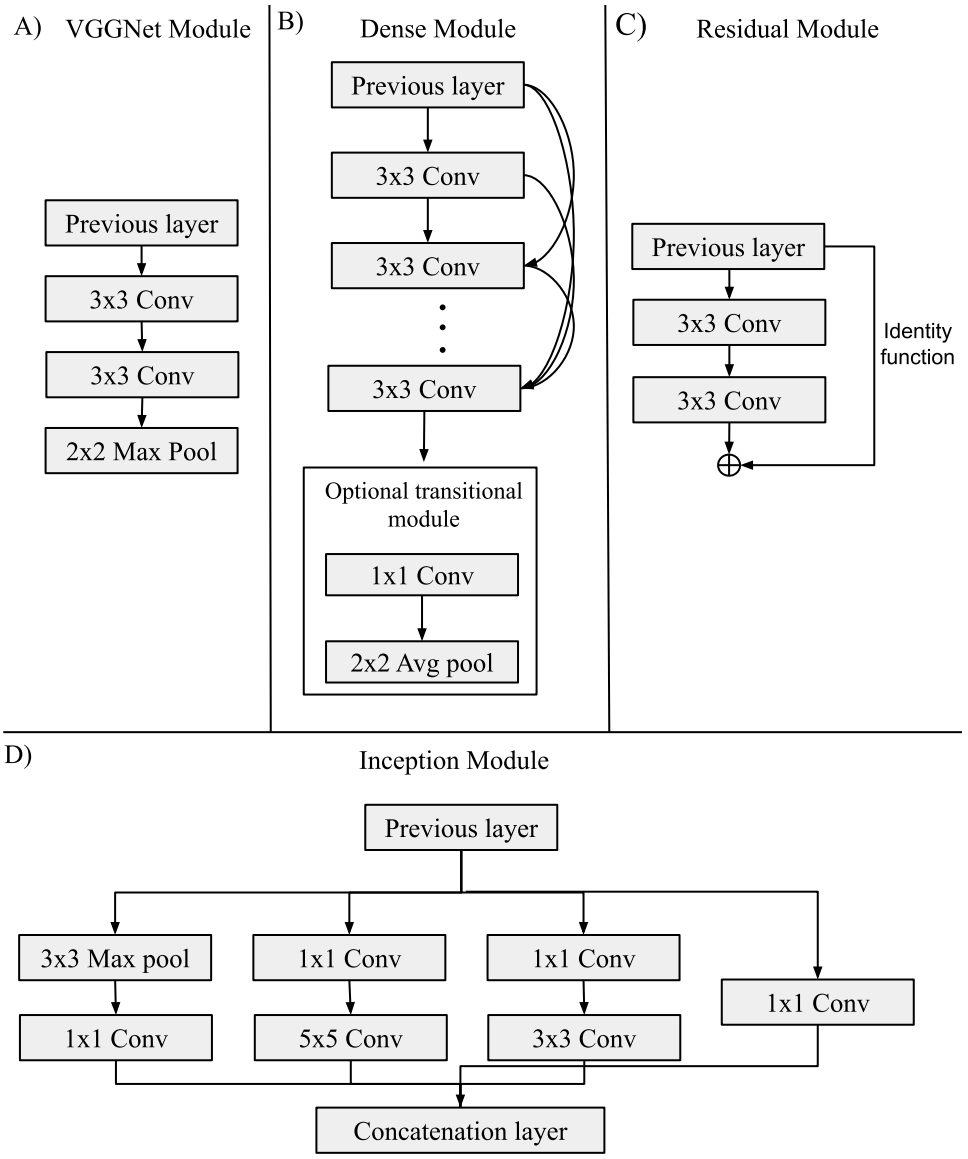


**FIGURE 5.** Schematics of the four types of ConvNet modules implemented, whereby each one can be stacked arbitrarily to form a full ConvNet model. Further details are provided in Table 1. A) VGGNet inspired. B) DenseNet inspired. C) ResNet inspired. D) GoogLeNet inspired.

these networks were wider and deeper to allow for complex mappings between input and output signals. Similar observations were noted by Ellen *et al.* [30]. Therefore, we extracted the main innovation from each model in a way that meant the ConvNet could be scaled to custom depths and widths.

While each of the four selected ConvNet architectures introduced a new concept, they contain a similar characteristic. The new concept is repeated throughout the network to form a deep learning model. The only variable is usually how many convolutions should occur at certain depths of the network (the width). Therefore, we took the approach of extracting the main innovation behind each selected ConvNet (Fig. 5) and implemented it in a standalone piece of Python code. For each model type, the standalone Python code was implemented in a way that made it possible to stack modules when constructing a ConvNet of a given depth (Fig. 1Aii). Each individual module was designed to accept parameters so that the width of any layer in the module (i.e. the number of convolution filters) could be adjusted; this was done according to the configuration found in the original paper that the module was based on [19], [21], [23], [24]. These implemented modules provided a basis for further experimentation to learn suitable ConvNet depths for the given FlowCam image data.

### C. MULTI-LAYER PERCEPTRON CONFIGURATION

Following Ellen *et al.*, we also investigated the impact of passing numeric, geometric and environmental data to the classification system [30]. The geometric data is generated automatically by FlowCam (a full list of variables is provided in Table S2). Since these variables are in an arbitrary order, a ConvNet is not suitable for this type of data, hence we opted

to handle this data using fully connected layers. One option was to let these data interact in the last few fully connected layers of the ConvNet model, as proposed by Ellen *et al* [30]. However, when testing this approach, we observed that the geometric data tended to have a disproportionate influence during the early stages of training, suggesting that the data are simpler to learn than the corresponding image data. Due to uncertainty regarding how this would affect the final convergence of the ConvNet model, we felt it necessary to perform training on the geometric data in isolation. Therefore, we constructed a separate MLP model to learn from the numeric context data.

Unlike a typical ConvNet, training times are drastically reduced using an MLP since only a one-dimensional array is passed to the network. A single particle's geometric data input is approximately 440 times smaller than its associated image (352 bytes vs 155,136 bytes) resulting in training times of minutes rather than hours on the workstation used for the study (Table S3). This meant that a suitable architecture could be learnt from a grid search on a range of different network configurations. To learn a suitable design for the network we tested all combinations of between one and five hidden layers; and 256, 512, 1024 and 2048 neurons per layer. The MLP model was highly susceptible to overfitting. Therefore, we included a dropout layer [34] between the hidden layer and output (SoftMax) layer. Dropout randomly deactivates a certain ratio of neurons (decided by a parameter) from the previous layer for every training step. These neurons then play no role as a signal flows forward through the network or during weight updates during the back-propagation phase. This forces the network to be adaptive during training, allowing it to learn robust features with less susceptibility to noise in the training data. For the MLP model, the dropout value had a high impact on validation accuracy; hence using the model architecture learnt from the network configuration grid search, we applied a second grid search to learn how many neurons should be removed during a training step in the last fully connected layer. Dropout was also included in the ConvNet models with the commonly accepted default value of 0.5 [13], [35]. Within the limits of the study, it was not computationally feasible to perform the same tests using the more complex ConvNet models.

### D. COLLABORATIVE DEEP LEARNING MODELS

Each model contains novel design choices and was trained in isolation on their respective datatypes to allow them to converge in their own time with no interference. With these steps complete, a method was implemented to allow multiple deep learning models to effectively work together (Fig 1C).

To construct a collaborative model, each individual learner (this term is used to help differentiate between previously trained models and the collaborative model) is loaded with their trained weights. Since each individual model had been optimised in isolation and had already converged, there was no need for the collaborative model to resume learning of the weights for the individual learners. Therefore, the weights for each individual learner were frozen when they were loaded into the collaboration. Upon loading an existing model into the collaborative model, the output SoftMax layer for every individual learner was removed. In its place, every model's previous output before the SoftMax layer was connected to a new concatenated layer. At the essence of the collaborative model is a method that allows these individual learners to effectively work together. To implement this, we introduced a fully connected layer containing 512 neurons. This layer acts as a new function for the collaborative model to learn how each individual model should contribute to the final prediction. Finally, a new single SoftMax layer was added to form the output of the collaborative model, providing a single probability distribution for the network. A learning phase is then initiated with the same training and validation set as earlier to learn the weights between the concatenated and fully connected layer.

### E. IMAGE PROCESSING

For each taxonomic group, one or more image collages were collated and labelled by a specialist taxonomist. Within an image collage there were several particles, with one image saved per individual. Accompanying each image collage are ASCII text files that record information about each sample, including the image collage filename and the coordinal location of the particle within the collage. Using this information, we implemented an algorithm (Fig. 6A) to segment each particle image and store the resulting image as a separate file. A CSV file was also generated that stores the file location of this image, alongside the additional geometric features extracted from the text file.

To efficiently train a ConvNet, numerous inputs can be presented to the network in parallel. The caveat is that ConvNets require all inputs to have identical dimensions. To ensure the images were uniformly sized, we analysed the individual particle images. The widths and heights were highly variable, with the smallest size being just 14 pixels, while the largest had 941 pixels. To improve results in the image resizing process, images that had a width longer than their height were flipped to portrait. To decide the new image resolution, we calculated the new median value for each dimension rather than the mean, since the extreme values were outliers. This resulted in a new desired image resolution of $64 \times 101$. Images were interpolated to this common size using bilinear interpolation (examples of the resulting images can be seen in Fig. 3). Adding borders to retain aspect ratio was tested but initial tests using the ConvNets showed this decreased performance, while there was some concern that the model would learn border patterns in certain classes. Finally, as an additional step every pixel value was divided by 255 to normalise pixel intensities.

### F. FEATURE ENGINEERING

We applied the following feature engineering steps (Fig. 6B) to enhance the performance of the MLP model trained on the additional contextual data FlowCam generated.

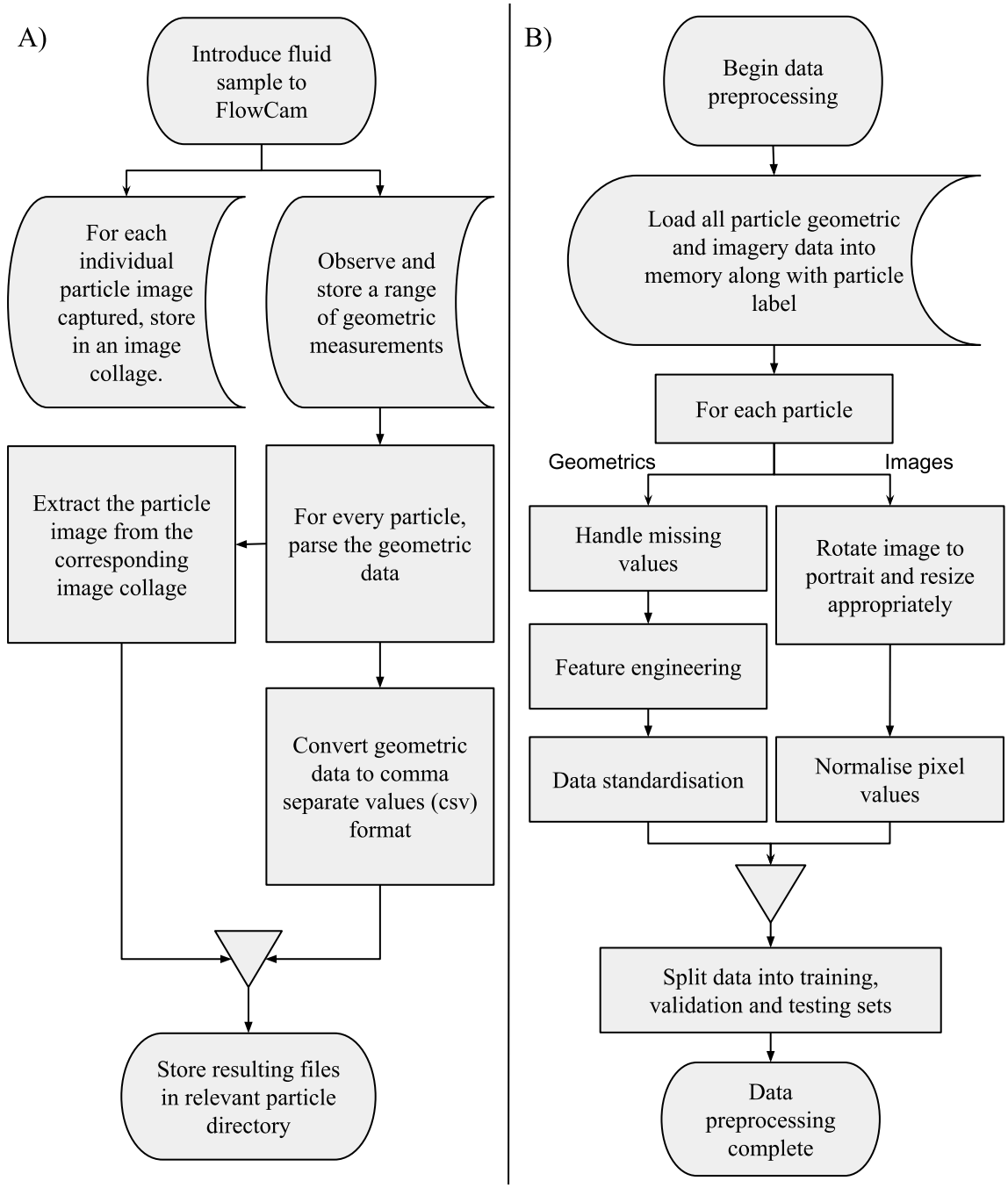


**FIGURE 6.** Flow charts showing the steps taken to prepare particle images for automatic classification. A) FlowCam processing steps, during which multiple images of individual particles are taken and descriptive numeric data generated. One single image per particle is saved along with its accompanying numeric data. B) FlowCam post-processing steps, during which the individual images and their accompanying numeric data are prepared for automatic classification. At the end of this step, the image set is broken up into the training, validation and testing datasets.

- Encoding cyclical features – FlowCam stores a timestamp as each particle is processed through the system, this was converted to seasonal format. To preserve the cyclical format of this data, cosine and sine transformations replaced the original seasonal value.
- Log transformation – Given the wide variation in geometric measures, which often spanned several order of magnitude, we applied log transformations [36] on all measurement based features such as length, area and diameter.
- Derived features – Researchers can intuitively understand relationships between measurements such as height and width. To provide the MLP model with the same intuition we calculated ratios between two features. Embleton *et al.* sets previous precedence for creating new features from existing features for a

plankton classification system [37]. These new features include calculating the ratio between the perimeter and the area, the ratio between the maximum and minimum ferret measurements, and the ratio between the perimeter and the length. The full feature list can be found in Table S2.

### G. DATA STANDARDIZATION

The range and scale of measurements varied widely between different features. This can result in some features having more importance when training a machine learning model. Therefore data standardization is a common technique which can be applied to put all features in the dataset on an equal footing [16], [38], [39]. To calculate this, the mean and standard deviation are computed for each individual feature. Then for each data point of the feature, the feature's mean is taken away and the result divided by the standard deviation. This results in a normalised dataset in which every feature has a mean of zero and a standard deviation of one.

### H. DATASET SPLITTING

The images and geometric features were split into training, validation and testing sets with taxonomic labels to match accordingly. The training set is used during the learning stage of a neural network to adjust the model's weights between neurons depending on how far it is from the ground truth (labels provided to the network). The validation set is provided such that at the end of each training epoch the model is tested on a separate dataset to obtain a measure of how well the model is generalising to novel data. We used this to identify overfitting and to allow training to be halted as necessary to save computation. By also checking the validation accuracy the best model was saved if it had outperformed any previous model. This is beneficial since occasionally models would tend to overfit as training progresses leading to a model with poor generalisation if a snapshot of the model was only saved after convergence. Finally, the testing set (also known as a holdout set) was not used during training. This set is used after all training has completed to indicate a model's performance on novel data that has not previously been presented to the network. Eventually, 8187 images were used in the training set and 1592 images in the validation and test sets.

### I. IMAGE AUGMENTATION

Lack of training data is a challenge for many real-world datasets, and small imagesets are a widely recognised problem in the object detection and recognition field [40]. In a realistic setting, plankton imagery belonging to the same taxonomic group will inherit variability in posture, rotation and size. However, for minority classes, this variability was often missing, which prevents the ConvNet from gaining a full understanding of a given class. Given this and the relatively small size of the dataset, we applied image augmentation to the dataset; a technique used to train most state-of-the-art ConvNets [19], [21], [23], [24]. Image augmentation artificially generates additional training images using a variety of image processing techniques such as rotations, translations, shears, and horizontal flips (Fig. 7). Since some classes in the dataset contained so few particles, an aggressive augmentation was implemented to allow the model the opportunity to gain a deeper understanding of these classes. All augmentations were applied randomly to original training images just before they were presented to the network. The parameters

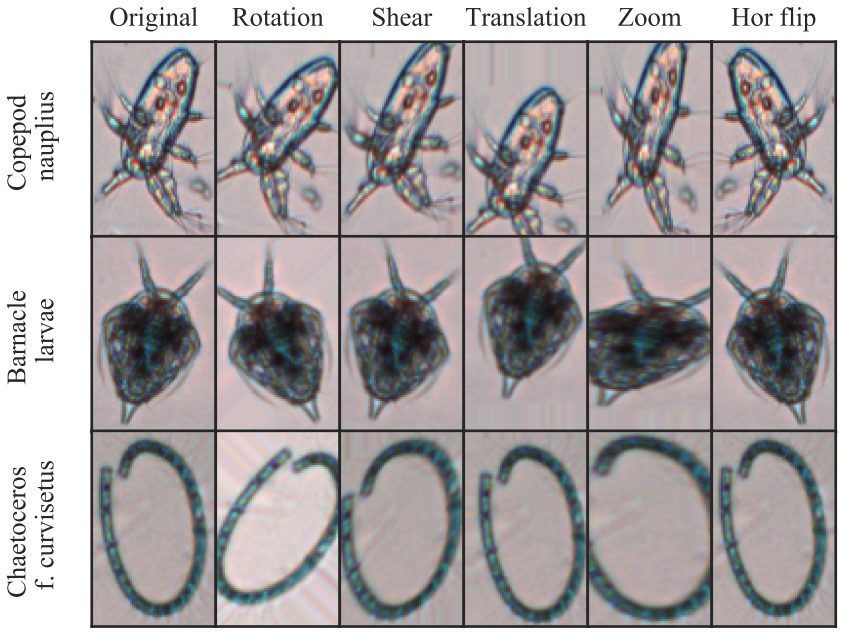


**FIGURE 7.** Image augmentation techniques as applied to individual images of three organism classes. Image augmentation is applied randomly to each image, meaning images may be altered to a lesser or greater extent, or not at all, as shown in the figure.

used for image augmentation can be found in Table S3 in the supplementary information.

### J. ASSIGNING CLASS WEIGHTS

The additional data gained from image augmentation is beneficial; however, the imbalance throughout the dataset remains since additional data are generated for all classes. To address this problem, the optimization function can be adapted to account for class imbalance [41]. In this study, we supplied class weights according to prior knowledge of class probabilities [42] to the neural network. These are set appropriately such that samples from minority classes are given a larger weight then those from a common class. These weights are used when the model's weights are updated during the back-propagation phase of training. Therefore, weights are adjusted more aggressively when the network is presented with a minority class while only slight adjustments of the weights are employed when a particle from a majority class is presented. The calculation for a given class weight value $w_i$, where $K$ is the total number of samples in the dataset, $N$ is the number of classes in the dataset and $K_i$ is the total number of samples belonging to class $i$ is given by the equation:

$$w_i = \frac{K}{NK_i} \tag{1}$$

### K. TRAINING CONFIGURATION

Training a deep neural network (especially ConvNets) is computationally challenging, meaning it is extremely expensive to do hyper parameter tuning. Despite this there are commonly accepted defaults used throughout the literature that we have also employed here:

- Activation function – Individual neurons are each a basic computational unit which decide whether to activate depending on the input signal. The ReLU activation function [43] is the most commonly used activation function in the recent ConvNet literature due to short computation and fast convergence times relative to competing activation functions such as Tanh or Sigmoid. ReLU is a simple linear function that returns the value entering the neuron or 0.0 if the number is 0.0 or less. The benefit of the function outputting a zero value is that this creates sparse representation throughout the network, which simplifies the model and assists in improving the convergence rate. While another advantage is that it drastically reduces the vanishing gradient problem, caused by the saturation of using other activation functions.
- Optimization algorithm – During the back-propagation phase of training, weighted connections between neurons within a neural network are optimised using gradient descent such that the loss function of the network is minimized. Algorithms to implement this have progressed with all current state-of-the-art ConvNets applying the Adam (Adaptive Moment Estimation) optimiser [44]. Adam is an extension of stochastic gradient descent, which applies techniques such as using the first and second moments of gradients to calculate the learning rate for each individual weight.
- Learning rate – This parameter indicates how much the optimisation algorithm should adjust the weights. A default of $1 \times 10^{-3}$ was selected for all networks, which is a common default applied throughout the literature. This allows the model to learn at a reasonable rate while ensuring the algorithm finds a suitable local minimum in the loss function. We also implemented plateau detection in training such that if the validation accuracy has not improved for a set amount of epochs the learning rate is decreased by a factor of 0.5 to allow the model to find the minimum of the valley that the loss function is residing in.
- Batch size – Due to the parallel nature of neural networks, numerous inputs may be presented to the network simultaneously, yielding reduced training times. The batch size describes how many inputs are given to the network during a training step. This was set to 500 for ConvNet / collaborative models and 5000 for MLP.
- Epoch – An epoch corresponds to a network seeing all data points in the training set once. Thus, the number of training steps per epoch is the number of particles in the training set divided by the batch size. At the end of each epoch various functions are run, including calculating the validation accuracy, adjusting the learning rate if necessary, and saving a snapshot of the model if it's required. Early stopping was also implemented such that if there was no improvement in validation accuracy after 13 epochs for ConvNet / collaborative models or 80 epochs for MLP, training was halted.

### L. EVALUATING MODEL PERFORMANCE

An intuitive approach to measuring accuracy is calculating the ratio between correctly classified taxa against the total number of particles in the test set. However, this can give misleading scores when working with imbalanced datasets, since models can perform well on classes that have abundant data, while their performance for minority classes is masked. To put every class on an equal footing when contributing to the model's score regardless of size, we calculated the F1 macro score. This score is the harmonic mean of the recall and precision scores. The kappa score was also calculated as an additional metric to test the performance of the collaborative models against individual models.

- Recall score - indicates the ratio of samples that were positively predicted against the sum of all negative and positive values. Recall calculates what proportion of a specific class were correctly predicted and calculated as:

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{2}$$

- Precision accuracy - describes the ratio of observations that were positively predicted for a given class against the false positives. This metric detects how precise the

model is at detecting certain classes and is calculated as:

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \quad (3)$$

- F1 score - considers both the precision and recall by calculating the harmonic mean between precision and recall accuracy that emphasizes the minimum value. The F1 score is calculated as:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (4)$$

- Kappa score ($\kappa$) - provides a classification accuracy score normalized by the imbalance of the dataset calculated as the following where $p_o$ is the observed agreement of a label, and $p_e$ the proportion of agreement between each set of labels corrected for chance.

$$\kappa = \frac{(p_o - p_e)}{(1 - p_e)} \quad (5)$$

## III. RESULTS

### A. SCALING ConvNet MODULES

For each ConvNet module implemented (Fig. 5), we assessed the effect of scaling these to learn suitable depths and widths for each ConvNet model. These were scaled based on configurations found in the original paper they were based on [21]–[24]. To form a baseline for each model type we constructed a simple model consisting of an input, one module and a SoftMax output layer. After convergence of any model the performance was recorded on a holdout test set and a decision undertaken to determine whether tests on the current model type should continue to be scaled. The factors considered included tracking the top-1 test accuracy (also referred to as validation accuracy) and the F1 score to assess if both scores were stagnating. The computational cost was also considered such that if a certain model was going to take too long to converge, testing for a model type was halted. However, if scaling was computationally cheap, testing was continued regardless as was the case with DenseNet models. It is worth noting that due to the stochastic nature of neural networks, the results could vary on different runs. Due to computational restraints, we were unable to perform cross validation or produce error bars.

The results of this process are shown in Table 2 and confirm the consensus that arbitrarily constructing deeper models does not guarantee improvement [23]. The models based on the inception modules showed remarkable consistency regardless of depth. This is attributed to the inception stem where the model has carried out more learning than the other models before the input even reached the first module. The design of the inception modules enabled the construction of deep networks without an exponential increase in the parameters, allowing our tests on Inception inspired models to reach eight modules. In comparison with DenseNet the number of parameters increased so significantly in the five-module version that testing had to be halted. We included tests with residual connections to allow inception modules deep in the network to learn from earlier Inception modules. This adaptation offered arguably little improvement yet did offer the best F1 score of all the models.

Despite offering worse results, DenseNet based models showed resilience to overfitting compared to other models. We eventually removed dropout since in this case it was a hinderance, since the widths for each layer were smaller than that of the competing models (for example 12 convolution filters in the first layer compared to 64 in Residual Net), removing neurons during the last layer restricted the possible network mappings. Overfitting was still present so a small dropout value could have been beneficial, but we lacked resources to test this. The fact that DenseNet models were unable to reach accuracy scores of the other models indicated that the number of filters within each layer was too low. For example, the reduced number of convolution functions in the early layers were missing features in the input image such as particular edges or colour patterns that other models were identifying. In the future it may prove beneficial to test this model based on the 169 or 201 configurations in the original paper [24].

ResNet uses an identity function to allow layers deep in the network to learn from the original input, resulting in the construction of ever deeper networks capable of gaining additional insight. The deepest Residual model containing five modules demonstrate this with the top validation accuracy of 0.933 although it offered a negligible improvement over the three or four module version. However, the F1 score appears to peak at 0.899 in the three-module version meaning that improvements in deeper models were coming from accuracy improvements in majority classes. Furthermore, connecting earlier layers to every other subsequent layer came at a computational cost. The five-module version containing 20 million parameters took approximately 24 hours to train with the resources available in the current study. The six-module version at 78 million parameters would have taken several days to converge (Table S3), therefore we were unable to conclude if an even deeper network would have increased prediction accuracy. It is worth noting that the claim in the original ResNet paper was that the residual architecture fixes problems affecting performance in very deep networks, so it is unclear if this model type was much benefit for shallower networks.

VGGNet offered the most surprising results given its comparatively simple architecture (we initially expected this to act as more of a baseline model). Eventually the four-module version provided 0.942 validation accuracy, the best results posted by any model. While the 0.901 F1 score was only just exceeded by the seven-module Inception (with residual connections) model. This result suggests that when attempting to scale back existing ConvNet architectures to work with plankton image data, further simplifications could have been made. For example, each of the other ConvNet models contained convolution layer and pooling layers designed to extract large receptive fields and down sample $228 \times 228$ input images. Considering the VGGNet inspired model did

**TABLE 2.** Experimenting building up ConvNet architectures using the fundamental building blocks inspired by recognised ConvNet architectures described in Table 1. Top-1 Test acc, also referred to as the validation accuracy is the performance of the model on previously unseen data.

| Inspired from | # Modules | Train Acc. | Top-1 Test Acc. | Top-3 Test Acc. | F1 score | # Params |
|---|---|---|---|---|---|---|
| DenseNet | 1 | 0.912 | 0.860 | 0.949 | 0.767 | 74k |
| DenseNet | 2 | 0.985 | 0.925 | 0.979 | **0.879** | 252k |
| DenseNet | 3 | 0.983 | **0.926** | 0.983 | 0.872 | 1.3m |
| DenseNet | 4 | 0.988 | 0.922 | 0.984 | 0.874 | 2m |
| DenseNet | 5 | 0.978 | 0.909 | 0.976 | 0.837 | 3m |
| Inception | 1 | 0.986 | 0.926 | 0.982 | 0.883 | 320k |
| Inception | 2 | 0.996 | 0.928 | 0.980 | 0.876 | 739k |
| Inception | 3 | 0.995 | 0.935 | 0.986 | 0.886 | 1.1m |
| Inception | 4 | 0.993 | 0.930 | 0.980 | 0.872 | 1.6m |
| Inception | 5 | 0.992 | 0.927 | 0.983 | 0.881 | 2.1m |
| Inception | 6 | 0.996 | **0.936** | 0.982 | 0.890 | 2.7m |
| Inception | 7 | 0.994 | 0.935 | 0.985 | **0.903** | 3.6m |
| Inception/Residual | 1 | 0.991 | 0.930 | 0.987 | 0.896 | 370k |
| Inception/Residual | 2 | 0.995 | 0.935 | 0.986 | 0.899 | 900k |
| Inception/Residual | 3 | 0.997 | 0.933 | 0.982 | 0.894 | 1.5m |
| Inception/Residual | 4 | 0.997 | 0.937 | 0.986 | 0.891 | 2.3m |
| Inception/Residual | 5 | 0.997 | **0.940** | 0.984 | 0.889 | 3m |
| Inception/Residual | 6 | 0.996 | 0.933 | 0.981 | 0.889 | 3.9m |
| Inception/Residual | 7 | 0.993 | 0.939 | 0.984 | **0.904** | 5.3m |
| Inception/Residual | 8 | 0.996 | 0.933 | 0.982 | 0.880 | 7m |
| Residual | 1 | 0.855 | 0.818 | 0.934 | 0.757 | 92k |
| Residual | 2 | 0.958 | 0.898 | 0.970 | 0.847 | 330k |
| Residual | 3 | 0.991 | 0.931 | 0.982 | **0.899** | 1.3m |
| Residual | 4 | 0.994 | 0.932 | 0.986 | 0.896 | 5m |
| Residual | 5 | 0.994 | **0.933** | 0.984 | 0.889 | 20m |
| VGGNet | 1 | 0.667 | 0.665 | 0.840 | 0.570 | 14k |
| VGGNet | 2 | 0.904 | 0.874 | 0.960 | 0.803 | 74k |
| VGGNet | 3 | 0.979 | 0.920 | 0.981 | 0.872 | 304k |
| VGGNet | 4 | 0.991 | **0.942** | 0.986 | **0.901** | 1.2m |
| VGGNet | 5 | 0.988 | 0.935 | 0.981 | 0.887 | 4.8m |

not contain these layers, they were clearly not required for our $64 \times 101$ plankton images.

### B. MLP GRID SEARCH

Two grid searches were constructed to learn a suitable MLP architecture from the geometric data (Fig. 8). The initial search consisted of learning the number of hidden layers in the network and the number of neurons within each layer. The simple nature of the input data meant that only a single hidden layer containing 512 neurons was required for the model to effectively learn representations for each class, while clearly as can be seen in Fig. 7, performance degrades as the model becomes too complex. The next grid search was used to determine a suitable parameter for the dropout value. We tested all values between 0.0 and 0.8 in 0.02 increments whereby 0.0 is effectively no dropout and 0.8 indicates 80 % of the neurons will be dropped. A dropout value of 0.62 provided a 0.906 validation accuracy and 0.836 F1 score. With a suitable model configuration and dropout value learned, we allowed the model to fully converge and a snapshot of the model's weights were stored.

### C. TESTING ConvNet COLLABORATIONS

The ConvNet scaling tests yielded four distinct model types each with unique characteristics. For each ConvNet model tested, a snapshot of the architecture and training weights was saved. We used the results in Table 2 to decide which model of each type to be included in the collaborative model. F1 score

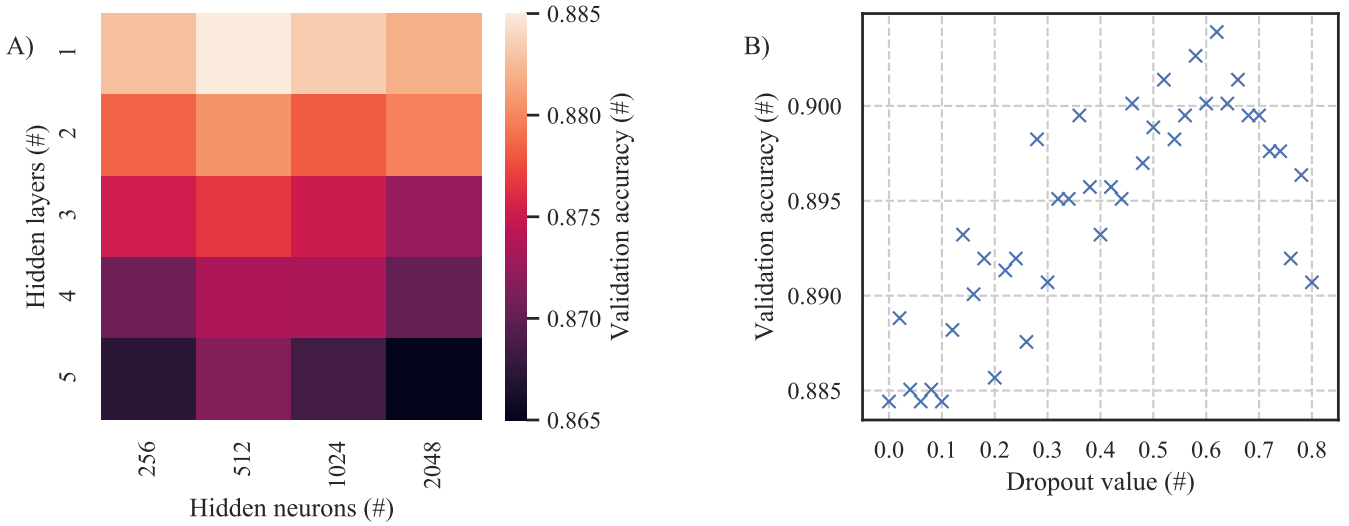


**FIGURE 8.** Results of two grid search experiments using the geometric data on Multi-layer Perceptron models. A) A grid search testing all combinations of number of hidden layers and number of hidden neurons within each layer. B) Using the top performing model from A, the effect of applying dropout before the last layer on validation accuracy is tested.

was the main metric used to decide the best model of each type. The models chosen to progress to collaborative model testing were DenseNet with two modules, Inception (residual connections) with seven modules, ResNet with three modules and VGGNet with four modules.

Although it would have been beneficial to test all combinations of ConvNets in the collaborative model, this was unfeasible due to computational limitations. Instead we initially constructed the collaboration model with all four ConvNets. Once the first training phase had completed, the weakest individual learner (based on F1 accuracy) was removed for the next test. This was repeated until there were only the two strongest ConvNets in the collaborative model.

Throughout, the collaborative models were able to outperform individual learners (Fig. 9). Noticeably, the inclusion of the weakest performing ConvNet (DenseNet with two modules) negatively impacted the F1 score when compared with the three model ConvNet collaboration. Eventually, the collaborative model containing the three strongest ConvNets offered a notable 3.9-point improvement in F1 score when compared with the top individual model (Table 3). With the overall accuracy improving by 2 points, this indicated that most of the improvement in the model came from improved accuracy within the minority classes.

### D. INCLUSION OF THE GEOMETRIC DATA

With a suitable collaborative configuration for ConvNets understood we included the MLP model that had previously been trained on the geometric data. Like the ConvNets, the MLP's trained weights were frozen before the model was loaded into the collaboration and the output SoftMax layer removed. We assessed two new collaborative models, one with the three ConvNets (Inception, Residual and VGGNet inspired), and another using a collaboration with only the VGGNet inspired to form a lightweight collaboration. After training for each had converged, the models were assessed using the holdout test set (Fig. 10, Table 3). While the 1.2 points classification accuracy over the ConvNet only collaboration was modest, the notable 4.3 points gain in F1 score demonstrated a significant accuracy improvement in minority classes. The lightweight VGGNet and MLP collaboration also provided improved results over the ConvNet only collaboration, suggesting that the uniqueness of each individual learner in the collaboration has the most effect on performance since all ConvNets outperformed the MLP on an individual basis.

## IV. DISCUSSION

Within a given aquatic environment, it is natural for some planktonic taxa to be numerically more dominant than others. When sampling, dominant taxa are imaged more frequently than rare and infrequently-encountered members of the community. This results in an imbalanced dataset, which reflects natural imbalances within the environment.

Dealing with imbalanced datasets is a long-standing research topic in machine learning [28], and within ConvNets [41]. Realistically, it is challenging to build a plankton dataset with ample particles for each taxa, especially when expert knowledge is required to label images. In this paper the plankton time series dataset inherited a severe imbalance, with almost half of the taxa represented by 50 or fewer sample images (Fig. 4). This creates two problems: firstly, a model's learning can easily be over-influenced by majority classes since they are presented more often. Secondly, in the minority classes there may not be enough data available for a model to gain a full understanding of rare taxa.

The class imbalance problem is an active and varied area of research, and several techniques to help improve

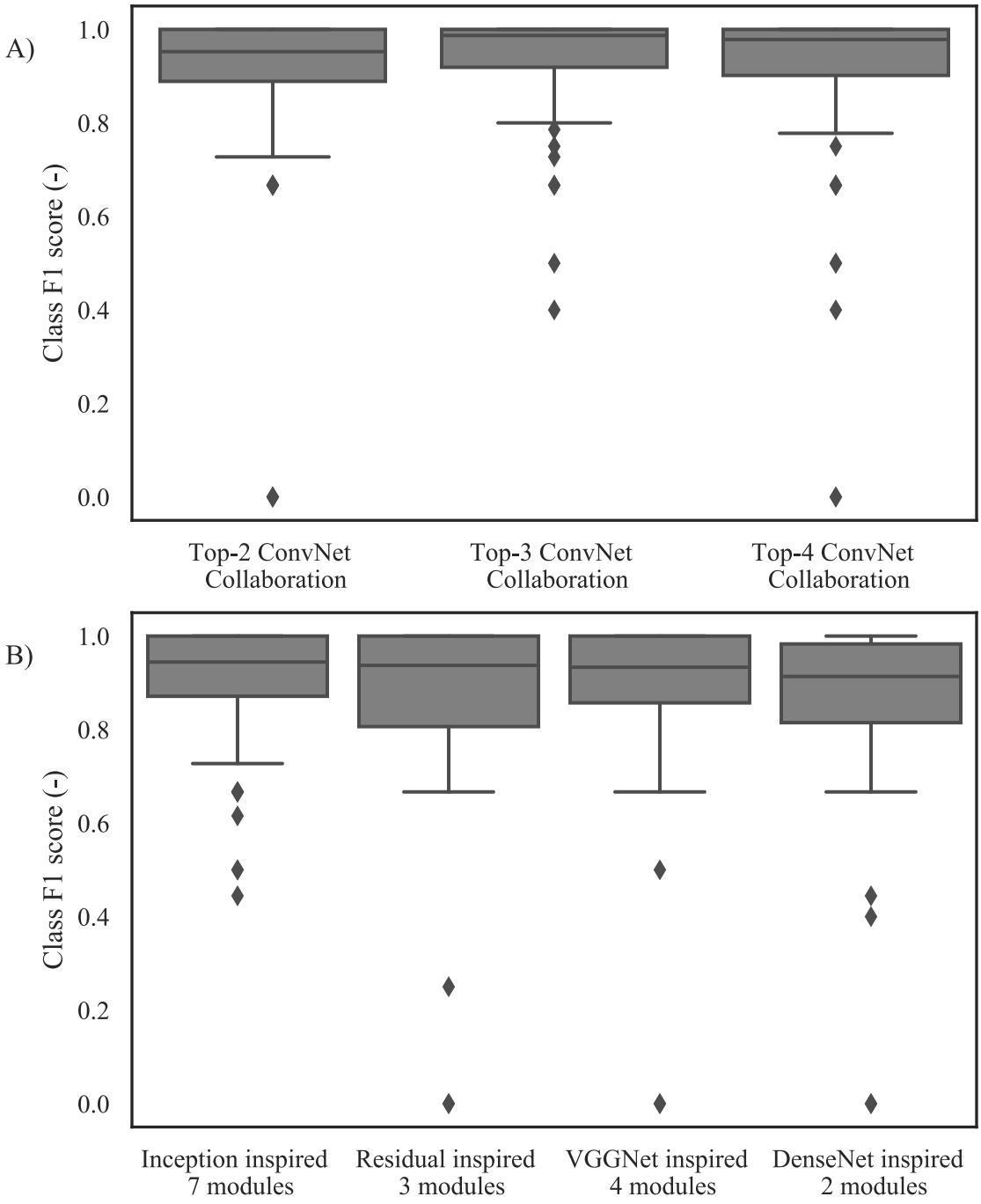


**FIGURE 9.** A comparison of collaborative ConvNet models against the top performing individual ConvNets for individual classes. The boxes represent the interquartile range (IQR) (25 % to 75 % percentile) of the results. The solid line within the IQR is the median result. 1.5 times the F1 score from the IQR boundaries are the whiskers. The diamonds represent outliers, defined as taxonomic groups with a low F1 score. A) Results of testing different ConvNet collaborations, performed by initially collaborating the top four ConvNet models, then removing the worse performing model for the next experiment. B) Performance of the four individual ConvNets chosen for the collaborative models.

classification accuracy for minority classes have been proposed. Some of these techniques operate at the algorithm level, including focal loss, which down weights loss values for common classes [45]. Other techniques operate at the data level, with a focus on boosting the amount of data associated with minority classes. One of these techniques is image augmentation, as used here. Another technique, called the Synthetic Minority Oversampling Technique (SMOTE), interlopes between minority classes in order to generate new artificial samples [46].

In this paper we applied a number of established techniques to deal with the issue of class imbalance, including

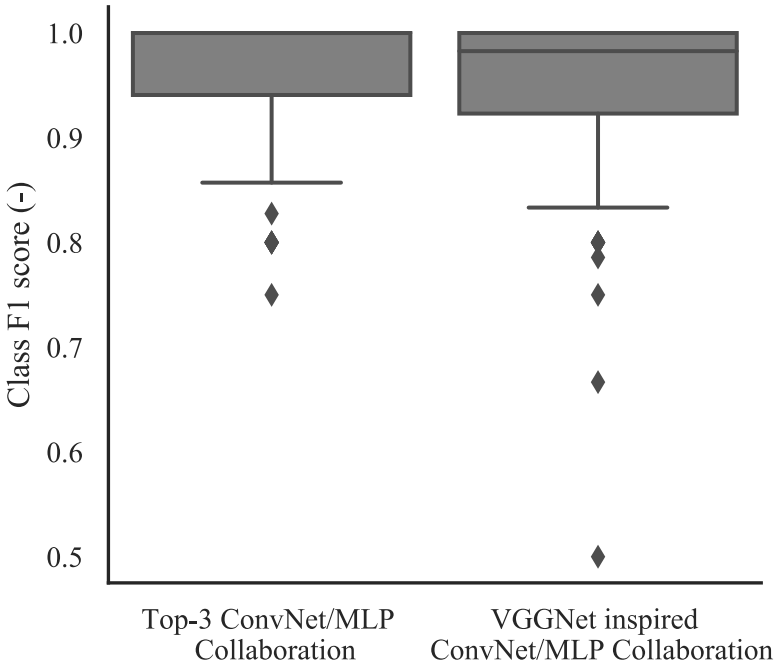


**FIGURE 10.** Comparison of F1 scores for the best performing collaborative and individual ConvNet models with the addition of MLP collaboration. The left-hand side plot shows results for the best performing top-3 ConvNet collaboration model, with MLP collaboration. The right-hand plot shows results for the best performing individual VGGNet inspired 4 module ConvNet with MLP collaboration.

**TABLE 3.** Comparing the performance of different collaborative models against the best individual ConvNet, MLP models and a typical ensemble where the outputs from each learner were averaged.

| Model | Top-1 accuracy | Top-3 accuracy | F1 accuracy | Kappa score | Number of parameters | Inference time per input (ms) |
|---|---|---|---|---|---|---|
| VGGNet 4 modules | 0.942 | 0.986 | 0.902 | 0.940 | 1.2M | 2 |
| MLP | 0.906 | 0.979 | 0.836 | 0.903 | 639k | < 1 |
| VGGNet & MLP collab | 0.966 | 0.995 | 0.946 | 0.964 | 2.3M | 2 |
| Inception/Res & Residual & VGGNet collab | 0.962 | 0.989 | 0.939 | 0.960 | 9.1M | 6 |
| Inception/Res & Residual & VGGNet & MLP ensemble | 0.963 | 0.992 | 0.941 | 0.962 | 8.7M | 4 |
| Inception/Res & Residual & VGGNet & MLP collab | **0.974** | **0.995** | **0.962** | **0.973** | 9.8M | 6 |

image augmentation in order to provide the ConvNets with additional insight into certain taxa; and class weights to help address the issue of certain classes being presented to a deep learning model more often than others. Despite this, individual ConvNets and the MLP still lacked understanding of certain classes. However, although individual models provided poor accuracy within certain minority classes this did not equate to the model having no understanding of these classes. We suggest that collaborative models can harness the limited understanding of each individual to allow it to form a collective, confident and more precise decision, as demonstrated here.

Our proposed collaborative model still showed it had issues with certain under-represented taxa. This can be seen when breaking down the performance between phytoplankton and zooplankton. With a larger majority of the zooplankton taxa belonging to minority classes when compared with phytoplankton, there was increased variability in F1 scores (Fig. 11) highlighting that our model showed some prediction weakness for certain classes.

In all tests, we found that collaborative models offered improved performance over individual learners. The top collaborative model consisting of three distinct ConvNets and an MLP gained a respectable 3.2-point improvement in overall accuracy against the best individual learner. However, of most significance is the 6-point improvement in F1 accuracy, leading to a 0.962 F1 score (Table 3). Meanwhile, a typical ensemble containing the same models was able to

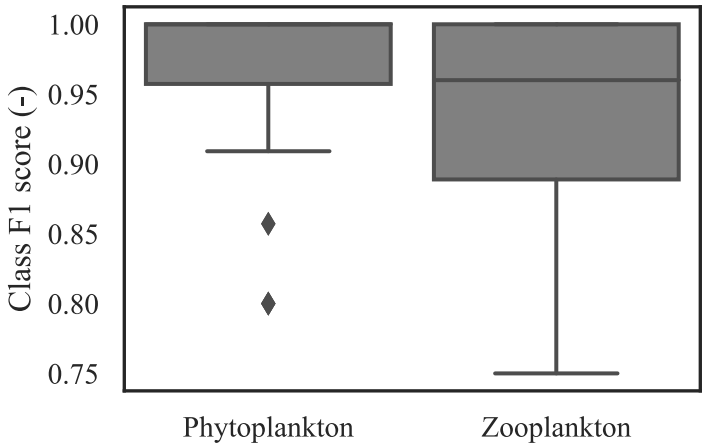


**FIGURE 11.** Using the full ConvNet with MLP collaborative model, a comparison is shown of the results on two taxonomic groups. The wider variation in the zooplankton results are a result of a large proportion of its training data belonging to minority classes.

reach 0.963 overall accuracy and 0.941 F1 score. Given the 2.1 F1 score improvement over the ensemble approach, this further demonstrates the effectiveness of teaching a model to collaborate through an additional learning phase. This result is also reflected in the kappa scores, which improved by 3.3 points when compared with the best individual learner. There was also a 1.3 point improvement with the inclusion of the MLP model. As such, this demonstrates that collaborative models are an effective technique that can be applied when dealing with imbalanced datasets.

We can reason this improvement from the uniqueness of each individual learner present in the collaborative model. The results of individual ConvNets were found to have unique strengths and weaknesses at predicting certain taxa. For example, the seven module Inception model (with residual connections) was the only individual model to correctly recall *Trichodesmium* yet offered poor performance on classes such as Ctenophore ctene or *Proboscia alata* compared to the other ConvNets. Examples such as these can be found throughout, suggesting that each distinct model has gained its own unique insight into the underlying structure of the dataset despite all posting similar overall accuracy scores. Therefore, by teaching these models to effectively work together, not only can these models combine their understanding, the trained model is able to harness the individual models that perform strongly for certain taxonomic categories.

We examined 40 classes that contained 50 or fewer particles and compared the prediction results of the full collaborative model against the VGGNet 4 module ConvNet. In 16 cases there was no improvement; of the remaining 24 classes we found 20 with improved F1 accuracy. The most striking results were for taxa where individual models struggled most (e.g. *Trichodesmium* and Trochophore, for which the collaborative model offered perfect accuracy). On investigation, of the 4 taxonomic groups where the model offered slightly worse results, we found that these were classes where results across the individual models were indifferent. With no stand out individual learner to harness, the collaborative model lacked confidence in its prediction.

Within the dataset, there were several classes that had similar morphological traits to other classes. These would usually require a human expert to differentiate them based on subtle differences in their features. In general the ConvNet's remarkable pattern recognition capability was able to separate such classes successfully. However, it is clear the collaborative model struggled with certain groups. The classes *Tripos* sp.1, *Tripos* sp.2 and *Tripos meulleri* (classes 3, 5 and 7 from Fig.3 respectively) are all morphologically similar. The Inception based ConvNet offered perfect accuracy on each of these classes while other models had a mixture of F1 scores ranging from 0.0 to 1.0. The full collaborative model (ConvNets and MLP) offered an F1 score of 0.857, 0.8 and 1.0 on the three classes respectively, showing a reduction in prediction quality. In this case, it would appear that weaker learners have introduced doubt into the model. In contrast, the morphologically similar classes *Proboscia alata* and *Rhizosolenia styliformis* (class 41 and 47 from Fig.3 respectively) showed clear improvement with the collaborative model. Individual models scored up to 0.89 and 0.93 F1 scores respectively while the collaborative model increased this to 1.0 and 0.94 respectively. This improvement was particularly noticeable with the MLP model present, indicating additional contextual insight was beneficial. A third group of morphologically similar classes were *Acartia* copepodite and *Oithona* copepodite. For these, the collaborative model was unable to improve on the best individual F1 scores of 1.0 and 0.8 respectively, suggesting no additional insight were discovered from any of the learners.

It is also worth noting that the individual learners included within a collaborative model need not have the strongest accuracy performance. Rather, how distinct their insight into the dataset is, seems to be the main factor governing

improvements in the collaborative model's prediction quality. This is supported by the inclusion of the MLP model trained on geometric data; despite posting poorer accuracy scores than any of the ConvNets, it was able to provide the most noticeable improvement. Meanwhile, the DenseNet inspired model posted higher accuracy scores than the MLP but was unable to bring any benefit to the collaboration, indicating that arbitrarily including models could potentially plateau or even offer poorer performance than collaborations with fewer models.

The ConvNet models are adept at locating and recognising abstract features, fine grain details and particle edges, allowing it to learn patterns within objects that provide it with the ability to make predictions. Despite this, a ConvNet has no understanding of the additional context associated with each image, such as geometric or seasonal data. Therefore, it is useful to provide the collaborative model with additional precise measurement information which allowed it to gain an even deeper understanding of the taxa. This final collaborative model has a 0.962 F1 score, 0.974 accuracy score and 0.973 kappa score on 104 classes.

Our method of including contextual data differed to that of Ellen *et al.* [30] in that we utilized the collaborative model to include this data. We noted improved gains by implementing the model in this way. For example, interacting the data in the last few fully connected layers in a network improved convergence time and accuracy. Due to the drastically differing learning rates of the ConvNet and MLP models we propose that for best accuracy results each individual model should be trained on the relevant datatype in isolation. This allows model training to converge without interference. For this reason, it is important that each trained individual learner should have its weights frozen before being loaded into a collaborative model as described in this paper, to remove the chance of certain datatypes or models distorting the training of a collaborative model. When training a collaborative model, the only concern should be how individual learners contribute to the final decision given a certain input, and not to learn new patterns in the dataset.

## V. CONCLUSION

Gathering data from most real-world settings will result in imbalanced datasets where observations in majority classes outnumber those in the minority classes. This is especially true when imaging planktonic particles in samples collected from natural aquatic environments. ConvNets have proven state-of-the-art performance at object detection and location yet become challenging to effectively train when presented with an imbalanced dataset [41]. In this paper we used common techniques such as image augmentation and class weights to combat this issue. However, the models still faced problems with certain minority classes. To help further address the problem, we have presented the application of collaborative deep learning models. The approach was shown to significantly improve prediction quality in minority classes.

The individual learners within a collaborative model should each contain distinct characteristics that will allow each one to gain a unique insight into the provided dataset. Therefore, we have also provided a suggested approach to construct and scale unique ConvNet architectures based on previous state-of-the-art designs that allow the approach to be implemented using only image data. We found that adding contextual data allows for a further improvement in accuracy, which mirrors the findings of Ellen *et al.* [30]. In the process, we have also demonstrated a data processing pipeline for FlowCam data that allows it to be converted into a format that can be used with machine learning models.

ConvNets are an active topic of research, with new state-of-the-art architectures being regularly released. EfficentNet [47] is a recent example that provides an efficient approach to scale ConvNets. Collaborative models can accept different types of deep learning model, and performance gains are realised as long as each individual learner gains a unique insight into the data. Using the scaling method proposed by the authors of EfficentNet to construct numerous unique ConvNets may offer improved performance over the scaling methods presented in this paper; in particular, by reducing the number of parameters in the model and improving inference times (Table 3).

The models presented here have been developed and tested using a single dataset from Station L4 in the Western English Channel. An obvious extension to the work is to test the same model with different datasets from around the world, such as the WHOI and Kaggle datasets. While many images will not be accompanied by the same context data that FlowCam generates, the collaborative ConvNet model can still be applied. However, due to the added performance achieved when using context data, there are clear advantages to using it where possible. It should also be noted that different areas of the world are characterised by different plankton communities, each adapted to local prevailing conditions; and it remains an open question regarding how portable a given system is. In this respect, having sufficient representative training data to help alleviate the class imbalance problem remains vital.

## CODE AVAILABILITY

The code used in the study can be downloaded from https://gitlab.ecosystem-modelling.pml.ac.uk/tk359/flowcamclassification.

## ACKNOWLEDGMENT

The authors wish to thank the officers and crew of Plymouth Marine Laboratory's research vessel Plymouth Quest for collecting the live plankton samples. The authors declare no conflicts of interest.

## REFERENCES

[1] A. Sournia, M.-J. Chrdtiennot-Dinet, and M. Ricard, "Marine phytoplankton: How many species in the world ocean?" *J. Plankton Res.*, vol. 13, no. 5, pp. 1093–1099, 1991.
[2] C. De Vargas *et al.*, "Eukaryotic plankton diversity in the sunlit ocean," *Science*, vol. 348, no. 6237, 2015, Art. no. 1261605.

[3] M. Benfield, P. Grosjean, P. Culverhouse, X. Irigolen, M. Sieracki, A. Lopez-Urrutia, H. Dam, Q. Hu, C. Davis, A. Hanson, C. Pilskaln, E. Riseman, H. Schulz, P. Utgoff, and G. Gorsky, "RAPID: Research on automated plankton identification," *Oceanography*, vol. 20, no. 2, pp. 172–187, Jun. 2007.

[4] P. B. Ortner, S. R. Cummings, R. P. Aftring, and H. E. Edgerton, "Silhouette photography of oceanic zooplankton," *Nature*, vol. 277, pp. 50–51, Jan. 1979.

[5] H. Jeffries, K. Sherman, R. Maurer, and C. Katsinis, "Computer-processing of zooplankton samples," in *Estuarine Perspectives*. New York, NY, USA: Academic, 1980, pp. 303–316.

[6] M. Rolke and J. Lenz, "Size structure analysis of zooplankton samples by means of an automated image analyzing system," *J. Plankton Res.*, vol. 6, no. 4, pp. 637–645, 1984.

[7] C. S. Davis, S. M. Gallager, M. S. Berman, L. R. Haury, and J. R. Strickler, "The video plankton recorder (VPR): Design and initial results," *Arch. Hydrobiol. Beih. Ergebn. Limnol.*, vol. 36, pp. 67–81.

[8] S. Samson, T. Hopkins, A. Remsen, L. Langebrake, T. Sutton, and J. Patten, "A system for high-resolution zooplankton imaging," *IEEE J. Ocean. Eng.*, vol. 26, no. 4, pp. 671–676, Oct. 2001.

[9] R. J. Olson and H. M. Sosik, "A submersible imaging-in-flow instrument to analyze nano-and microplankton: Imaging FlowCytobot," *Limnol. Oceanogr., Methods*, vol. 5, no. 6, pp. 195–203, Jun. 2007.

[10] C. Sieracki, M. Sieracki, and C. Yentsch, "An imaging-in-flow system for automated analysis of marine microplankton," *Mar. Ecol. Prog. Ser.*, vol. 168, pp. 285–296, Jul. 1998.

[11] P. Grosjean, M. Picheral, C. Warembourg, and G. Gorsky, "Enumeration, measurement, and identification of net zooplankton samples using the ZOOSCAN digital imaging system," *ICES J. Mar. Sci.*, vol. 61, no. 4, pp. 518–525, Jan. 2004.

[12] J. L. Bell and R. R. Hopcroft, "Assessment of ZooImage as a tool for the classification of zooplankton," *J. Plankton Res.*, vol. 30, no. 12, pp. 1351–1367, Dec. 2008.

[13] H. Lee, M. Park, and J. Kim, "Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3713–3717.

[14] C. Wang, X. Zheng, C. Guo, Z. Yu, J. Yu, H. Zheng, and B. Zheng, "Transferred parallel convolutional neural network for large imbalanced plankton database classification," in *Proc. MTS/IEEE Kobe Techno-Oceans (OTO)*, May 2018, pp. 1–5.

[15] H. P. Jeffries, M. S. Berman, A. D. Poularikas, C. Katsinis, I. Melas, K. Sherman, and L. Bivins, "Automated sizing, counting and identification of zooplankton by pattern recognition," *Mar. Biol.*, vol. 78, no. 3, pp. 329–334, 1984.

[16] X. Tang, W. K. Stewart, L. Vincent, H. Huang, M. Marra, S. M. Gallager, and C. S. Davis, "Automatic plankton image recognition," *Artif. Intell. Rev.*, vol. 12, pp. 177–199, Feb. 1998.

[17] M. B. Blaschko, G. Holness, M. A. Mattar, D. Lisin, P. E. Utgoff, A. R. Hanson, H. Schultz, and E. M. Riseman, "Automatic *in situ* identification of plankton," in *Proc. 7th IEEE Workshops Appl. Comput. Vis. (WACV/MOTION)*, Jan. 2005, pp. 79–86.

[18] A. Gislason and T. Silva, "Comparison between automated analysis of zooplankton using ZooImage and traditional methodology," *J. Plankton Res.*, vol. 31, no. 12, pp. 1505–1516, Dec. 2009.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–14.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[24] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.

[25] E. C. Orenstein, O. Beijbom, E. E. Peacock, and H. M. Sosik, "WHOI-plankton—A large scale fine grained visual recognition benchmark dataset for plankton classification," 2015, *arXiv:1510.00745*. [Online]. Available: http://arxiv.org/abs/1510.00745

[26] R. K. Cowen, S. Sponaugle, K. L. Robinson, J. Luo, Oregon State University, and Hatfield Marine Science Center, "Plankton-Set 1.0: Plankton imagery data collected from F.G. Walton Smith in Straits of Florida from 2014-06-03 to 2014-06-06 and used in the 2015 National Data Science Bowl (NCEI Accession 0127422)," Dataset, NOAA Nat. Centers Environ. Inf., Asheville, NC, USA, Tech. Rep., 2015. [Online]. Available: https://data.nodc.noaa.gov/cgi-bin/iso?id=gov.noaa.nodc:0127422, doi: 10.7289/v5d21vjd.

[27] C. Cardie and N. Howe, "Improving minority class prediction using case-specific feature weights," in *Proc. 14th Int. Conf. Mach. Learn.*, 1997, pp. 57–65.

[28] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Syst. Appl.*, vol. 73, pp. 220–239, May 2017.

[29] J. Dai, R. Wang, H. Zheng, G. Ji, and X. Qiao, "ZooplanktoNet: Deep convolutional network for zooplankton classification," in *Proc. OCEANS*, Apr. 2016, pp. 1–6.

[30] J. S. Ellen, C. A. Graff, and M. D. Ohman, "Improving plankton image classification using context metadata," *Limnol. Oceanogr., Methods*, vol. 17, no. 8, pp. 439–461, 2019.

[31] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get M for free," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–14.

[32] T. Smyth, A. Atkinson, S. Widdicombe, M. Frost, I. Allen, J. Fishwick, A. Queiros, D. Sims, and M. Barange, "The western channel observatory," *Prog. Oceanogr.*, vol. 137, pp. 335–341, Sep. 2015.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[35] E. Bochinski, G. Bacha, V. Eiselein, T. J. Walles, J. C. Nejstgaard, and T. Sikora, "Deep active learning for in situ plankton classification," in *Proc. Int. Conf. Pattern Recognit.*, in Lecture Notes in Computer Science, Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, 2019, pp. 5–15.

[36] O. N. Keene, "The log transformation is special," *Statist. Med.*, vol. 14, no. 8, pp. 811–819, Apr. 1995.

[37] K. V. Embleton, "Automated counting of phytoplankton by pattern recognition: A comparison with a manual counting method," *J. Plankton Res.*, vol. 25, no. 6, pp. 669–681, Jun. 2003.

[38] H. M. Sosik and R. J. Olson, "Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry," *Limnol. Oceanogr., Methods*, vol. 5, no. 6, pp. 204–216, Jun. 2007.

[39] F. Zhao, F. Lin, and H. S. Seah, "Binary SIPPER plankton image classification using random subspace," *Neurocomputing*, vol. 73, nos. 10–12, pp. 1853–1860, Jun. 2010.

[40] N. Pinto, D. D. Cox, and J. J. DiCarlo, "Why is real-world visual object recognition hard?" *PLoS Comput. Biol.*, vol. 4, no. 1, p. e27, Jan. 2008.

[41] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Netw.*, vol. 106, pp. 249–259, Oct. 2018.

[42] S. Lawrence, I. Burns, A. Back, A. C. Tsoi, and C. L. Giles, "Neural network classification and prior class probabilities," in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 1998.

[43] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[44] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.

[45] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 2980–2988, Feb. 2018.

[46] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Proc. Int. Conf. Intell. Comput.* Berlin, Germany: Springer, 2005, pp. 878–887.

[47] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*. [Online]. Available: http://arxiv.org/abs/1905.11946

**THOMAS KERR** received the B.Sc. degree in computer science from the University of Plymouth, U.K., in 2016, and the M.Sc. degree in data science from the University of Exeter, U.K., in 2019, where he will shortly be pursuing the Ph.D. degree in partnership with Plymouth Marine Laboratory, U.K. His current research interests include data analytics, data processing, and deep learning.

**JAMES R. CLARK** received the M.Phys. degree in physics with astrophysics from the University of York, in 2003, and the Ph.D. degree in environmental science from the University of East Anglia, in 2010. He is currently a Senior Scientist at the Plymouth Marine Laboratory, where he has been working since 2013. He is also an Honorary Researcher at the University of Exeter and an Honorary Lecturer at the University of East Anglia. His research interests include marine ecosystem modelling, marine ecology and biogeochemistry, and marine pollution.

**ELAINE S. FILEMAN** received the B.Sc. degree (Hons.) in biological sciences and the M.Phil. degree from the University of Southampton, in 1987 and 2001, respectively. She has been a Plankton Ecologist at the Plymouth Marine Laboratory since 1989. She is specifically interested in the ecology of microzooplankton and their trophic interactions within the marine food web. Her research activities include the automated imaging of plankton using FlowCam.

**CLAIRE E. WIDDICOMBE** received the M.Sc. degree in applied marine science from Plymouth University, in 1992. She has been working at the Plymouth Marine Laboratory as a Plankton Ecologist since 1993. Her research and expertise focus on phytoplankton community structure and diversity. She leads the Western Channel Observatory's Station L4 phytoplankton long-term time-series, which started, in 1992. She is also a member of the ICES Working Group on Phytoplankton and Microbial Ecology (WGPME) and the International Group for Marine Ecological Time Series (IGMETS).

**NICOLAS PUGEAULT** (Member, IEEE) received the Ph.D. degree from the University of Goettingen, Germany, in 2008. He was at the University of Edinburgh, University of Southern Denmark, University of Surrey, and University of Exeter. He has been awarded a Fellowship (2018–2020) by the Alan Turing Institute. He is currently a Reader in Computer Vision and Machine Learning at the University of Glasgow. His researches focus on aspects of machine learning, computer vision, and intelligent robotics.