Ecole Polytechnique

Research Internship Report - 3A

MAP-594

Computer Vision for plankton recognition

Author:

Jean-Baptiste Himbert
jb.himbert@gmail.com

July 16, 2023

Anti-Plagiarism statement

I declare on my honour that what has been written in this work has been written by me and that, with the exception of quotations and referenced work, no part has been copied from scientific publications, the Internet or from research works already presented in the academic field by me or by other students.

Déclaration d'intégrité relative au plagiat

Je déclare sur l'honneur que ce qui a été écrit dans le présent travail l'a été par moi-même et que, à l'exception des sources et travaux référencés, aucune partie n'a été copiée, que ce soit de publications scientifiques, d'Internet ou d'autres travaux de recherche déjà présentés par moi-même ou par d'autres étudiants.

Abstract

Recent advancements in deep learning computer vision methods, combined with technological hardware innovations, have enhanced automation of plankton image classification as traditionally carried out by marine biologists and taxonomists. Computer vision methods have come a long way since hand-engineered feature extraction classifiers, and are now primarily based on Convolutional neural networks ("CNNs") and Vision transformer ("ViT") models which are now dominating research in the space. In this paper, we test several CNNs and ViT models on two raw datasets of plankton images, with an aim to establish a baseline accuracy using deep learning models. Our differentiated approach is to keep the original raw datasets completely unaltered, to develop a baseline classification on unbalanced data using both CNNs and ViT models, and refine techniques to overcome class imbalances.

Les récentes avancées en *computer vision*, et particulièrement en *deep learning*, ont révolutionné la classification automatique d'images biologiques. Cela s'applique notamment aux images de plancton, dont la classification est traditionnellement effectuée par les biologistes et taxonomistes marins. Les méthodes de vision assistée par ordinateur, anciennement basées sur des modèles d'aggrégation de caractéristiques conçus manuellement, sont aujourd'hui dominées par les réseaux neuronaux convolutifs ("CNNs") ou les modèles *Vision transformer* ("ViT"). Dans ce papier, nous testons plusieurs CNNs et ViTs sur deux ensembles de données d'images de plancton, afin d'établir une base de résultats à l'aide de modèles d'apprentissage profond (*deep learning*). Notre approche se distingue des travaux précédemment effectués sur le sujet en cela que nous ne modifions pas les ensembles de données brutes, afin de développer une classification de base à partir de données originales fortement déséquilibrées. Nous utilisons différentes architectures reconnues et nous nous efforçons d'affiner plusieurs techniques spécifiques aux problèmes déséquilibrés.

Personal Acknowledgement

First and foremost, I would like to thank all the individuals without whom I could not have undertaken this fantastic internship at Inria Chile and the related work. In particular, words cannot express my deepest gratitude to Andrew Berry and Luis Marti from Inria Chile, for their invaluable patience, feedback and mentorship throughout the duration of my internship. This journey would not have been possible without the support of Nayat Sanchez-Pi, director of Inria Chile, who granted me access to the impressive knowledge and expertise from all other researchers, especially Luis Valenzuela, Hernan Lira and Hugo Carrillo who accepted to share with me their inspiring experiences and perspectives. Thank you as well to the other employees of the firm, especially Julia Allirot and Eliana Ruiz Sanchez, for creating so many happy memories during my time in Santiago. I am also grateful to my friends, especially Jean Dimier, Gabriel Buffet and Félix des Rotours, for their fruitful debates, and to my parents and sisters for their moral support throughout this journey.

1 Introduction

1.1 Motivation for Plankton Recognition

Understanding how climate change affects our planet in its entirety is one of the key scientific problems of our times. One approach in this quest is to observe the oceans, which account for 71% of the Earth's surface and represent an important bellwether for the environment. The majority of the oceanic biomass is made of plankton[2], a generic term encompassing millions of drifting aquatic micro-organisms, both in plant ("phytoplankton") and animal ("zooplankton") form. An organism is considered plankton if it cannot swim against tides and current. Phytoplankton in particular is an important contributor to our biosphere, as it constitutes the main source of food for organisms in the ocean, including zooplankton, and forms the very base of the marine food chain.

Plankton is not only key to marine ecosystems; it also plays a crucial role for our environment as a whole. In fact, the presence of phytoplankton in the oceans directly impacts the composition of gases in the atmosphere. Like most ground plants, ocean phytoplankton lives through photosynthesis, a biological process through sunlight energy and carbon dioxide are converted into organic compounds, with oxygen released as a byproduct. As a result, phytoplankton is estimated to produce up to fifty percent of the world's oxygen, and to cycle similar amounts of carbon dioxide from the atmosphere into the biosphere, thus participating in the storage of greenhouse gases[35].

Whilst phytoplankton contributes to air quality and helps mitigating climate change, the frenetic pace of global warming is not without consequences for these micro-organisms. As a matter of fact, greenhouse gas emissions coupled with the increase of ocean temperatures have led to more acidic waters, which may be detrimental to marine life. For example, acidity has a drastic impact on zooplankton exoskeleton[29], thereby threatening their development and ultimately impacting other marine animals down the food chain. Rising sea levels and temperatures have also modified the distribution of plankton throughout the oceans, leading to the formation of dense biomasses of phytoplankton blooms which negatively impact local ecosystems, starting from zooplankton populations themselves[44]. Indeed, the distribution of organisms in the oceans is heavily correlated to the intensity of ocean mixing affects light levels, nutrient recycling from deep layers and surface temperatures.

As a result, plankton is both an actor and an indicator of climate change. The close interrelations between plankton populations and our environment have fostered deep interest from the research community. "As the fundamental component of marine ecosystems, plankton is very sensitive to environment changes, and the study of plankton abundance and distribution is crucial, in order to understand environment changes and protect marine ecosystems." [48] As such, monitoring plankton is key to the understanding of marine ecosystems, and tracking plankton populations in real time provides tangible elements on the magnitude and consequences of climate change.

In this paper, we focus our analysis on the recognition of zooplankton species. As their role in the ocean food chain puts them at the crossroads between phytoplankton communities and larger ocean species, such as small fish or whales[21], their understanding will provide tangible elements for the study of other type of marine life.

Traditional methods for surveying zooplankton are either based on remote sensing via satellite imagery, on samples extraction from large nets dragged by sea vessels or, more recently, on autonomous underwater vehicles capturing plankton images *in situ*[10]. Despite these advances, there remains a bottleneck in the identification and verification of plankton species from oceanophotography, as marine biologists are required to annotate each image manually. In this context, deep learning computer vision models have been developed in an attempt to (semi-)automate the process of identifying, analysing and classifying features associated to these micro-organisms.

1.2 Foundations of Image Classification

Deep machine learning models have proved highly successful in completing a wide range of image-related tasks, and are now ubiquitous amongst computer vision benchmarks. Image classification is a type of computer vision task in which models automatically learn representations from images, assign them specific labels or categories based on observed features, and accurately classify them into predefined classes. Industrial applications are numerous, ranging from satellite imagery, medical diagnosis, self-driving cars, or in our case, automatic plankton recognition.

Convolutional neural networks ("CNNs") and Vision transformers ("ViTs" or "transformers") are two

prominent architectural approaches typically used for image classification, each having its own characteristics.

- CNNs leverage the concept of convolution, a mathematical operation often used in probabilities and signal processing. More practically, it involves sliding small filters across an image and performing element-wise multiplications and summations to extract spatial hierarchies of features, thus allowing to identify complex visual patterns. CNNs have achieved great success in image classification problems, and at the time of this paper, they are the most widely used approach in the field[1].
- Transformer models, on the other hand, were initially developed for Natural language processing ("NLP") [43]. Thanks to remarkable results in the field[2], the original *Transformer* architecture was adapted in 2020[7] to suit computer vision applications, with the development of ViTs. ViTs perform image analyses by dividing them into small patches and capturing their dependencies through multiple self-attention mechanisms, thus aggregating, in the end, the images' main characteristic patterns. ViT's attention mechanism is considered a *global* operation, as opposed to the intrinsically *local* operation performed by CNNs. This provides a strong advantage to ViTs, which tend to outperform CNNs in most of the traditional benchmarks in object detection[15].

For the purpose of our analysis, we chose to use state-of-the-art ("SOTA") models based on both types of architectures, namely the *Swin* transformer [24], Data-efficient image transformer ("DeiT") [41], *Inception* CNN [39], *EfficientNetV2* CNN [40], *ConvNeXt* CNN [25] and *Xception* CNN [4] models.

1.3 Plankton Datasets

There exist many annotated datasets of animal species covering plankton. For the most part, our work is based on two plankton-only datasets: *Zooscan* [9] and *Woods Hole Oceanographic Institution ("WHOI")* [30]. We also considered two smaller and more balanced datasets (*ZooLake* and *LensLess*), but concluded that they were less representative of real-life data, being small in size and lacking the long-tail distribution usually found in large-scale plankton expeditions. We thus choose to focus on the two aforementioned sets.

The *Zooscan* dataset is comprised of 1,433,278 images from 93 different taxa. It includes a taxonomic tree representing the relationships between plankton species, which will prove particularly useful in section 4 of our work. The *WHOI* dataset is even larger and contains 3,563,596 images of 103 taxa. Both datasets have been traditionally used as benchmarks for plankton classification, although often only a fraction is actually processed [19] [6] given their largely unbalanced nature. In fact, their respective imbalances factors[3] are 13,120 and 650,000, which is far superior to common benchmarks of long-tail distribution classification [47] such as *ImageNet-LT* [26] (imbalance factor of 256) and *iNaturalist2017* [14] (imbalance factor of 500). In spite of choosing the most complete datasets, we are facing a strong imbalance in class sizes, which poses a challenge for the application of traditional machine learning and deep learning frameworks [1].

2 Related Work

2.1 Plankton Classification

There is already a significant amount of research in the literature aiming at classifying plankton images[4]. One of the first papers on the topic was written in 1980 by Jeffries et al.[16], who focused on differentiating

[1]The *AlexNet* model developed in 2012 was a first breakthrough in the field, beating by a substantial margin every other model in the *ImageNet* competition. Since then, CNNs have held a monopoly on image-related tasks, including image classification. From hand-writing recognition for Optical character recognition ("OCR") technology, to the new smartphone picture modification software, or in other fields like audio processing or text analysis, CNNs have conquered a wide scope of applications.

[2]Large language models based on the transformer architecture have transformed the world of NLP, outperforming previous SOTA Recurrent neural networks ("RNN") baselines in language translation, emotion detection, text analysis, automatic document tagging, etc. *ChatGPT* and *Google Bard* are renowned examples of transformer models.

[3]The imbalance factor of a class distribution corresponds the ratio between the sizes of the largest and smallest classes. It is an indicator of skewness. While other metrics may provide a more complete information on the imbalance of a distribution, the imbalance factor is widely used for its simplicity.

[4]Driven by data availability (as a by-product of marine biologists work) and quality (as plankton images produced in labs), as well as genuine interest in the field from active research centres and associations such as *TaraOceans*, a partner of Inria Chile.

large plankton families thanks to geometric features of microscopic images. Along the years, researchers have applied a variety of machine learning methods for automatic plankton recognition, such as Linear discriminant analyses ("LDAs"), Artifical neural networks ("ANNs"), Support vector machines ("SVMs"), Random forest ("RF"), etc.[13][12][38]. Historically, research was based on hand extracted image features, and on very small samples. In years 2015 however, the deep learning revolution finally reached the plankton recognition world, with researchers successfully leveraging CNNs [6][27] on a combination of raw images and hand-crafted features. More recent work even leveraged the ViT architecture[19] to run ensembles of DeiTs on various ecological classification benchmarks, including plankton images datasets.

These developments quickly led to the creation of zooplankton benchmarks. Researchers then focused on aggregating large samples to complete datasets and enhancing classification methods and strategies. A few examples include *ZooLake*[20], *LensLess*[31], the *Kaggle* plankton dataset[6], *ZooScan*[9] and *WHOI*[30]. Our rationale for using the two latter in our work was explained in section 1.3.

One element to note about available research in the field, including aforementioned papers, is that it is generally based on balanced or almost balanced subsets of the *Zooscan* and *WHOI* datasets. Whilst using subsets allows for easier comparisons and computations, we believe that this does not reflect the unbalanced nature of plankton observed in oceans. Given that researchers and marine biologists aim to ultimately embark automatic recognition models in actual plankton expeditions, we have decided to work on the entirety of the *Zooscan* and *WHOI* datasets, including their inherently unbalanced data. We will then be able to assess the feasibility of such endeavour and, if applicable, to establish strong baselines for any future classification work in the field.

2.2 Classification under Long-Tail Distribution

Long-tail distribution problems ("LT") encompass a field of machine learning in which researchers focus on datasets and classification problems having very different class sizes. This if often the case in scientific research problems applied to biology, as not all objects or classes occur with similar frequencies in real life. Skewed distributions particularly prevail in ecological classification, given the difficulty to observe species having different living environments, behaviours or population sizes.

Looking at our own plankton classification problem, we notice that more than half of the images in our two datasets are labelled as *detritus*, and for some plankton species only a limited number of samples is available. This is a strong challenge in itself. To overcome such imbalances, we assess different methods traditionally deemed effective to tackle LT classification problems [47]: oversampling and undersampling, image augmentation and weighted penalisation.

2.2.1 Oversampling and Undersampling

One classic way to overcome imbalances is to reintroduce a degree of balance via oversampling minority classes, or undersampling head classes. This technique *de facto* harmonises class sizes. Various methods have been developed to better pick which images will be over or under sampled. For example, one is to assign scores to majority samples in order to keep only the most representative instances of a particular class [22]. Over and under sampling are efficient and easy to implement. However, undersampling is often conducted at the expense of using the full dataset. Given our objective to establish a baseline without sacrificing part of the data, we stayed away from undersampling strategies, and only used oversampling of the minority classes. However, in severe imbalanced situations, oversampling often runs the risk of overfitting on the minority samples. This is why such strategy is best used in combination with other techniques, as further explained in sections 2.2.2 and 2.2.3.

2.2.2 Image augmentation

Another important technique in computer vision is image augmentation, which corresponds to slight transformations made to sample images in the aim of increasing model robustness. This is a critical strategy for most image recognition problems - for example, a model classifying dogs and cats must be able to recognise a cat even if the picture is upside-down (see Figure 1 for augmentation examples).

In the LT context, we can leverage image augmentations as a way to create new samples from the minority classes (thus effectively oversampling minority classes) using a single base image. By doing so, the risk of overfitting remains limited as newly generated images are different.

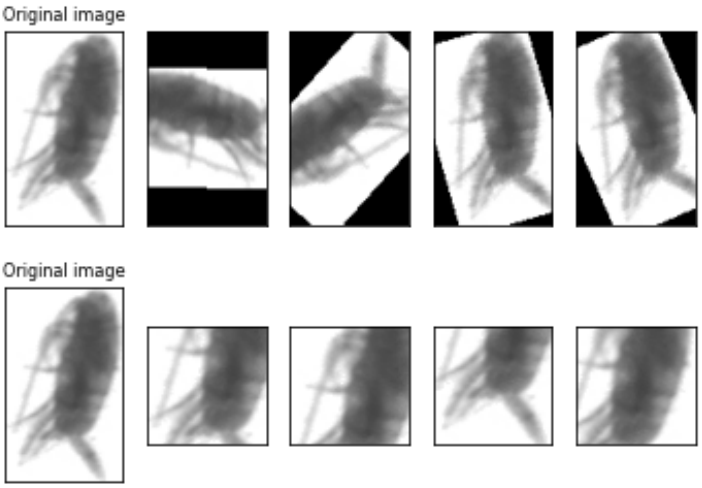


Figure 1: Example of two augmentation transformations: rotation and cropping

On top of the most common augmentation techniques used in computer vision - mainly a combination of rotations, cropping and gaussian blur -, newly developed augmentations have proven very efficient in classification situations. Among them, we can cite *RandomAugmentation*, which applies transformations in random order and magnitude, and *AutoAugmentation*, which selects the most adapted transformations to a classification problem [5].

2.2.3 Weighted penalisation

Last but not least, another idea for taming the LT classification problem is to implement a loss penalising more errors on minority samples than those on majority samples. The most common one is a focal loss [23], by which the final loss corresponds to the weighted sum of every class loss, taking into account model confidence, and pondered by the inverse weight of class sizes. Weighted penalisation forces the model to focus on difficult samples and minority classes, thus improving its accuracy on these classes. However, the use of weighted penalisation often decreases overall accuracy, due the inherently imbalanced way accuracy is calculated[5].

In deep learning, loss functions are at the core of the learning process. Their role is to evaluate the distance (or difference) between the model predictions and its targets. As the model aims to minimise the loss function, its parameters are iteratively adapted through backpropagation. In this paper, we assess two different types of loss functions often used in supervised classification tasks:

- Cross-entropy: in a N-class classification setting where each sample has a one-hot encoded target vector $t = [t_1, \ldots, t_N]$ and a prediction vector $c = [c_1, \ldots, c_N]$, the loss function for a sample from class m is:

$$L_{CE} = -t_m \log(p_m) - \sum_{i=1, i \neq m}^{N} (1 - t_i) \log(1 - p_i)$$

  where p_i is the Softmax probability for the i^{th} class.

- Focal loss: with the same notation, focal loss is a weighted modification of Cross-entropy, taking into account the relative size of each class $\alpha = [\alpha_1, \ldots, \alpha_N]$ and a γ parameter to create the following loss:

$$L_{FL} = -t_m \alpha_m p_m^{\gamma} \log(p_m) - \sum_{i=1, i \neq m}^{N} \alpha_i (1 - t_i)(1 - p_i)^{\gamma} \log(1 - p_i)$$

[5]This is a well-known fact about machine learning models on imbalanced datasets. When tackling imbalances, one should not assess a model looking at its accuracy only - indeed, as a classifier returning always 1 on a binary classification problem could achieve 90% accuracy if the dataset has a 90/10 imbalance. As larger classes have a larger impact on final accuracy, an important increase in tail classes combined with a small decrease in performance on the majority class would bring model accuracy down. Hence the importance of also calculating Precision, Recall and F1Score when handling skewed distributions.

2.3 Hierarchical Classification

Aside from the LT distribution problem explained above, plankton classification can also be formulated as a hierarchical classification problem. We take this approach in our work carried out for section 4.

In machine learning, hierarchical classification corresponds to a classification problem based on data structured as a direct acyclic graph ("DAG") or more often, as a tree. Such a structure features hierarchies between classes, by discriminating between parent and children nodes (see Figure 2).

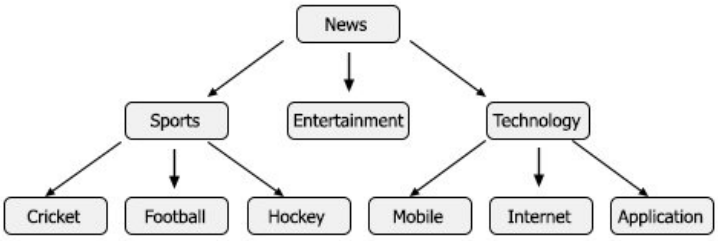


Figure 2: Example of a hierarchical classification problem, with multiple nodes (classes) distributed on three different layers

2.3.1 Application fields

Hierarchical classification is a very common framework. In fact, real-life data often comes structured as a tree, as used for instance in large-scale text classification [28], in e-commerce [11], in sound classification [17], in ecological classification [8], or more generally to tackle few-shot learning in many-classes datasets [45].

Hierarchical problems can even be extended outside of the supervised world to unsupervised or semi-supervised problems [33], by leveraging the metadata tree structure of samples to better analyse unlabelled data.

2.3.2 Hierarchical models

Hierarchical models have been a very active field of research between 2002 and 2011 [37]. Their study however lapsed with significant progress being made by deep neural networks. Hierarchical classification regained traction around 2017 as researchers realised that increasing the size of deep learning models was not enough to tackle difficult classification problems. Leveraging the underlying hierarchical structure of classes seemed an interesting approach to improve performance.

There exists a variety of techniques for hierarchical classification, but the most common one is made through a custom model taking the same structure as the hierarchy. In computer vision, this was first implemented by Xinqi Zhu and Michael Bain [49], who used a CNN-like backbone feeding into branches representing the different levels of depth in the tree (see Figure 3). All branches are trained simultaneously, with a weighted sum of every branch cross-entropy loss as a final loss function.

Following the path initiated by [49], researchers successfully tweaked the B-CNN model to better take into account the relationships between labels. We can cite the works of Zhang and al. [46] who implemented top-down residuals from brand to branch with a model called hierarchical bilinear convolutional neural network ("HB-CNN"), and of Kolisnik and al.[18] who implemented condition-based relationships between classes.

One recent work by Pizarro and al. [32] investigates an innovative way of leveraging class hierarchies by imposing a soft-constraint hierarchy through a combination of branch residuals and attention weighting (see following paragraph for more details on attention) (Figure 4). This mechanism fulfils two objectives: it makes relationships between labels more fluid (being softly linked rather than hardcoded), and it enables the flow of information to move both top-down and bottom-up. Thanks to these two improvements, the architecture is able to output more coherent hierarchies, and outperforms other branched networks (between other B-CNN [49] and HB-CNN [46]) on classic baselines, namely *CIFAR10*, *CIFAR100* and *Fashion MNIST*.

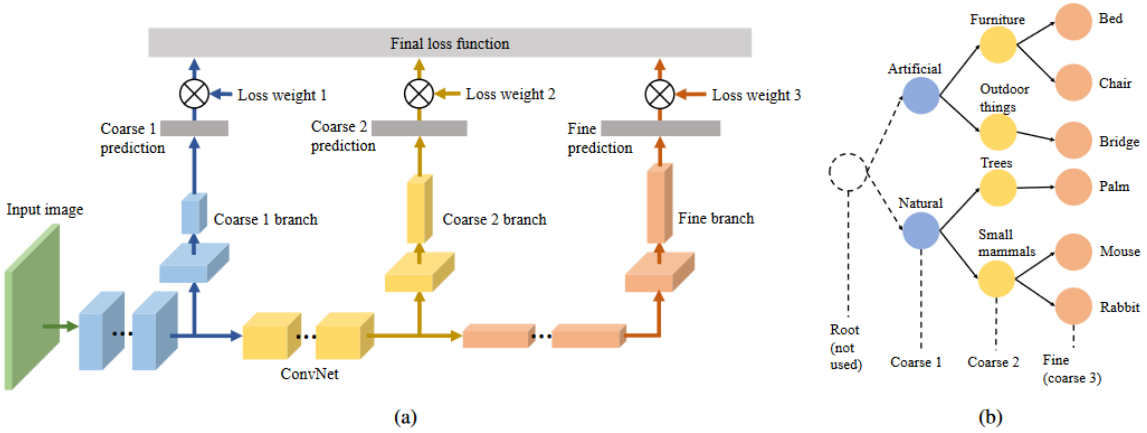


Figure 3: (a) Architecture of a branch convolutional neural network ("B-CNN"). (b) Sample hierarchical label tree where classes are taken from *CIFAR-100* dataset *(Image taken from [49])*

Attention: It is a mechanism inspired by the human capacity to focus on a selective part of the information only when facing a large set of data. It allows a specific branch to receive outputs from all other branches as a weighted sum. Coefficients of the weighted sum being trainable, the model tweaks them to extract information by giving more importance to relevant neighbour classes of the considered class. Formally, given a list of vector inputs $I = \{\mathbf{i}_0, \mathbf{i}_1, ..., \mathbf{i}_N\}$ corresponding here to the outputs of the feature extractor of every branch, the attention mechanism computes a new vector $\mathbf{a}$ as follows:

$$\mathbf{a} = \sum_{n=0}^{N} \alpha_n(\mathbf{i}_0, \mathbf{i}_1, ..., \mathbf{i}_N)\mathbf{i}_n$$

where the α is a learned function, often via a small fully-connected model.

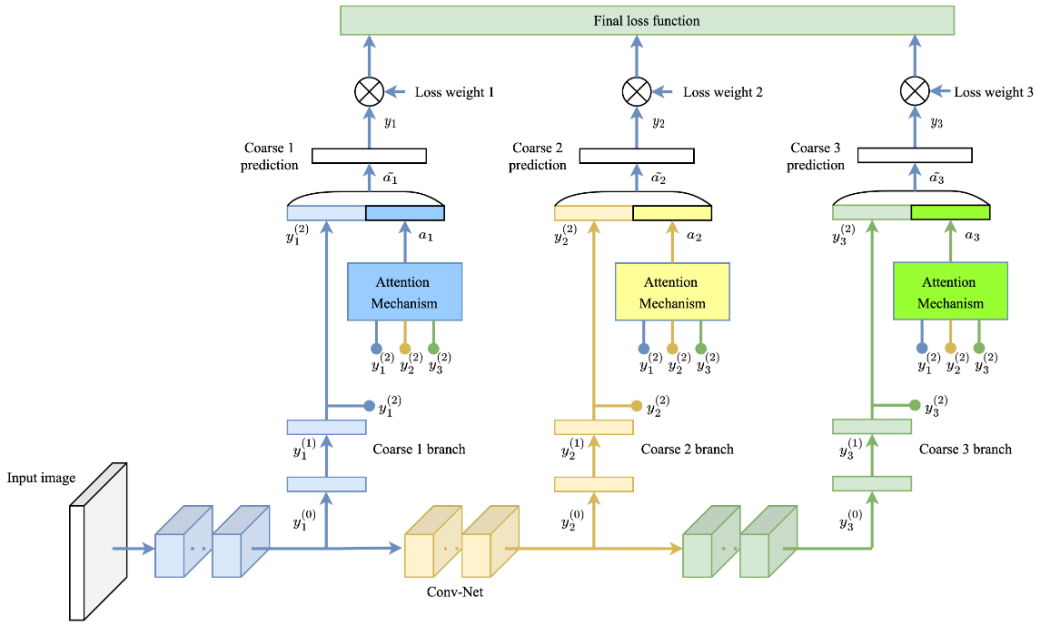


Figure 4: Architecture of the BA-CNN [32] model, composed of a CNN backbone followed by three branches linked by attention mechanism *(Image taken from [32])*

Having reviewed relevant work and techniques for plankton recognition, overcoming LT distributions and hierarchical classification, we present in the next two sections of this paper our experiments on plankton image classification, including the different methods used and result interpretation. In section 3, we first establish a plankton classification baseline on the *Zooscan* and *WHOI* datasets by testing several models and hyper-parameters. In section 4, we go further into our analysis on the *Zooscan* dataset by reformulating our plankton recognition problem as a hierarchical one, in order to increase model performance by taking into account the taxonomic relationships of plankton species. Follows a shorter section 5 on the development of a pipeline to use the *Grid5000* cluster in France - whilst there is less direct relevancy for plankton classification, this was an essential piece of work during my internship which allowed me to increase computational power for running plankton experiments, and ended up being adopted for other projects carried out by the science team of Inria Chile.

Disclaimer: The next three sections detail the three main workstreams I completed during my internship. I had the privilege to work with Andrew Berry, a researcher at Inria Chile, who supported me along the three missions. As the internship report guidelines request a clear distinction between the intern's work and the work from other researchers, I would like to emphasise on the following:

- For the work related to section 3, Andrew Berry and I worked hand-in-hand in almost every aspect of the project (literature review, hyper-parameters choice, model training and results analysis). The only distinction to be made is that I personally focused on experiments with ViT models, whereas Andrew worked on CNN-like architectures.
- I completed section 4 alone, with Andrew Berry only keeping a supervising role. I had full freedom to experiment anything related to hierarchical classification, my single objective being to outperform the results of the work completed in section 3.
- In respect of the workstream detailed in section 5, I have carried it out independently and at my own initiative, although the scale of this project rapidly raised interest from other researchers who provided me with their feedback.

As a result, the formal "we" used in this paper refers to Andrew and I in the first section, and to myself in the second and third ones. In addition, please note that at the time of my arrival, the project was in early stages whereby only the data processing had been completed (importing datasets and visualising images).

3 Establishing Baseline

3.1 Objectives

Building on the literature review from section 2 and the long term motivation for plankton recognition from section 1, our goal for this section is to build a deep learning model which would (i) take the plankton images available in the *Zooscan* and *WHOI* datasets, and (ii) classify images according to plankton species (each corresponding to a "class" in the rest of this paper). The main challenge we endeavour to overcome in this section is our ability to use these two raw datasets in their entirety, without any pre-modification. In addition, and as opposed to the work described in 2.1, we only use sample images without any additional feature pre-extracted from the images. The first objective is to assess the feasibility of such endeavour, then perform a hyperparameter search in order to establish a baseline, to which any future plankton classification work can be compared.

3.2 Proposed Method

To accomplish those objectives, we propose to evaluate different hyperparameters (see list thereafter), using various SOTA CNN and transformer models. Every experiment is launched using *Pytorch Lightning*, and we juxtapose to every backbone a linear layer (called *head*) to classify into the correct number of classes. The chosen image size is $224 * 224$, except for the *ConvNeXt* model which requires a $299 * 299$ image size. We have decided to use the following models:

- CNN types:
  - An *Xception* [4] model, using pretrained weights from *ImageNet1K*[6] training;
  - *Efficientnet* v2[40], using pretrained weights from *ImageNet1K* training;
  - The *ConvNeXt* architecture[25], using pretrained weights from *ImageNet1K* training;
  - The *Inception* architecture[39], using pretrained weights from *ImageNet1K* training.
- For ViT architectures:
  - A *Swin s3* vision transformer [24], using pretrained weights from *ImageNet1K* training;
  - A *DeiT* vision transformer [41], using pretrained weights from *ImageNet1K* training.

Using these models, we test the following hyperparameters:

- Augmentations, in particular *RandomAugmentations* 2.2.2 instead of traditional manual augmentations;
- Loss functions, especially focal loss 2.2.3 instead of cross-entropy loss;
- Samplers 2.2.1, notably a weighted random sampler ("WRS") instead of a random sampler.

We train every experiment for 25 epochs[7], and we divide the dataset in "Train", "Validation" and "Test" subsets according to a random 60/20/20 separation. We will monitor Accuracy, F1Score, Precision and Recall to assess our model performance. We use the *ReduceLRonPlateau*[8] learning rate scheduler, alongside the *Pytorch Lightning* automatic learning rate finder, which is conditioned to the F1Score metric.

3.3 Results

3.3.1 Experiment results on *Zooscan*

Our results on the *Zooscan* dataset are displayed in Table 1, and results on the *WHOI* dataset in Table 2. Given the fact that the two datasets are similar in structure, we have decided to thoroughly experiment our hyperparameter search on *Zooscan*, then apply our results on *WHOI*. Also, this also contributes to increasing search speed, as *WHOI* is three times larger than *Zooscan*.

Table 1 summarizes our most relevant experiments and results. For every experiment, we can see the model architecture, the choice of loss function, the scheduler, the use or not of the WRS, and the type of augmentation used. From these, we conclude that the most suitable and powerful model on *Zooscan* of all the ones we have tested is the *Swin* transformer coupled with a cross-entropy loss, a WRS, manual augmentations and a *ReduceLRonPlateau* learning rate scheduler.

Let's try to finely interpret these results and explain why certain parameters, which intuitively seemed beneficial to our analysis, did not work. In particular:

- When it comes to the model itself, ViTs, and notably *Swin* transformers, give overall better results for every metric than CNN-like models (experiments Zooscan_v2 and Zooscan_v5). This is consistent with usual computer vision benchmarks, in which ViTs have outperformed CNNs since 2020[7]. We note however that the metrics from the DeiT models are tailed by the *Xception* (experiment Zooscan_v8) and *Inception* (experiment Zooscan_v11) models, which score respectively 2.2 and 2.7 points below in F1Score.

[6] ImageNet is a very large annotated image database, hosting more than 14 million images classified in more than 22 000 different classes. ImageNet1K refers to a subset of this database limited to 1000 classes. It was notably used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) between 2012 and 2017. ImageNet1K holds a particular place in computer vision, as it is the most used benchmark, on which every model is tested. Due to the variety of images in ImageNet1K, it is often used to pretrain deep learning models.

[7] In machine learning, the number of epochs refers to the number of time all samples from a dataset is run through the model during training time. In our case, 25 epochs means that each image went through the model 25 times.

[8] A learning rate scheduler is a function that modulates the learning rate during training. There exist many types of scheduler, such as the exponential scheduler, the cosine annealing scheduler or ReduceLRonPlateau. ReduceLRonPlateau is a particular scheduler, in the sense that it is conditioned to a specific metric. Once a certain metric stops improving during the training, the learning rate will be reduced by a specified amount, to prevent the training staling.

- With respect to other architectures, we note the poor performance of the *EfficientNet* and *ConvNeXt* models, being respectively 17.3 (89.6 vs. 71.9) and 40.1 (86.4 vs. 46.3) points lower than their *Swin* conterpart in F1Score. For the *EfficientNet* V2 model, this is directly due to its smaller size (22 million parameters vs. 70 million for our *Swin*). For *ConvNeXt*, this substantial difference is probably related to the fact that it requires larger image sizes (299 pixels vs. 224 for all other models), which puts it at a disadvantage with poorer resolution, given that base images are already very small.
- The WRS has a significant impact. Zooscan_v7 performs better than Zooscan_v10 by about 11.3 points (84.9 vs. 73.6) in F1Score. The difference between Zooscan_v11 and Zooscan_v8 is more than 40.7 points (85.5 vs 43.8) for the F1Score. We therefore conclude that the WRS combined with cross-entropy enable the model to handle the LT distribution correctly[9], while the WRS with focal loss creates an overfit on minority classes.
- In terms of loss, we find that focal loss is underperforming cross-entropy on all metrics, at least for the *Swin*, DeiT and *Xception* models (experiments Zooscan_v1 vs Zooscan_v2; Zooscan_v3 vs Zooscan_v5; Zooscan_v7 vs Zooscan_v8). Indeed, when it comes to imbalances, the WRS is better than focal loss (Zooscan_v10 vs Zooscan_v8, where the WRS has a 11.9 points advantage in F1score - 85.5 vs 73.6 -).
- Random augmentations are not effective to solve our problem. Zooscan_v4 scores 35.3 points lower in F1Score compared to Zooscan_v3. We believe that this is due to the intrinsically poor quality of the dataset images combined with the relatively small distance between samples. In short, random augmentations are too violent for our plankton classification problem. Even when tweaking the number and magnitude of the random augmentations, manual augmentations remain a better choice.

Experiments on Zooscan dataset									
Experiments Names	Hyperparameters					Metrics			
	Model	Loss	Scheduler	Sampler	Augm.	Acc	F1	Pre	Rec
Zooscan_v1	DeiT	Focal	ReduceLR	Yes	Manual*	75.1	77.8	82.7	75.1
Zooscan_v2	DeiT	CE	ReduceLR	Yes	Manual*	86.2	87.7	90.2	86.2
Zooscan_v3	Swin	Focal	ReduceLR	Yes	Manual*	85.1	86.4	88.8	85.1
Zooscan_v4	Swin	Focal	ReduceLR	Yes	Rand†	47.5	51.1	61.2	47.5
Zooscan_v5	Swin	CE	ReduceLR	Yes	Manual*	88.5	89.6	91.1	88.5
Zooscan_v6 ‡	Swin	CE	ReduceLR	Yes	Manual*	87.7	89.1	90.9	87.7
Zooscan_v7	Xception	Focal	ReduceLR	Yes	Manual*	84.9	84.9	85.4	84.9
Zooscan_v8	Xception	CE	ReduceLR	Yes	Manual*	84.7	85.5	87.3	84.7
Zooscan_v9 ‡	Xception	CE	ReduceLR	Yes	Manual*	84.9	85.5	87.0	84.9
Zooscan_v10	Xception	Focal	ReduceLR	No	Manual*	74.8	73.6	75.4	74.8
Zooscan_v11	Xception	CE	ReduceLR	No	Manual*	75.1	43.8	40.7	48.2
Zooscan_v12	Inception	CE	ReduceLR	Yes	Manual*	84.4	85.0	86.0	84.3
Zooscan_v13	ConvNeXt	Focal	ReduceLR	Yes	Manual*	42.9	46.3	57.2	42.9
Zooscan_v14	EfficientNet	CE	ReduceLR	Yes	Manual*	68.6	71.9	79.2	68.6

* Manual augmentation corresponds to random flips and rotation of the images;
† Random augmentation uses all augmentation available in the *torchvision* library including flips, crops, gaussian blurs, perspective and colour inversion (magnitude and augmentation number were 8 and 4 respectively);
‡ Experiment uses already trained backbone and finetunes it for an additional 10 epochs.

Table 1: Results from all *Zooscan* experiments. Specific hyperparameters are shown: model architecture, loss function (focal loss or cross-entropy), learning-rate scheduler, use or not of a weighted random sampler, and type of image augmentation used (random or manual augmentations). Metrics used are Accuracy, F1Score, Precision and Recall, calculated as a macro-average (except F1Score which uses a weighted average).

[9] A high accuracy combined with a low F1Score, Precision and Recall would be a sign that the model is not able to identify tail classes

Finally, we test several experiments which do not lead to meaningful conclusions:

- In an attempt to finetune[10] already trained backbones on Zooscan, in experiments Zooscan_v6 and Zooscan_v9, we freeze the feature extractors from experiments Zooscan_v5 and Zooscan_v8 respectively, and train the classifier layer for 10 additional epochs. This, however, does not improve the results as it shows that our classifier head had already converged. According to the metrics and loss graphs, 25 epochs are therefore enough to have properly converging models.
- We also try to train an *Xception* model from scratch with random initial weights (rather than the pre-trained weights from *ImageNet1K*). We find that the difference in metrics is insignificant, the only difference being that we have to train the model way longer (50 epochs) for it to converge.

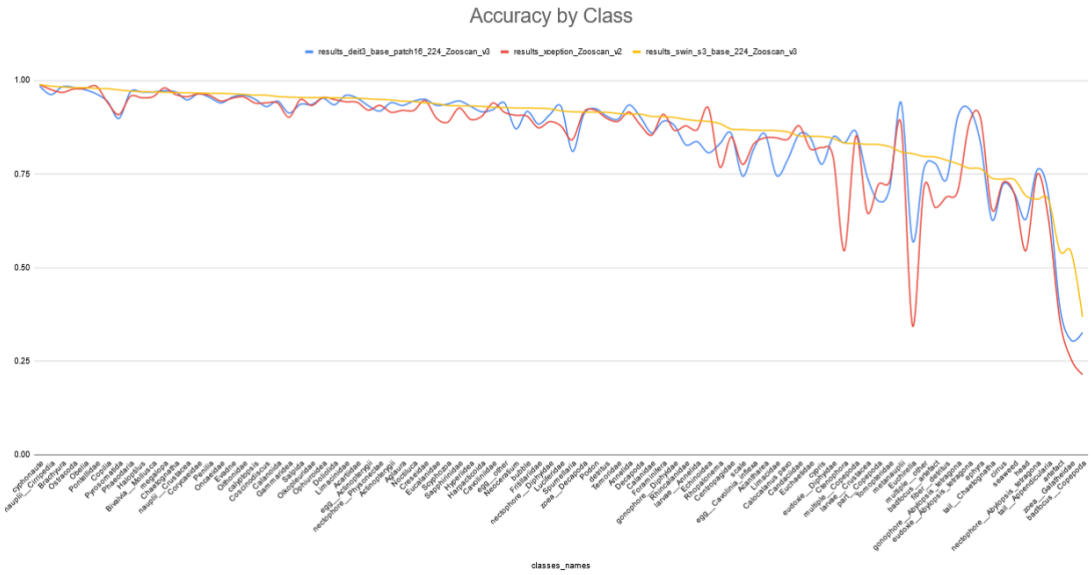


Figure 5: Accuracy by class for the *ZooScan* dataset for three different experiments: Zooscan_v2 in blue, Zooscan_v5 in yellow and Zooscan_v9 in red. Class order follows results of Zooscan_v5.

In Figure 5 we plot the accuracy of each individual class experiments Zooscan_v2, Zooscan_v5 and Zooscan_v9. This provides a better grasp of what is happening at a lower level rather than on global metrics, as we can see which classes suffer from poor performance (*i.e.* the hard classes):

- First, we note that although our models perform relatively well as a whole (between 85% and 99% for individual class accuracy for the first 60% of classes), results are not uniformly distributed across classes. We clearly see that individual class accuracy starts to drop below the global average accuracy from Table 1 at the 60% mark, and that this is worse for the last 5 classes, which have individual accuracy of less than 60%. This was expected given the imbalanced nature of our dataset.
- However, we note that unrecognised classes are not always the smallest ones in size, but rather the classes sharing similar features with other classes (see Section 3.3.3 for visualisation). This shows that class imbalance is only one part of the problem. For example, the class *Badfocus_Copepoda* (the class with the lowest accuracy over all) shares a lot of features with the *Copepoda* and *Badfocus* classes, which leads to a degree of confusion between these classes despite its 1000-sample size. Another example are classes *head_Chaetognatha* and *tail_Chaetognatha* which both rank amongst the lowest classes despite having respectively 300 and 3300 samples. There is indeed some confusion with the *Chaetognatha* class, which ranks amongst the best classes, because the only difference between the three lies in the complete or partial nature of the image. On the other end of the performance spectrum, the smallest class, *Ctenophora*, reaches 80% accuracy despite being only 39 samples large, due to the fact that it is relatively isolated in the feature space.

[10]In classification, finetuning refers to the act of freezing the backbone or feature extractor, and train only the classifier head. This proves particularly useful with pretrained models where we want to assess the transferability of a particular model (see section 3.3.2 for such an example). In our specific case, finetuning assesses if, by training the classifier head more, performance is increased.

- We also observe that, while performance order is similar across models, the DeiT and *Xception* models seem to have very similar class results on the second half of the graph, which could indicate that they have similar feature extraction. This may impact ensembling of these two architectures (see section 3.3.5).

3.3.2 Experiment results on *WHOI*

Experiments on WHOI dataset									
Experiments Names	Hyperparameters					Metrics			
	Model	Loss	Scheduler	Sampler	Augm.	Acc	F1	Pre	Rec
WHOI_v1	Swin	CE	ReduceLR	Yes	Manual*	84.7	86.1	88.2	84.7
WHOI_v2†	Swin	CE	ReduceLR	Yes	Manual*	76,6	27.2	36.9	19.1
WHOI_v3	Swin	Focal	ReduceLR	Yes	Manual*	74.2	57.3	59.1	54.6

* Manual augmentation corresponds to random flips and rotation of the images;
† This experiment was performed with a backbone trained on Zooscan, which was frozen during the WHOI training. This helps to evaluate the transferability of backbones for different datasets.

Table 2: Results from all *WHOI* experiments. Specific hyperparameters are shown: model architecture, loss function (focal loss or cross-entropy), learning-rate scheduler, use or not of a weighted random sampler, and type of image augmentation used (random or manual augmentations). Metrics used are Accuracy, F1Score, Precision and Recall, calculated as a macro-average.

After determining the best hyperparameters for the *Zooscan* dataset, we have decided to apply them to the larger *WHOI* dataset. We also added two other experiments:

- The same experiment with *Swin* transformers, but with focal loss, under the intuition that the even larger imbalances of *WHOI* would necessitate an additional LT technique;
- A second one using the backbone trained on *Zooscan* finetuned on *WHOI*, to evaluate the transferability between the two datasets.

  Results can be seen in table 2. We note the following outcomes:

- All metrics favor the cross-entropy experiment WHOI_v1, which is particularly effective given the large imbalance of the dataset. Indeed, a 86% F1Score on such a dataset is a remarkable result highlighting the strength of current computer vision tools and architectures. However, drawing a comparison between cross-entropy and focal loss is not exactly fair: results for the focal loss are underwhelming, and do not reflect the performances we noticed for *Zooscan* in section 3.3.1. One hypothesis is that the focal penalisation of the focal loss is too harsh, making model convergence impossible. If true, an alternative would be to tune the γ parameter of the focal loss (of which the value is 2).
- The finetuning experiment does not reach the desired results, and the learning process during *Zooscan* training is not sufficient to tackle the *WHOI* dataset. This was to be expected as *WHOI* is, in a way, a more extreme dataset, being larger with a even more imbalanced distribution. In addition, the *mix* class representing 70% of the *WHOI* dataset is more diverse than the *detritus* class in *Zooscan* (the two corresponding to poor plankton images, dead species, small particles found in the oceans...). As a result, direct transfer from *Zooscan* to *WHOI* is not possible by classic means.
- We note however that the backbone pre-training made on *Zooscan* accelerated the convergence of our *Swin* transformer, making the transfer-learning process between the two still useful.

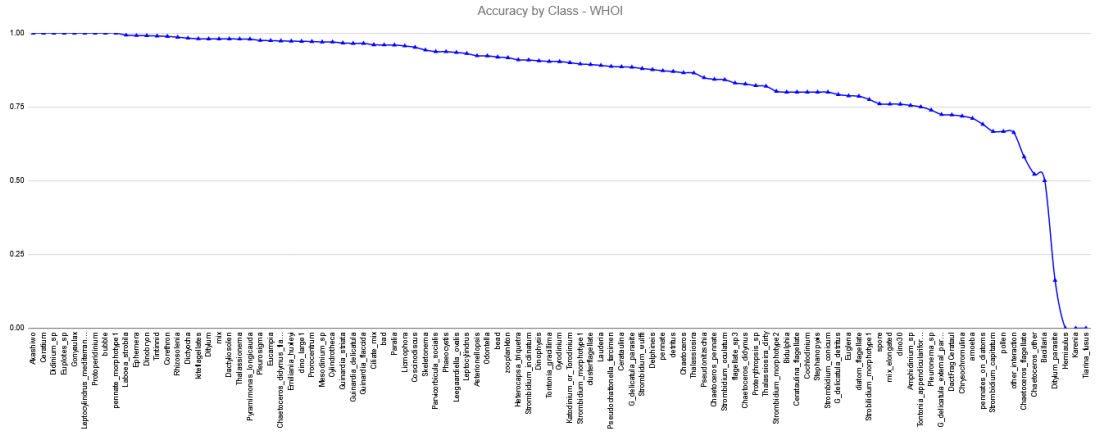


Figure 6: Accuracy by class for the *WHOI* dataset for the WHOI_v1 experiment, using *Swin* transformers and cross-entropy loss

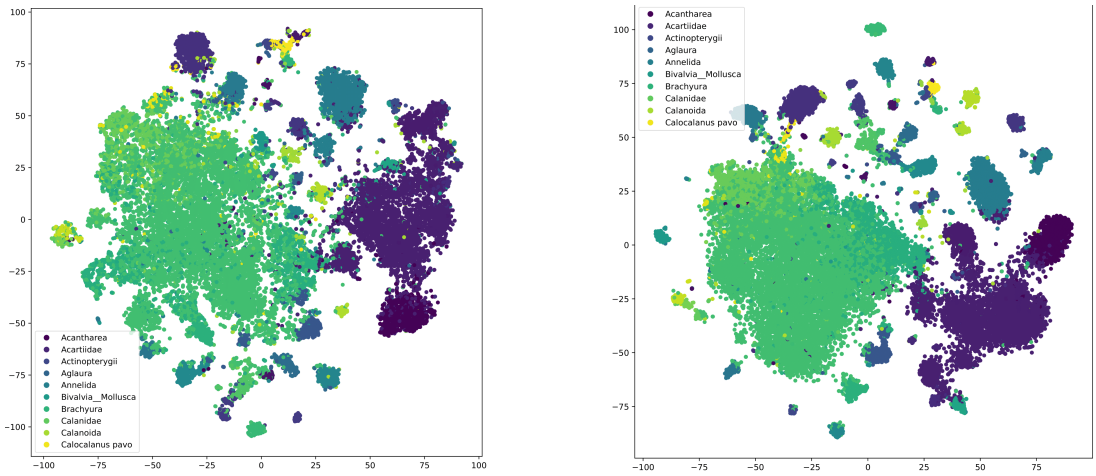
- Finally, looking at accuracy by class graph in Figure 6, we note that, although accuracy is very high for a lot of classes, tail classes suffer from a significant decrease in performance. This is similar to the behaviour we found in *Zooscan*. However, this process is even harsher in *WHOI*, as some classes, such as the *Ditylum_parasite*, *Hemiaulus*, *Karenia* and *Tiarina_fusus*, all have an individual accuracy of 0%. While these statistics are not always consistent (*e.g.* the *Ditylum_parasite* class only displays 4 samples, one of which in the test subset), the classifier is still unable to recognise these 4 classes.

3.3.3 Embedding visualisation

To better visualise our model's mechanics, we have extracted the embedding space created by our best experiments after the first training epoch and at the end of training. We can see these spaces in Figures 7a, 7b, 8a and 8b using t-distributed stochastic neighbor embedding ("t-SNE")[11] visualisation. Embedding comes from the last layer of each backbone, just before the classifier layer.

- Our first observation is the difference of sample distribution between the first epoch in figure 7a and the last training epoch in figure 7b. Despite a lot of superposition due to drastic reduction in dimension, we note that class clusters are cleaner in the latter thanks to the effect of the cross-entropy loss function.
- We note however that distributions are blurred by the sheer size and standard deviation of some classes, or subset thereof. This can be observed in figure 8a representing the different sub classes taken from the *Zooscan* taxonomic tree, where two sub classes (*Irrelevant* and *Opithonska-Crustacea*, in purple and yellow respectively) dominate the embedding projection. When plotting the same sub classes distribution but excluding *Irrelevant* samples (artefacts, detritus and non plankton images), we achieve a much cleaner projection which shows clear boundaries between different classes, even though it is dominated by the *Opithonska-Crustacea* sub class.

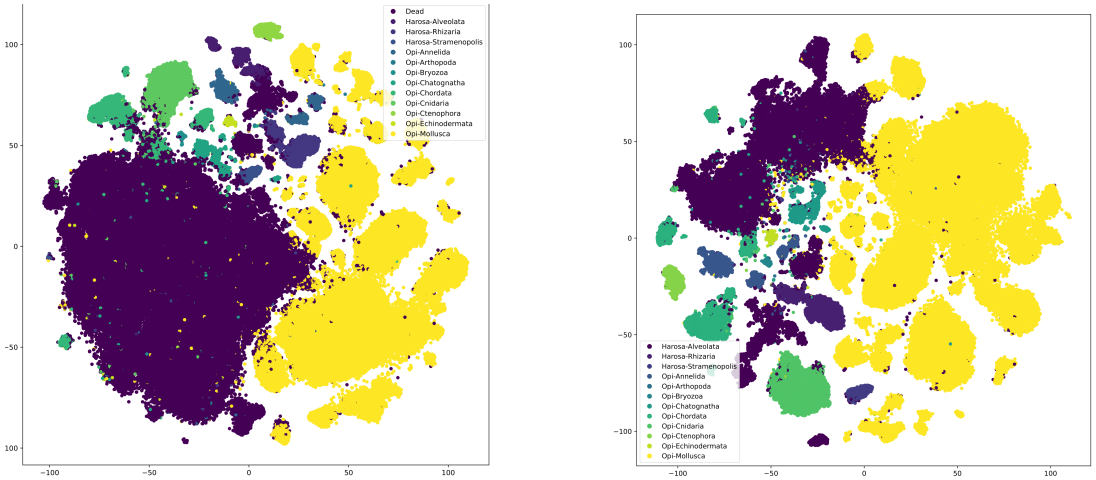
[11] t-SNE is a dimensionality reduction technique used for visualisation of spaces of large dimension. Although it is computationally more expensive than other methods such as Principal Component Analysis ("PCA"), it is proven to preserve more of the manifolds' structures. In our case, we combine PCA and t-SNE for faster computation by firstly applying PCA dimension reduction (embedding size 768 to 100), and then the t-SNE algorithm.



(a) Embedding at epoch 1

(b) Embedding at epoch 25

Figure 7: T-SNE visualisation of feature space at the beginning 7a and at the end7b Zooscan_v5 training. Labels refer to individual classes.



(a) T-SNE visualisation of subclasses

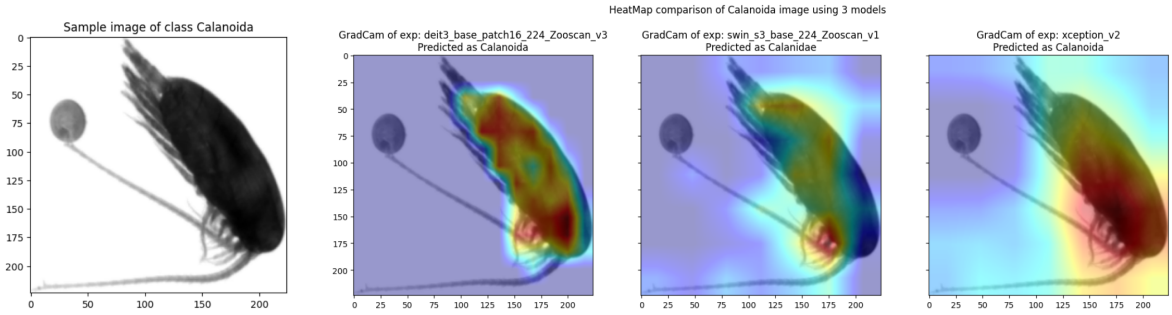
(b) T-SNE visualisation of subclasses

Figure 8: T-SNE visualisation of feature space at the end of Zooscan_v5 training for subclasses from the taxonomic tree. Subfigure 8a includes *irrelevant* samples, Subfigure 8b is without.

This highlights the major difficulty we face working with our datasets: in order to classify plankton species, the classifier should first be able to distinguish between *Irrelevant* and *Relevant* samples, so as to simplify its task. All the more so as sub classes are empirically close to one another in the embedding projection (see figures 8a and 8b), this new distinction problem lays the foundations for the second part of our work (see section 4).

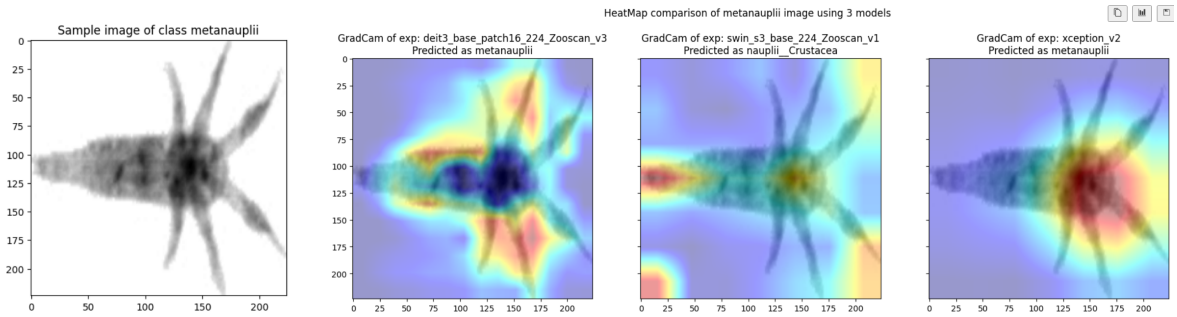
3.3.4 GradCam visualisation

To further refine our analysis, we also use a visualisation technique for computer vision deep learning models called *GradCam* [36], which allows to detect points of interest in an image by mapping gradients from a particular layer. Our initial intuition is that areas where gradient is the highest would correspond to the most discriminating part of the image, *i.e.* the location responsible for the model prediction. This would help explaining better the mechanics of our deep learning models.



(a) Base Image (b) *GradCam* heatmaps from 3 experiments using different models

Figure 9: Sample from the *Calanoida* class and its heatmap signatures in 3 different models (from left to right: DeiT, *Swin* and *Xception*) using *GradCam*



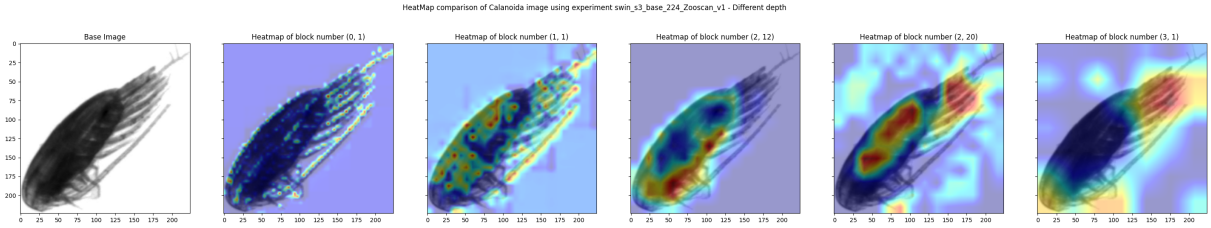
(a) Base Image (b) *GradCam* heatmaps from 3 experiments using different models

Figure 10: Sample from the *Metanauplii* class and its heatmap signatures in 3 different models (from left to right: DeiT, *Swin* and *Xception*) using *GradCam*

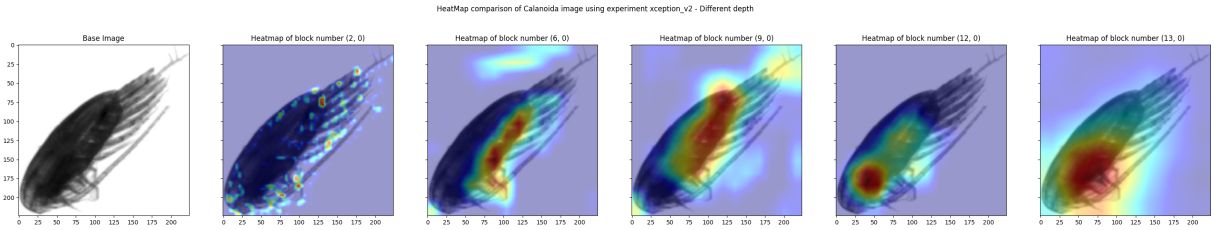
- Figures 9 and 10 display *GradCam* heatmaps from the *Calanoida* and *Metanauplii* classes. The former is a very well documented plankton specie with more than $200,000$ samples, and as such has a large standard deviation; the latter refers to an intermediate development stage of crustacean plankton and is relatively small with only 186 samples. These figures provide comfort that our models are able to identify important locations within the images, namely the body and the legs of the *Calanoida*, or the claws of the *Metanauplii*, which are some of the class features. Additionally, in these two samples the *Swin* transformer misclassifies the images, as shown by rough heatmaps, especially for the *Metanauplii* sample where heat points focus on less relevant parts (corners and tail). *GradCam* images are particularly beneficial in identifying misclassifications and errors in our models.

- Figures 11 and 12 display other samples from the same *Calanoida* and *metanauplii* classes, but now visualising gradients at different depth inside the models (respectively *Swin*, *Xception* and DeiT, from top to bottom). This allows us to follow gradient flow from layer to layer, thus providing information on how a model handles an image, further helping us understanding its inferences.

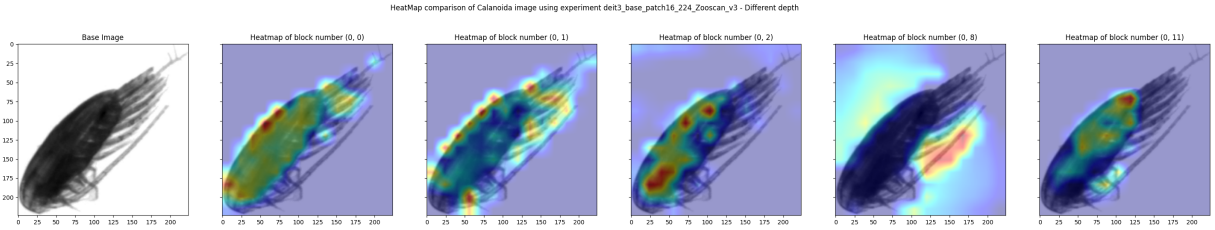
- From Figure 9, we observe that all models pick up the different parts and features of the sample. First, they identify the legs, then highlight the dark oval body, and finally settle on the intersection between the two. There are however some notable differences between our models: *Swin* focuses on the tail of the sample (at the top of the sample image) whereas *Xception* is geared towards the head of the plankton (at the bottom of the image) and DeiT highlights the long antenna of the sample in the 4^{th} heatmap of Figure 11c.



(a) *GradCam* visualisation at different depths in a *Swin* transformer



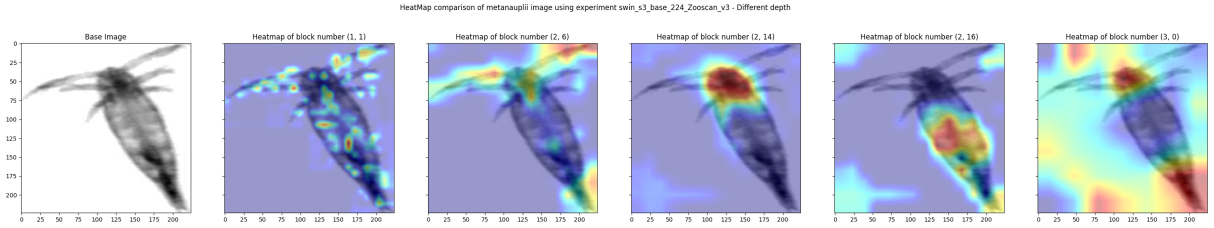
(b) *GradCam* visualisation at different depths in an *Xception* model



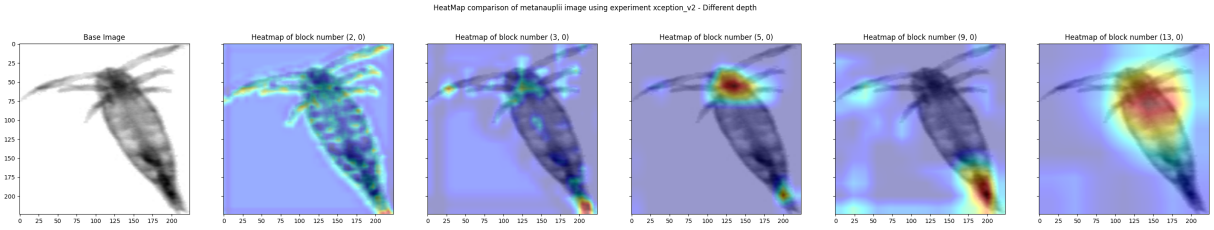
(c) *GradCam* visualisation at different depths in a DeiT transformer

Figure 11: *GradCam* heatmaps at different depths of 3 different models using a *Calanoida* class sample

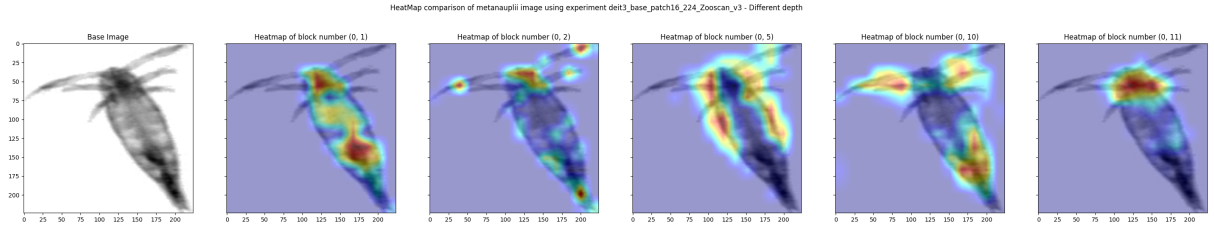
- A similar analysis can be drawn from Figure 12. All models sequentially focus on the dark body of the *metanauplii*, its claws, its tail and finally its head, where body and claws intersect. In Figure 12a, we understand the benefit of looking at multiple depths: while the final image is overexposed, we can decompose important parts of the images by observing other layers.
- A final observation relates to the difference between ViT and CNN models. On *GradCam* images, CNN is able to progressively recognise more important local features, resulting in concentrated heatmaps (see Figure 11b). Transformers on their side aggregate small images patches, enabling recognition of multiple non-adjacent parts of the image, hence less concentrated heatmaps (see Figure 12c).



(a) *GradCam* visualisation at different depths in a *Swin* transformer



(b) *GradCam* visualisation at different depths in a *Xception* model



(c) *GradCam* visualisation at different depths in a DeiT transformer

Figure 12: A figure with three subfigures

3.3.5 Ensemble experiments

As a last endeavour as we benefit from numerous trained backbones, we test whether results could be boosted by aggregating our models. Our inspiration came from [19], in which ensembles are used to establish new baselines on various ecological classification benchmarks and datasets. For this purpose, we used the models from experiments Zooscan_v2, Zooscan_v5 and Zooscan_v9, and created ensembles according to three different voting strategies (see info paragraph below). Results from our ensemble experiments are displayed in Table 3.

Ensembling is a strategy used in machine learning to improve the performance of a model. The idea is to aggregate different models as well as their results, in order to produce a stronger, more robust and performing model by leveraging model diversity. The concept is based on the stochastic nature of statistical learning. Ensemble methods are however more costly in terms of computational power, since multiple models are used. Several ensemble strategies differ in the way results are aggregated to produce the final prediction. In this paper, we used three types of aggregations:

- The majority voting strategy is simple: each model predicts a class, and the ensemble's prediction is the class with the highest amount of predictions.
- The arithmetic voting strategy uses the logits of each model, and adds them to create a new logit vector. The final predicted class is the $argmax$ of this new vector. An equal weight is given to every model in the final prediction.
- The geometric voting strategy also uses the logits of each model. The class weights of each model are multiplied to produce a new logits vector. The final predicted class is the $argmax$ of this new vector. This strategy tends to give more weight to very confident models (very high or very low class score) in the ensemble.

Ensembling experiments on Zooscan dataset								
Experiments Names	Models used in ensemble			Voting Strategy	Metrics			
	Swin	Deit	Xception		Acc	F1	Pre	Rec
Swin	X			None	88.5	89.6	91.1	88.5
DeiT		X		None	86.2	87.7	90.2	86.2
Xception			X	None	84.9	85.5	87.0	84.9
AllThree_M	X	X	X	Majority	87.8	89.4	92.1	87.8
AllThree_A	X	X	X	Arithmetic	88.4	90.2	93.1	88.4
AllThree_G	X	X	X	Geometric	88.4	90.2	93.8	88.3
SwinDeiT_A	X	X		Arithmetic	87.5	89.5	92.4	87.6
SwinDeiT_G	X	X		Geometric	87.6	89.7	92.6	87.6
SwinXcep_A	X		X	Arithmetic	88.5	89.9	92.3	88.5
SwinXcep_G	X		X	Geometric	88.3	89.9	92.8	88.3
DeiTXcep_A		X	X	Arithmetic	87.8	89.2	92.0	87.9
DeiTXcep_A		X	X	Geometric	87.8	89.3	92.7	87.8

Table 3: Results table of *Zooscan* ensemble experiments. Specific hyperparameters are shown: models and experiments used, as well as voting strategies employed. Metrics used are Accuracy, F1Score, Precision and Recall, calculated as a macro-average.

- From this table, we first notice that results are very similar for the arithmetic and geometric voting strategies, but that the majority voting strategy lags behind. This is to be expected, given the fact that our ensembles are relatively small (three different models maximum), and that the two other strategies are implemented at the logit level rather than at the class inference level. Overall, ensembling our models produces better results, although the increase in performance is moderate compared to the increase in required computational resources.
- Another important observation relates to the potential of combining CNN and transformers in ensembles: we note that *SwinXception* ensembles outperform the *SwinDeiT* ensembles, although the *DeiT* model is individually better that the *Xception*. One hypothesis is that the difference in the way CNN and ViT aggregate features makes the resulting ensemble more diverse, and therefore more accurate. This is also the case for the *DeiTXception* experiments, where the ensemble performance for every metric is significantly higher than the base accuracy of individual models.

3.4 Perspectives

Overall, our objectives for this section have been achieved through our experiments and analyses. We have successfully leveraged SOTA architectures and classic LT techniques to create simple yet performing models on two challenging datasets, notably:

- Our *Swin* transformer trained with WRS and cross-entropy from experiment Zooscan_v5 with a F1Score of 89.6%; and
- Our ensemble experiment AllThree_G with a 90.2% F1Score (*Xception*/DeiT/*Swin* ensemble under geometric voting strategy).

We have thus established new baselines in plankton classification by identifying relevant hyperparameters to tackle the plankton detection challenge, and we have effectively implemented some visualisation techniques to explain our models results in Sections 3.3.3 and 3.3.4.

That being said, our model's performances remain far away from the universal automatic plankton classifier we mentioned in this paper's introduction 1. Whilst overall results look promising, individual class results remain uneven, and other tools are needed to identify the fine grain differences of similar neighbouring classes. We try to tackle this new challenge in section 4.1.

4 Leveraging Taxonomic data

The following section is dedicated to my work during the second part of my internship, and builds upon the results of section 3, as well as the introduction on hierarchical classification in section 2.3.

4.1 Objectives

Even though the results of section 3 contribute to establishing a new baseline by combining SOTA computer vision models with traditional techniques to tackle class imbalances, they are far from what could be expected of an automatic plankton classifier. Some classes remain poorly classified (see Figure 5), and our models still lack explainability and robustness in the face of out-of-distribution samples ("OOD"s)[12]. In their current state, our models would not be suitable for industry usage.

In search of an improvement, we decided to leverage the fact that plankton species are organised in a taxonomic tree: every specie is part of a larger group called *genus*, which is in turn part of a taxonomic family (see Figure 13). From a computer vision perspective, we can draw on the idea that members of a same plankton family share common phenotype characteristics. Intuitively, as soon as the model is able to recognise a given family feature, it should be able to limit its computations to a subset of classes corresponding to the family members, thus reducing risks of misclassification. We should be able to leverage the metadata on classes to improve our plankton classification models in a similar fashion to the hierarchical classification (see section 2.3) typically used to increase performance in difficult classification problems (*e.g.* very large class number setup [45], few-shot learning [17] or LT distribution [42]).

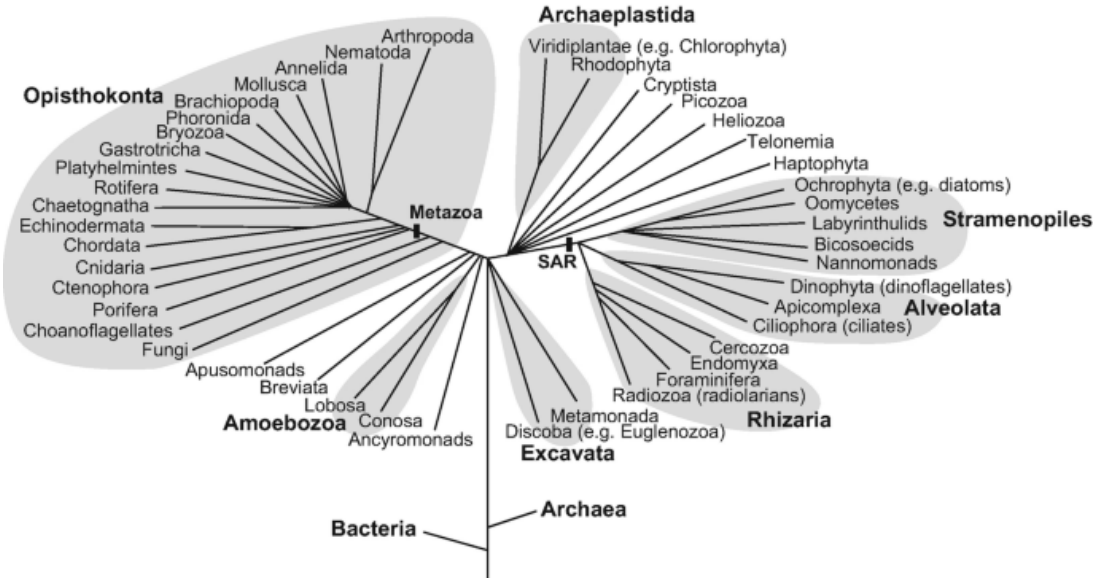


Figure 13: Example of plankton taxonomic tree

4.2 Proposed Method

Building on the baseline from previous section 3, we aim at leveraging the taxonomic plankton data available with the *Zooscan* dataset in order to increase our model performance both globally (for all classes) and locally (for each individual class). Our work for this section is divided in three parts: first, we build a hierarchical dataset adapted to deep learning, then we replicate proven hierarchical architectures and frameworks to the *Zooscan* dataset, to *in fine* develop a novel architecture fitted for largely imbalanced datasets in ecological classification.

[12]OOD is a field of machine learning which aims at tackling samples which follow a different distribution than the one on which the model has been trained on. In our classification case, OOD would help answering "How does our model behave when we show it a picture of a cat instead of a plankton image ?". Correctly handling OOD is a difficult task, but OOD detection is an important feature in many usages, particularly in open-world settings. Whilst in this section we do not implement OOD detection *stricto sensu*, we believe that implementing a hierarchical model can allow a better OOD handling.

4.2.1 Building the Hierarchical Dataset

Class hierarchy is paramount to a model's performance and coherence. As such, an effective hierarchy must abide by the following rules:

- Firstly, the hierarchy should closely match phenotype characteristics for the model to be guided towards the objective;
- Secondly, for the hierarchy to compatible with deep learning models, it must be balanced and shallow. Such requirement makes even more sense when facing LT distributions;
- Finally, in real-life examples such as plankton recognition, this class hierarchy should be based on observable characteristics. Taxonomic data, for example, is perfectly adapted for this purpose.

While the taxonomic tree available with *Zooscan* is of particular relevance and thoroughness, it does not fully comply to the aforementioned criteria, especially as it is deep and unbalanced.

We therefore settle[13] for a 3-layer deep tree (corresponding to the usual depth of trees of hierarchical baselines like [49] [32] [46]), a visualisation of which can be found in figure 14.

- A first layer differentiates between *relevant* images, which correspond to actual plankton images, and *irrelevant* images which group together other types of images (namely detritus, seaweeds, badly focused images, and plankton eggs);
- A second layer corresponds to large families of plankton (*e.g. crustacea* or *mollusca*) for the *relevant* subtree, and thematic groups for the *irrelevant* subtree (*e.g. artefact* or *eggs*);
- A third layer matches the 93 classes of the *Zooscan* dataset.

Here, the classification problem remains heavily imbalanced, but we preferred to adhere to the actual plankton taxonomy. That being said, the binary classification at the beginning is almost balanced.

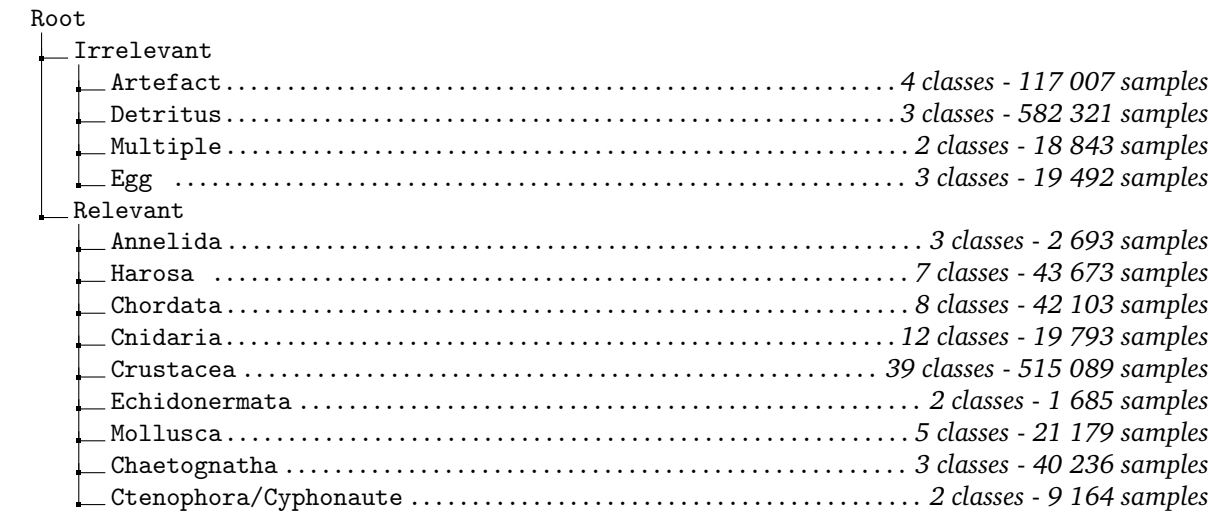


Figure 14: Hierarchical tree structure of the *Zooscan* dataset

4.2.2 Replicating baseline

Now that we have a coherent taxonomy, the next step is to apply SOTA architectures in hierarchical classification to our problem. For this, we have selected 3 baselines described in section 2.3:

- The B-CNN described in [49] which corresponds to the baseline for hierarchical models;
- The HB-CNN from [46] which adds a top-down feature flow as well as a bilinear module to the B-CNN architecture;

[13] This choice is empirical, and as mentioned thereafter in section 4.4, we believe that optimising the tree could lead to model improvements. However, our goal is first and foremost to point out that hierarchical data is useful in the plankton recognition context.

- The BA-CNN from [32] which implements attention mechanisms to relate labels.

We trained each model using the four-fold Branch training ("BT") strategy described in [49], switching loss weights at epoch 15, 25 and 35. We trained all models until epoch 50. Used weights are respectively $(0.98, 0.01, 0.01)$, $(0.1, 0.8, 0.1)$, $(0.1, 0.2, 0.7)$ and $(0.01, 0.01, 0.98)$.

Branch training: This is a training strategy used in hierarchical models to help the model recognise visual patterns in a coarse to fine manner. The global loss function used is a weighted sum of cross-entropy at each layer, the underlying idea being to iteratively switch the weights of the loss during training to focus on different depths. We start on the first layer, then the second, the third, and end by almost only considering the cross-entropy of the last layer.

All models use the best backbone from section 3, that is to say a *Swin s3* transformer pretrained on *ImageNet1K*, fine-tuned on *Zooscan* for 25 epochs with a weighted random sampler and cross-entropy loss function. For the branch feature blocks, we used a *Swin s3* block for each, initiated with random weights. To accelerate training, reduce energy consumption and evaluate the transferability potential of our backbone, we freeze the parameters of the *Swin* backbone (but not the ones from the small branch feature blocks).

To follow training and assess model performance the Accuracy, Precision, Recall and F1 score are computed at every layer.

4.2.3 Separated BA-CNN ("SBA-CNN")

In addition to our chosen hierarchical baselines (section 4.2.2), we propose a custom architecture named SBA-CNN. Building on the efficiency of the BA-CNN architecture, we realise that half of *Zooscan* images are considered detritus or artefact-like, which means that that half of the samples a model will test is not actual plankton imagery. This creates some issues for fine-grained classification - indeed, if detritus images increase model robustness at a coarse level, they may also occult differences between smaller and more similar classes. We propose to solve this issue by splitting the flow of images after a binary classification, to eliminate parameter bias towards larger irrelevant classes.

One downside of segregating between *relevant* and *irrelevant* images is a risk of error propagation, as the errors of the binary classifier may be propagated to the rest of the model. That being said, baselines experiments have shown that it is possible to differentiate between relevant and irrelevant images with a very high accuracy (more than 98% according to Table 4), and another mitigant would be to add an OOD detector at each model step to tag such misclassification. This makes our initial approach coherent with the objectives of our work.

After initial segregation, images are fed into two BA-CNN-like models. An important consideration is that both models' attention mechanisms must be connected to allow for efficient communication between the two model branches (see Figure 15 for visual details). This idea is supported by the confusion matrices of section 3.3 experiments, in which neighbouring *relevant* and *irrelevant* classes were mixed-up. This could be the case for the class *Badfocus_Copepoda*, which is at the intersection between the *Badfocus* class, and all other classes from the *Copepoda* family. Our intuition is that features from the *irrelevant* branches could help during the *relevant* branch inference, and vice-versa: each branch receiving the other branch features would certainly enable that. The loss will however not count parameters from the other branch, leaving each branch independent in practice. Figure 15 provides a visual explanation of this model.

This model is also trained using BT strategy, with the same weights, backbone and epochs threshold as the ones described in section 4.2.2. To follow training and assess model performance, we compute the model Accuracy, Precision, Recall and F1 Score at each of the five steps (the binary classification, and the two levels of each branch).

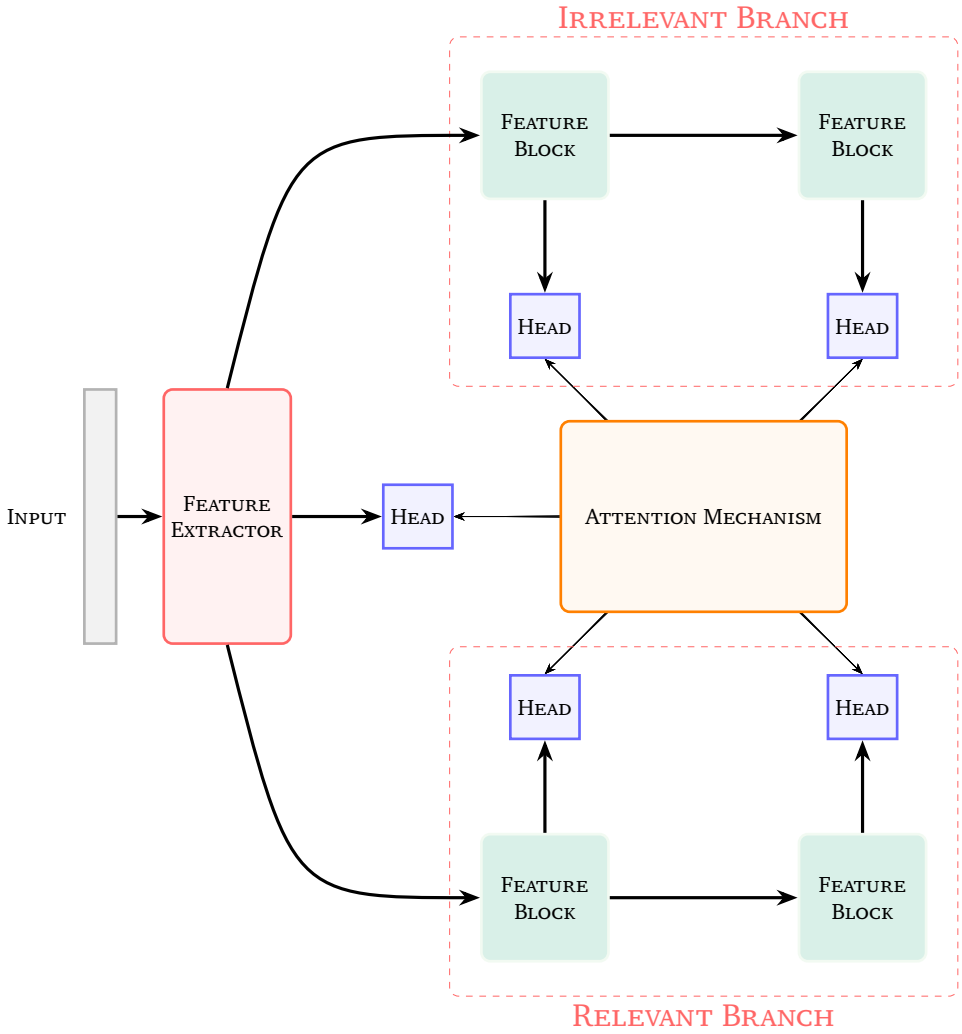


Figure 15: Global Architecture of the SBA-CNN model, featuring two separate branches for *relevant* and *irrelevant* samples, and a common attention mechanism.

4.3 Results

4.3.1 Quantitative Results

We train the four models (B-CNN, HB-CNN, BA-CNN and SBA-CNN) on the custom hierarchical version of *Zooscan* described in Section 4.2.1. To assess and compare the results of each architecture, we have gathered different types of metrics for each experiment. Main results can be found in Table 4: for each of the three layers, we computed the Accuracy, F1Score, Precision and Recall. We have also added the flat baseline, which correspond to the Zooscan_v5 experiment of section 3.3 (a *Swin s3* transformer trained on *Zooscan* with cross-entropy) - to state the obvious, the flat baseline does not have metrics for the first two layers. In Table 5, we have also added the results from the SBA-CNN model for each of its 5 branches (the binary classification, two branches for *relevant* samples, two branches for *irrelevant* samples.). We will however not directly analyse those results, as they cannot be compared to other baselines.

- The first thing to note from Table 4, is that the results from our hierarchical experiments are better than our *Swin* flat baseline by a substantial margin - by 3 to 4% in all metrics - at least for the B-CNN, HB-CNN and BA-CNN architectures. This is particularly important, as it proves that label hierarchy are indeed relevant to the classification problem - we are searching into the right direction.

There however are some differences in the results from the four architectures:

- Although the B-CNN and HB-CNN models have a similar performance in layers 1 and 2, the B-CNN performs slightly better on the third layer. This observation is somewhat unexpected, as HB-CNN is

supposed to be more advanced. This can be due to the fact that coarse features are not fully relevant in the final layer classification, and such features may in fact disturb the final classification [14].

- The BA-CNN model significantly outperforms all other models on layer 1, and is very similar to B-CNN in other layers. According to its original implementation [32], outperformance on branch 1 is to be expected given BA-CNN is the only model for which the first branch has a feedback from the deeper layers, which helps in the inference process. At this stage, BA-CNN is our best model.
- Our SBA-CNN model disappoints with performance systematically behind all other models in branch 2 and 3, but is on par with hierarchical models for branch 1[15]. However, we note that, from the original results in Table 5, there is a performance difference between branch 1 (corresponding to *irrelevant* samples) and branch 2 (corresponding to *relevant* samples) by about 8 points for all metrics (Branch_1_1 vs. Branch_1_2; Branch_1_2 vs. Branch_2_2), meaning that the SBA-CNN model is superior in the *Relevant* branch than in the *Irrelevant* branch. In fact, removing the distinction between *Relevant* and *Irrelevant* samples makes classification of relevant images easier (notably by removing the *detritus* class which is disturbing our results, see section 4.3.2), but increases the difficulty of *Irrelevant* classification, as it becomes a classification exercise between types of detritus and artefacts, hence explaining the underwhelming results from SBA-CNN for branch 2 and 3.

Hierarchical Experiments												
Experiment Names	First Level				Second Level				Third Level			
	Acc	F1	Pre	Rec	Acc	F1	Pre	Rec	Acc	F1	Pre	Rec
Flat Baseline	X	X	X	X	X	X	X	X	88.5	89.6	91.2	88.5
B-CNN	97.1	97.1	97.1	97.1	95.8	95.8	95.1	94.7	92.3	92.3	92.6	90.6
HB-CNN	97.2	97.2	97.2	97.2	95.6	95.5	94.8	94.0	92.0	91.9	91.8	89.0
BA-CNN	98.7	98.7	98.7	98.7	95.9	95.8	95.5	94.9	92.2	92.1	92.3	89.5
SBA-CNN	97.2	97.2	97.2	97.2	90.2	90.8	89.1	90.2	83.7	88.4	89.7	83.7

Table 4: Results table from hierarchical experiments. Individual results for each level in the taxonomic tree are shown. Metrics used at each level are Accuracy, F1Score, Precision and Recall, aggregated using a macro mean, except for F1Score which uses a weighted mean. The flat baseline results correspond to the Zooscan_v5 experiment from Section 3.

Results from the SBA-CNN architecture at each branch level				
Metric Name	Accuracy	F1Score	Precision	Recall
Binary Head	97.2	97.2	97.2	97.2
Branch_1_1	93.9	91.2	92.4	90.0
Branch_1_2	89.5	82.8	85.8	80.6
Branch_2_1	99.5	98.5	98.5	98.4
Branch_2_2	95.8	90.6	90.4	91.0

Table 5: Detailed results table from the Segregated BA-CNN experiments. Individual results for each branch of the model are shown for every metric (Accuracy, F1Score, Precision and Recall).

As a next step, building from Table 4 and 5, we analyse all 4 models on *relevant* and *irrelevant* separately, meaning that the same four metrics were calculated on each respective subsets of samples. Results from the two last branches of each models are available in Table 6.

From this Table 6, we draw the following observations:

- Models perform better on the *relevant* branch than on the *irrelevant* branch. This may come as a surprise as the *relevant* samples are the ones suffering from smallest class sizes. However, this is

[14]This may also be related to the fact we freeze the initial backbone, which had already learned fine-grain features. Nevertheless, additional experiments would be necessary to settle this question.

[15]A similar performance on branch 1 is expected, as for this layer all models are similar.

explained by the fact the taxonomic hierarchy is actually relevant for the *relevant* branch only, and that *relevant* classes have a smaller variance compared to the *irrelevant* classes. This is good news, as differentiating between different types of *detritus* in our context is not a priority.

- The SBA-CNN model for individual branches is on par with all other models, being the best with BA-CNN for the *relevant* sub-tree. This means that the level of information from the *irrelevant* branch needed to classify *relevant* samples is low, thus supporting the irrelevance of the *irrelevant* sub-tree from a computer science point of view.

Hierarchical results for individual branches										
Experiment Names	Specific Branch		Second Level				Third Level			
	Relevant	Irrelevant	Acc	F1	Pre	Rec	Acc	F1	Pre	Rec
B-CNN	X		98.9	99.6	94.3	98.9	94.5	96.5	95.3	94.5
HB-CNN	X		98.9	99.5	94.0	98.9	93.2	96.4	95.0	93.1
BA-CNN	X		99.2	99.7	96.8	99.2	93.1	96.5	95.3	93.1
SBA-CNN	X		99.2	99.7	99.2	99.2	93.1	96.2	91.4	93.1
B-CNN		X	91.9	94.2	87.7	91.9	83.8	90.1	87.0	83.1
HB-CNN		X	91.1	94.1	87.8	91.1	80.7	89.8	87.1	80.7
BA-CNN		X	92.0	94.4	91.7	92.0	83.1	90.1	88.0	83.1
SBA-CNN		X	89.8	93.9	92.6	89.8	80.0	89.5	85.3	80.0

Table 6: Results table from hierarchical experiments for the two branches. Individual results for each level in the taxonomic tree are shown. Metrics used at each level are Accuracy, F1Score, Precision and Recall, being aggregated using a macro mean, ecep for F1Score which uses a weighted mean.

Finally, we assess the output coherence for all four models. While classes follow a strict hierarchy, models have the ability to output a triplet of labels which may not fall in line with this hierarchy. For example, a model may predict the triplet (*Relevant*, *Harosa*, *Pontellidae*), even though the *Pontellidae* class is not from the *Harosa* family, but rather in the *Crustacea* subtree. This is to be avoided, first because we want the model to perform well on each level, second because we want our models' outputs to be trustworthy for outside users (a necessity for the model to be used in embarked vehicles). We have therefore calculated in Table 7 the Coherence accuracy [16] of all 4 models, both for the whole test set as well as for individual branches.

- Results from Table 7 show that, although coherence accuracy is high for every model, BA-CNN remains the clear winner. The flow of information both top-down and bottom-up allows the model to make better informed and more coherent inferences. We also note that the SBA-CNN displays a high coherence accuracy, in between the B-CNN/HB-CNN duo and the BA-CNN model. This was to be expected as its architecture inherently restricts it to more coherence, thanks to the initial segregation between *relevant* and *irrelevant* samples.

Coherence Accuracy				
Branches	Model architecture			
	B-CNN	HB-CNN	BA-CNN	SBA-CNN
Relevant	93.4	93.4	95.2	93.7
Irrelevant	87.0	87.2	88.5	87.7
Total	90.1	90.2	91.8	90.1

Table 7: Coherence accuracy table from all hierarchical experiments. Results are shown for the two separate branches, and for the total test set. The coherence accuracy is calculated as a *micro* average of all samples.

[16] We define coherence accuracy as accuracy applied to label triplets, meaning that only outputs matching the target for all three layers are considered correct.

As such, and viewing from Tables 4, 5, 6 and 7, we can affirm that the BA-CNN architecture is the best performing one on our *Zooscan* hierarchical dataset.

4.3.2 Qualitative Results

Alongside the quantitative results from section 4.3.1, we try to explain what happens qualitatively in the model's decision-making process. In Figures 17 and 16, we reconstruct the actual hierarchy built by the different models, from each model's predictions at every level. Here, the width of an edge (u, v) corresponds to the relative number of times the model predicted the nodes (u, v) compared to the actual number of samples of u. This impeaches the chart to be one-sided due to class imbalances. We then apply a square root regularisation to improve readability.

Our takeaways are as follows:

- In all three sub-graphs, the *detritus* subclass is confused with almost every class. This is coherent with our observations from section 3, in particular the visualisation from the embedding space of our feature extractors - the *detritus* class being broad and diverse, it inherently interferes with every other class.
- By comparing Figure 17c with 17a and 17b we also observe that, between the first layer and the second layer, predictions are less incoherent between *irrelevant* and *relevant* samples (*i.e.* samples for which the model predicts the *irrelevant* sample alongside a subclass from the *relevant* part of the tree, and vice-versa). Simply put, the two flows (*relevant* and *irrelevant*) are more separated. This correlates positively with results from Table 4 suggesting that the BA-CNN model is better for the first layer prediction. Indeed, the fact that information from all branches flow up and down through the attention mechanism makes the first branch receive information from the deeper layers, helping it perform better and be more coherent.
- By adding in Figure 16, the same visualisation for our SBA-CNN model, we are able to confirm that the two branches are properly separated, due to the model's structure. However, there is a confusion between classes at the third layer, where model coherence plummets compared to the other three models. This is consistent with our observations from Table **??** drawn in the previous subsection 4.3.1. Model coherence is a key element to improve in future research for this architecture to outperform the other three baselines.

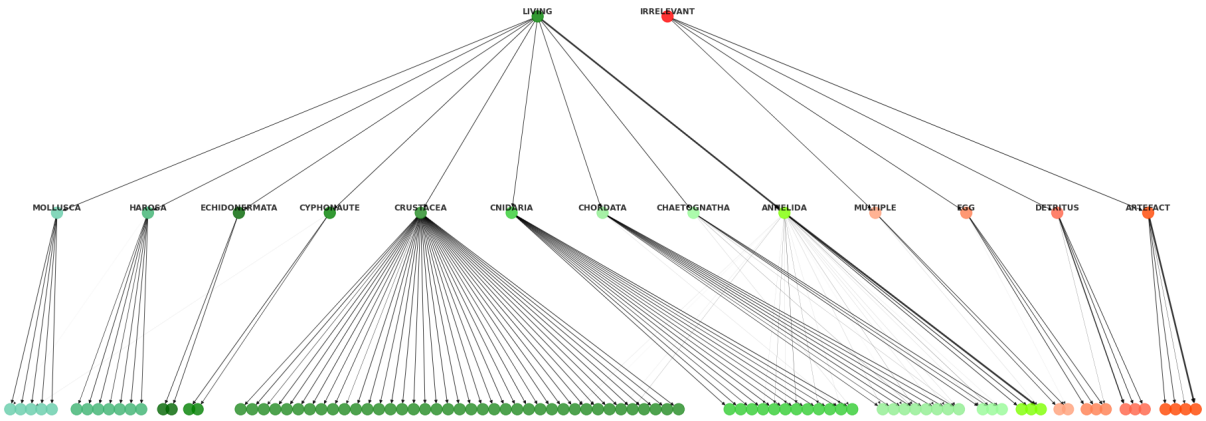
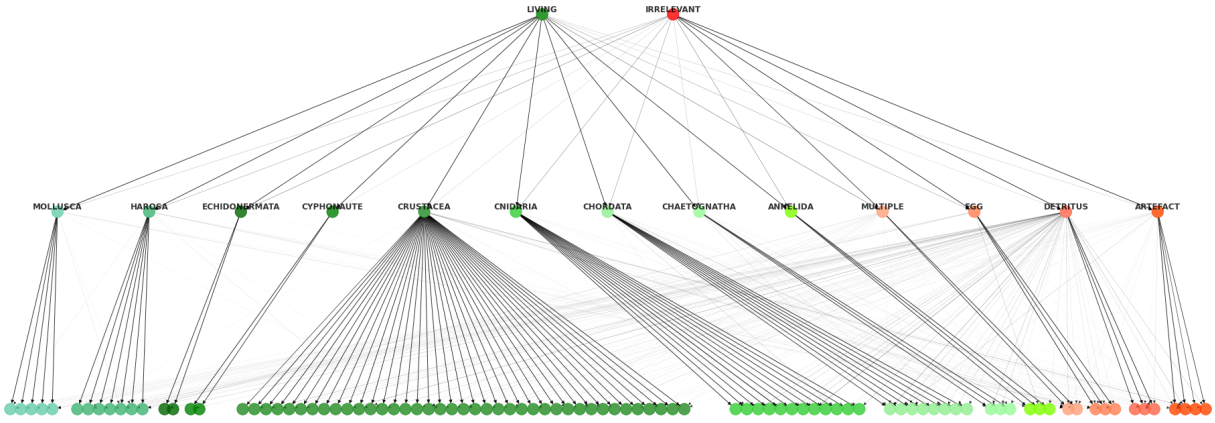
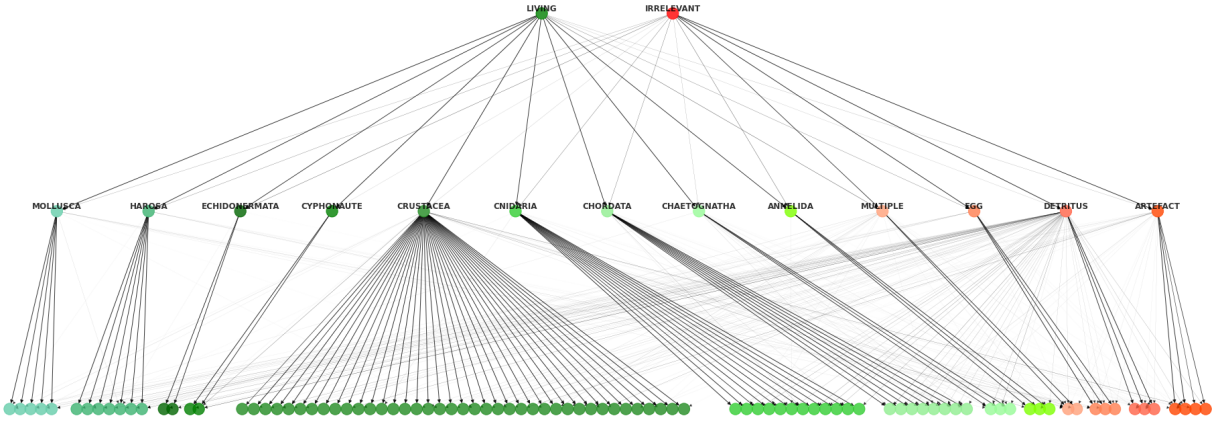


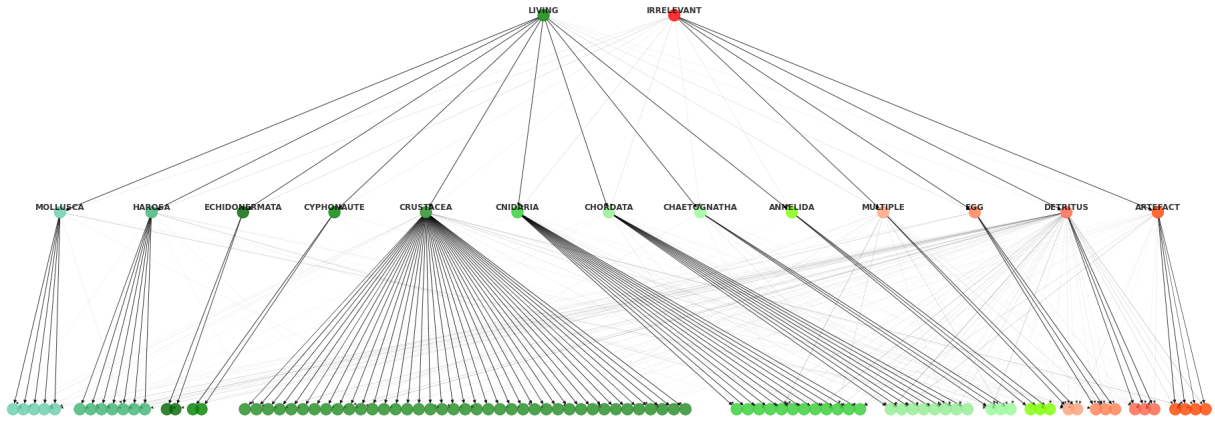
Figure 16: Effective hierarchy visualisation for the SBA-CNN model. The edges width is proportional to the relative number of samples classified.



(a) Effective hierarchy visualisation for the B-CNN model



(b) Effective hierarchy visualisation for the HB-CNN model



(c) Effective hierarchy visualisation for the BA-CNN model

Figure 17: Effective hierarchy visualisation for the three baseline architectures: B-CNN 17a, HB-CNN 17b and BA-CNN 17c. The edges width is proportional to the relative number of samples classified.

4.4 Perspectives

In this section, we successfully explored the potential of leveraging the taxonomic data available for plankton images, by building deep hierarchical models outperforming the baseline established in Section 3. The

BA-CNN architecture seems particularly efficient at solving our *Zooscan* classification problem. Although the SBA-CNN architecture did not perform as well, we believe that results from Table 6 for *relevant* samples suggest a potential for this design to achieve similar objectives.

Further research work would be necessary improve our methods and models, in order to reach a perfectly efficient and industrially usable plankton classifier. In particular, evaluating our models on different taxonomic trees and with different attention sizes would help improve performance. It could also prove useful to apply the same framework to the *WHOI* dataset, although this would require building a taxonomic tree of its plankton species from scratch.

5 Harnessing Grid5000 Cluster

The third and final section of this paper may be less experimentation and result-heavy, but it is of relevance for this research internship report since it corresponds to a central piece of my work a Inria Chile. This section relates to the handling of a French High performance computing ("HPC") cluster named *Grid5000* [3].

5.1 High Performance Computing infrastructure

Since the boom of deep learning models for image recognition tasks, computer vision models have been increasing in size and complexity - from small *LeNet-5* ($60,000$ trainable parameters) to current SOTA classification and segmentation models (such as ViT [7] or You Only Look Once ("YOLO") models [34]), often displaying tens of thousands to hundreds of thousands parameters. If the development of highly efficient hardware (Graphic Processor Units - "GPU" - and Tensor Processor Units - "TPU") and software (*CUDA* and CuDNN[17] libraries) for training acceleration made it possible to train models on small computers, large models still require immense computational resources which are rarely available to researchers or professionals. Research institutes, governments and large companies have thus started to invest into supercomputers called *clusters*, enabling researchers to develop and train large models efficiently.

Grid5000 [3] is one of them, with Inria and the French *Centre national de la recherche scientifique* ("CNRS") among key shareholders. *Grid5000* is comprised of several clusters based in various French cities and in Luxembourg (Figure 18). It hosts a large number of resources (more than 800 computers or *nodes*), some benefiting from GPU technologies.

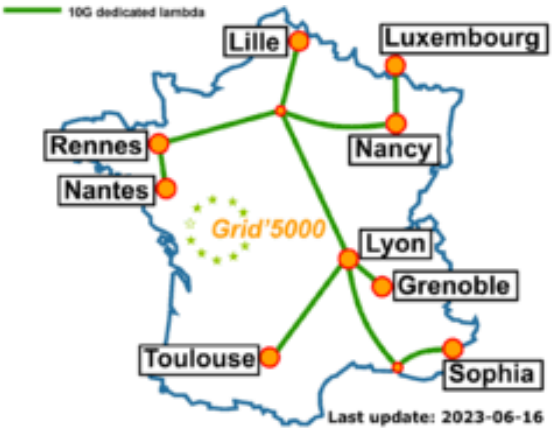


Figure 18: *Grid5000* network structure and logo (Image taken from the cluster site)

5.2 Software Development at Inria Chile

Although the offices of Inria Chile benefits from quality hardware resources, among which a local server hosting 3 GPUs, they are not close to being sufficient in the light of the work undertaken. The unprece-

[17] *CUDA* and *CuDNN* are two librairies developed by *Nvidia* in order to speed-up deep learning computations. It allows the computation to be performed by graphics cards, which scale better compared to computer processor units.

dented scale of the project combined with the outdated nature of some of the hardware made some experiments particularly challenging, if not impossible.

Inria Chile being a partner of Inria in France, researchers in theory have virtual access to Inria's hardware clusters such *Grid5000*. However, given that previous projects carried out in Chile were of smaller scale, researchers at Inria Chile did not require such infrastructure, and nobody was using it regularly or reliably.

Starting my research with no other option than the existing hardware, I decided to explore the potential of such virtual connection to *Grid5000* in order to enhance my model's computational power. I thus worked on creating a base framework to use *Grid5000* which would be as broad and general as possible, so that other researchers at Inria Chile could easily adapt it to their respective projects.

To accomplish this objective, we firstly created a repository (which can be found at this address `https://github.com/Inria-Chile/Cluster-Framework.git`) with all source code for a running pipeline. The repository includes:

- A library with the models, datamodules and helper functions;
- A setup notebook handling datasets and their download;
- An experiment setup notebook that generates on-demand *.json* files holding each experiment hyperparameters;
- A *bash script* that creates a virtual environment to handle requirements dependencies, properly configures the cluster environment variables and topology according to the available and requested resources, and launches the appropriate *train.py* script;
- Two *train.py* scripts (one for single node and one for multiple nodes jobs) launching experiments with the appropriate hyperparameters;
- A guide on how the cluster works, how to configure the *ssh* connection to the cluster sites and how to handle jobs reservations, alongside other tips and tricks for using the cluster.

Having tested and experimented the framework on all computational tasks required to produce the research in this paper, and for the entire duration of the internship, we also gave live tutorials to other researchers to walk them through the novelty and encourage them to systematise usage of *Grid5000*.

Developing such a framework was a research task in itself. For this independent project carried out at my own initiative in order to enhance both computational speed and quality of results for my work on plankton classification, I defined my own problematic, developed my models, tested them in various contexts, and generalised my solution to make it applicable to other usages. Not only did this project allowed me to run all the experiments presented in section 3 and 4 (which would not have been possible without access to *Grid5000*); it also made me discover a new field on distributed systems for deep learning training, to which I was unfamiliar prios to my internship.

Conclusion

In this paper, we comprehensively went through the several missions I have undertaken during my fascinating internship experience at Inria Chile. After explaining the importance of plankton for our environment, which ultimately motivates the need for systematic and automatic plankton recognition by marine biologists, we have reviewed the state of literature on image classification techniques and, in particular for our purposes, on hierarchical classification. This allowed us to create performing classifiers on untouched *Zooscan* and *WHOI* datasets, after an extensive hyperparameter search, thus setting a baseline for large-scale automatic plankton detection. Building on these results, we leveraged the taxonomic data available for plankton families to implement several known hierarchical models alongside a custom architecture named SBA-CNN. These models successfully outperformed our baseline, proving the effectiveness of plankton taxonomic data in our field of research, and the possibility of tackling huge amount of plankton data efficiently, thus bringing us closer to a universal plankton recognition model. I also took the opportunity to present another workstream completed for the Inria Chile's science team at my own initiative, to help them connect to and use the *Grid5000* cluster, thus unleashing available computational power for future projects.

It remains that the work we initiated is only a small step further into plankton recognition, and is far from being finalised nor optimal. In order to continue improving our models, we could broaden the search space for hyperparameters, testing other models, other architectures, and perhaps unveil a better flat classifier. Another direction would be to continue towards the hierarchical route and improve both the referenced architectures and our custom model, by finetuning the branch-training process, evaluating changes in class hierarchies (*e.g.* with additional layers, or different arrangement of classes and families), or by testing other hyper-parameter such as other models, losses or communication mechanisms between branches. Finally, in relation to the final objective of having an automatic plankton director embarked in *TaraOceans* expeditions, we could also implement an OOD detection system in our hierarchical models, to help tackle real-life OOD samples. This would help evaluate the model performance and confidence *in situ* to have a better embarked system, and it would allow the model to discover new plankton species or families, locate them in the zooplankton taxonomic tree, and adapt itself in an active learning manner.

Acknowledgment

Experiments presented in this paper were carried out using the *Grid'5000* test-bed [3], supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organisations (see https://www.grid5000.fr).

All experiments presented in this paper were carried out using *Python3*, and the deep-learning frameworks used were *Pytorch* and *Pytorch Lightning*. Pretrained models and weights, when used, were taken from the *Timm* and *Pytorch Flash* libraries. Experiments were tracked using *Weights and Biases*. Additionally, usual Python libraries were also used for scientific computing (*Numpy, Scipy*), data processing (*Panda, PIL, Scikit-Learn, Torchvision*) and data visualisation (*Matplotlib, Seaborn, python-igraph, GradCam*).

References

[1] Kushankur AU Ghosh, Colin Bellinger, Roberto Corizzo, Paula Branco, Bartosz Krawczyk, and Nathalie Japkowicz. The class imbalance problem in deep learning. *Machine Learning*, 2022.

[2] Yinon M. Bar-On and Ron Milo. The biomass composition of the oceans: A blueprint of our blue planet. *Cell*, 179(7):1451–1454, 2019.

[3] Franck Cappello, E. Caron, Michel Daydé, Frédéric Desprez, Yvon Jegou, Pascale Vicat-Blanc, Emmanuel Jeannot, Stéphane Lanteri, Julien Leduc, Nouredine Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard. Grid'5000: A large scale and highly reconfigurable grid experimental testbed. *Proceedings - IEEE/ACM International Workshop on Grid Computing*, 2005:8 pp.–, 12 2005.

[4] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[5] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018.

[6] Jialun Dai, Ruchen Wang, Haiyong Zheng, Guangrong Ji, and Xiaoyan Qiao. Zooplanktonet: Deep convolutional network for zooplankton classification. pages 1–6, 2016.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[8] Mohannad Elhamod, Kelly M. Diamond, A. Murat Maga, Yasin Bakis, Henry L. Bart Jr., Paula Mabee, Wasila Dahdul, Jeremy Leipzig, Jane Greenberg, Brian Avants, and Anuj Karpatne. Hierarchy-guided neural network for species classification. *Methods in Ecology and Evolution*, 13(3):642–652, 2022.

[9] Jalabert Laetitia 1 Olivier Marion 1 Romagnan Jean-Baptiste 1 Costa Brandao Manoela ORCID1 Lombard Fabien 1 Llopis Natalia 1 Courboulès Justine 1 Caray-Counil Louis 1 Serranito Bruno 1 Irisson Jean-Olivier ORCID1 Picheral Marc 1 Gorsky Gaby 1 Stemmann Lars ORCID1 Elineau Amanda 1, Desnos Corinne 1. Zooscannet: plankton images captured with the zooscan. *SEANOE*, 2017.

[10] D. R. Yoerger et al. Mesobot: An autonomous underwater vehicle for tracking and sampling midwater targets. *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV), Porto, Portugal*, pages 1–7, 2018.

[11] Dehong Gao, Wenjing Yang, Huiling Zhou, Yi Wei, Yi Hu, and Hao Wang. Deep hierarchical classification for category prediction in e-commerce system. *CoRR*, abs/2005.06692, 2020.

[12] Gaby Gorsky, Mark D. Ohman, Marc Picheral, Stéphane Gasparini, Lars Stemmann, Jean-Baptiste Romagnan, Alison Cawood, Stéphane Pesant, Carmen García-Comas, and Franck Prejger. Digital zooplankton image analysis using the ZooScan integrated system. *Journal of Plankton Research*, 32(3):285–303, 03 2010.

[13] Philippe Grosjean, Marc Picheral, Caroline Warembourg, and Gabriel Gorsky. Enumeration, measurement, and identification of net zooplankton samples using the ZOOSCAN digital imaging system. *ICES Journal of Marine Science*, 61(4):518–525, 2004.

[14] Grant Van Horn, Oisin Mac Aodha, Yang Song, Alexander Shepard, Hartwig Adam, Pietro Perona, and Serge J. Belongie. The inaturalist challenge 2017 dataset. *CoRR*, abs/1707.06642, 2017.

[15] Khawar Islam. Recent advances in vision transformer: A survey and outlook of recent work, 2022.

[16] Berman M. S. Poularikas A. D. Katsinis C. Melas I. Sherman-K. Bivins L. Jeffries, H. P. Automated sizing, counting and identification of zooplankton by pattern recognition. *Marine Biology*, 1984.

[17] Huy Phan Jinhua Liang and Emmanouil Benetos. Leveraging label hierarchies for few-shot everiday sound recognition. *DCASE*, 2022.

[18] Brendan Kolisnik, Isaac Hogan, and Farhana Zulkernine. Condition-cnn: A hierarchical multi-label fashion image classification model. *Expert Syst. Appl.*, 182(C), nov 2021.

[19] S Kyathanahally, Thomas Hardeman, Marta Reyes, Ewa Merz, Thea Bulas, Francesco Pomati, and Marco Baity-Jesi. Ensembles of vision transformers as a new paradigm for automated classification in ecology. *arXiv preprint arXiv:2203.01726*, 2022.

[20] Sreenath P Kyathanahally, Thomas Hardeman, Ewa Merz, Thea Bulas, Marta Reyes, Peter Isles, Francesco Pomati, and Marco Baity-Jesi. Deep learning classification of lake zooplankton. *Frontiers in Microbiology*, 12, 11 2021.

[21] Peter V.Z. Lane, Leopoldo Llinás, Sharon L. Smith, and Dora Pilz. Zooplankton distribution in the western arctic during summer 2002: Hydrographic habitats and implications for food chain dynamics. *Journal of Marine Systems*, 70(1):97–133, 2008.

[22] Sudhansu Ranjan Lenka, Sukant Kishoro Bisoy, Rojalina Priyadarshini, and Biswaranjan Nayak. Representative-based cluster undersampling technique for imbalanced credit scoring datasets. pages 119–129, 2022.

[23] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030, 2021.

[25] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *CoRR*, abs/2201.03545, 2022.

[26] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu. Large-scale long-tailed recognition in an open world. *CoRR*, abs/1904.05160, 2019.

[27] A. Lumini and L. Nanni. *Ocean Ecosystems Plankton Classification*, pages 261–280. Springer International Publishing, Cham, 2019.

[28] Abdul Arfat Mohammed and Venkatesh Umaashankar. Effectiveness of hierarchical softmax in large scale classification tasks. *CoRR*, abs/1812.05737, 2018.

[29] Hiromichi Nagasawa. The crustacean cuticle: Structure, composition and mineralization. *Frontiers in bioscience (Elite edition)*, 4:711–20, 01 2012.

[30] Eric C. Orenstein, Oscar Beijbom, Emily E. Peacock, and Heidi M. Sosik. Whoi-plankton- A large scale fine grained visual recognition benchmark dataset for plankton classification. *CoRR*, abs/1510.00745, 2015.

[31] Zimmerman Thomas G. Biswas Sujoy K. Bianco Simone Pastore, Vito P. Annotation-free learning of plankton for classification and anomaly detection. *Scientific Reports*, 2020.

[32] Iván Pizarro, Ricardo Ñanculef, and Carlos Valle. An attention-based architecture for hierarchical classification with cnns. *IEEE Access*, 11:32972–32995, 2023.

[33] Xingcheng Ran, Yue Xi, Yonggang Lu, Xiangwen Wang, and Zhenyu Lu. Comprehensive survey on hierarchical clustering algorithms and the recent developments.

[34] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[35] Yadigar Sekerci and Ramazan Ozarslan. Oxygen-plankton model under the effect of global warming with nonsingular fractional order. *Chaos, Solitons Fractals*, 132:109532, 2020.

[36] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[37] Carlos N. Silla and Alex A. Freitas. A survey of hierarchical classification across different application domains.

[38] Bandera Antonio González Martín Hernández-León Santiago Sosa-Trejo, David. Vision-based techniques for automatic marine plankton classification. *Artificial Intelligence Review*, 2023.

[39] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[40] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. *CoRR*, abs/2104.00298, 2021.

[41] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877, 2020.

[42] Pei Wang Tz-Ying Wu, Pedro Morgado, Chih-Hui Ho, and Nuno Vasconcelos. Solving long-tailed recognition with deep realistic taxonomic classifier. *CoRR*, abs/2007.09898, 2020.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[44] Elżbieta Wilk-Woźniak. An introduction to the 'micronet' of cyanobacterial harmful algal blooms (cyanohabs): cyanobacteria, zooplankton and microorganisms: a review. *Marine and Freshwater Research*, 2019.

[45] Kai Yi, Xiaoqian Shen, Yunhao Gou, and Mohamed Elhoseiny. Exploring hierarchical graph representation for large-scale zero-shot image classification. *ArXiv*, abs/2203.01386, 2022.

[46] Xiang Zhang, Lei Tang, Hangzai Luo, Sheng Zhong, Ziyu Guan, Long Chen, Chao Zhao, Jinye Peng, and Jianping Fan. Hierarchical bilinear convolutional neural network for image classification. *IET Computer Vision*, 15, 04 2021.

[47] Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *CoRR*, abs/2110.04596, 2021.

[48] Haiyong Zheng, Ruchen Wang, Zhibin Yu, Nan Wang, Zhaorui Gu, and Bing Zheng. Automatic plankton image classification combining multiple view features via multiple kernel learning. *BMC Bioinformatics*, 2017.

[49] Xinqi Zhu and Michael Bain. B-CNN: branch convolutional neural network for hierarchical classification. *CoRR*, abs/1709.09890, 2017.