Zooplankton images classification using machine learning

Abstract

This project aims to perform image classification using convolutional neural networks. The objective of this project is to improve the classification performed by the Zooscan, which sometimes contains errors, especially in the distinction between individuals and their moult. Unfortunately, the classification did not work, probably because of the quality and the low number of images.

1. Introduction

Zooplankton play a key role in the functioning of aquatic ecosystems, allowing energy from primary production to be transferred to secondary consumers. In particular, they are thought to be responsible for fish population fluctuations, algal stock regulation, as well as an important role in the carbon cycle.

In order to support these hypotheses, a fine dynamics of the Zooplankton population is necessary. These dynamics can then be compared to other data, including biotic factors, such as predation or food resources, but also to abiotic factors, such as pH, temperature, dissolved oxygen level, or carbonate content. The main drivers of these dynamics can thus be identified, as well as the underestimated roles of Zooplankton. To achieve this fine population dynamics, a high frequency analysis is necessary. Usually, counting individuals is done by microscopy, but this time consuming method limits the number of samples that can be processed. In order to overcome this limitation, a digital imaging obtained with the Zooscan can be used, which is an imaging system allowing to measure and digitize microorganisms. A semi-automatic analysis to annotate the large set of images by Deep learning can then be performed, using EcoTaxa, a web application used for taxonomic classification of plankton images. However, errors can be made when recognizing digitized images.

A large part of the zooplankton present in Lake Geneva is composed of micro-crustaceans, such as daphnids, calanoids, or cyclopoids. Like all crustaceans, the carapace of daphnids cannot grow, so they must moult during their growth. In the Zooscan, the remaining carcass of the carapace can be identified as a whole Daphnia, which can therefore bias the result. The objective of this project is therefore to make a classification between Daphnia and Daphnia carcasses, in order to improve the classification provided by the Zooscan.

2. Methods and dataset

2.1. Data collection

The dataset used here is composed of 17900 images of plankton, previously annotated in 35 classes. Two classes were used here, corresponding to daphnia (Figure 1), with 721 images, and Daphnia shells (Figure 2), with 46 images. The images were collected at Lake Grieffensee (Switzerland) in 2021. A link to the dataset is available here.

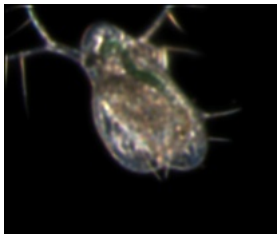


Figure 1. Example of Daphnia image from the dataset

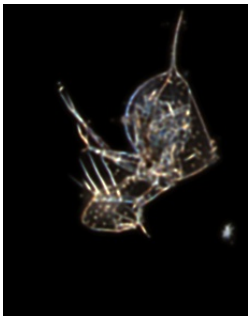


Figure 2. Example of Daphnia shell image from the dataset

Although the images do not have a high resolution, we can see here that the image of the Daphnia shell (Figure 2) appears more transparent than the one of the Daphnia (Figure 1).

The images were then split into separate folders, and split

into train-test-val with an 80-20 ratio. A summary of this distribution can be seen in the Table 1.

Table 1. Folder structure after the split

CATEGORY	TRAIN (0.8*0.8*T)	VALIDATION (0.2*0.8*T)	TEST (0.2*T)
DAPHNIA: (721)	461	115	145
SHELLS: (46)	30	7	9

The "Train" folder contains all the data used to train the model. The "Validation" folder contains the images necessary to validate the model at each time, in order to obtain the precisions and the losses of training and validation. Finally, the "Test" folder is composed of new images, to evaluate the accuracy of the model.

2.2. Methods

Architecture

VGG19 is a deep Convolutional Neuronal Network (CNN) architecture pre-trained with ImageNet database, consisting of 14,197,122 images(V.Nithyashree,2022). VGG19 composed of 19 layers, including 16 convolution layers, 5 MaxPool layers, 3 fully connected layers, and 1 SoftMax layer. A summary of the model has been produced and can be visualize in Table 2

Table 2. Summary of the VGG19 model

LAYER	OUTPUT SHAPE	PARAM
INPUT1	[(NONE, 224, 224, 3)]	0
BLOCK1-CONV1	(NONE, 224, 224, 64)	1792
BLOCK1-CONV2	(NONE, 224, 224, 64)	36928
BLOCK1-POOL	(NONE, 112, 112, 64)	0
BLOCK2-CONV1	(NONE, 112, 112, 128)	73856
BLOCK2-CONV2)	(NONE, 112, 112, 128)	147584
BLOCK2-POOL	(NONE, 56, 56, 128)	0
BLOCK3-CONV1	(NONE, 56, 56, 256)	295168
BLOCK3-CONV2	(NONE, 56, 56, 256)	590080
BLOCK3-CONV3	(NONE, 56, 56, 256)	590080
BLOCK3-CONV4)	(NONE, 56, 56, 256)	590080
BLOCK3-POOL	(NONE, 28, 28, 256)	0
BLOCK4-CONV1	(NONE, 28, 28, 512)	1180160
BLOCK4-CONV2	(NONE, 28, 28, 512)	2359808
BLOCK4-CONV3	(NONE, 28, 28, 512)	2359808
BLOCK4-CONV4	(NONE, 28, 28, 512)	2359808
BLOCK4-POOL	(NONE, 14, 14, 512)	0
BLOCK5-CONV1	(NONE, 14, 14, 512)	2359808
BLOCK5-CONV2)	(NONE, 14, 14, 512)	2359808
BLOCK5-CONV3	(NONE, 14, 14, 512)	2359808
BLOCK5-CONV4	(NONE, 14, 14, 512)	2359808
BLOCK5-POOL	(NONE, 7, 7, 512)	0
FLATTEN	(NONE, 25088)	0
DENSE (DENSE)	(NONE, 3)	75267

Image pre-processing

The images in the dataset do not all have the same dimensions. They must therefore first be resized. The input of the network contains a fixed size image of 224*224, so with a matrix of the form (224,224,3). The other images are then all resized to the size of 224*224.

Model training

The next step is to use transfer learning. This consists in loading the pre-trained model (VGG19) for image recognition from ImageNet, then freezing all layers, and finally removing the last layer and replacing it with my data of interest.

Compilation of the model

The model has been compiled to define the loss function, the optimizer and the metrics. The optimizer called "adam optimizer" was used. This optimizer itself defines the best learning rate.

Fitting of the model

In order for the model not to be overfitted during training, the early stop is used to stop training if the validation loss increases.

Evaluation of the model

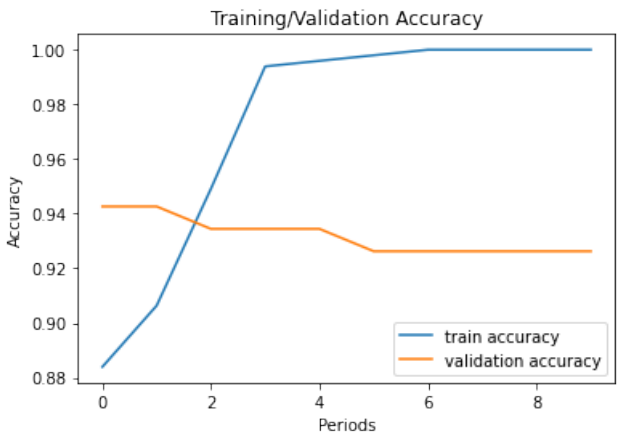
The last step is to evaluate the model. To do so, it has been tested on the test dataset

3. Results

The different results obtained are presented in this section.

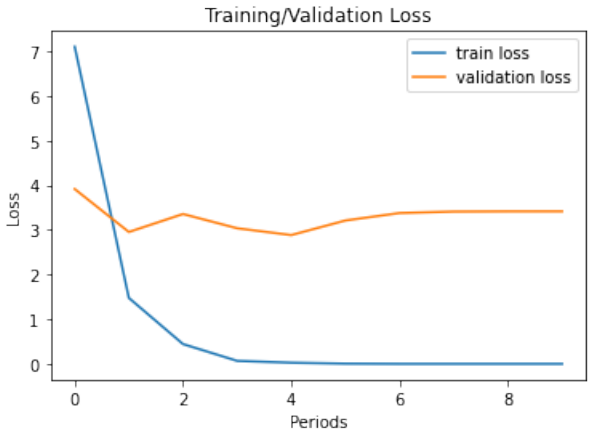
Figures 3 and 4 shows the performance of the model on the training and validation datasets, using accuracy and loss plots.

Figure 3. Training and validation accuracy)



Here we can see that the accuracy graph (Figure 3) shows

Figure 4. Training and validation loss)



that the training curve first rises and then stabilizes. On the other hand, the validation curve decreases only slightly, then remains flat. A similar dynamic can be observed on the loss graph (Figure 4), where the training curve first decreases, then stabilizes at a value close to 0. The validation curve remains relatively flat.

The first model evaluation shows the following results:

- Speed : 94s, 19s/step
- Loss : 2.625492572784424
- Accuracy : 0.9415584206581116.

The model therefore shows an accuracy of 94%, with a test time of 94 seconds.

A classification report Table 3 and a confusion matrix Table 3 were also produced, in order to visualize the quality of the model. We can see that the 145 images of Daphnia have been effectively classified as Daphnia, but that the shells have also been identified as Daphnia.

In Table 4, the class of 0 corresponds to Daphnia, and the class of 1 corresponds to shells

	DAPHNIA	SHELLS
DAPHNIA	145	9
SHELLS	0	0

Table 3. Confusion matrix

4. Discussion

On the precision graph, we can observe a large gap between the training and validation curve. This large gap indicates overfitting. The training curve follows a trajectory that seems right, but the overfitting is clear when looking at the validation curve.

CLASS	PRECISION	RECALL	F1-SCORE	SUPPORT
0	1.00	0.94	0.97	154
1	0.00	0.00	0.00	0
ACCURACY			0.94	154
MACRO AVG	0.50	0.47	0.48	154
WEIGHTED AVG	1.00	0.94	0.97	154

Table 4. Classification report

Concerning the loss curve, we see that there is a good learning rate, but when we look at the validation curve, we see again that there is an overfitting.

Also, the f1-score of 0 for the shell class shows that there is a problem in the code.

5. Conclusion

Unfortunately, the classification did not work. This is probably due to the poor quality and low number of images, mainly concerning the shells

6. References

Data for: Deep Learning Classification of Lake Zooplankton - ERIC-open. (s. d.). Consulté 27 mai 2022, à l'adresse https://opendata.eawag.ch/dataset/deep-learning-classification-of-zooplankton-from-lakes

Nithyashree, V. (2021, juillet 19). Image classification — Step-by-Step guide for Image Classification. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/07/step-by-step-guide-for-image-classification-on-custom-datasets/

Nithyashree, V. (2022). VGG-19-for-Rock-Paper-and-Scissors-classification [Jupyter Notebook]. https://github.com/Nithyashree-2022/VGG-19-for-Rock-Paper-and-Scissors-classification (Original work published 2021)

7. Python code used to train the model

https://github.com/JerKeller/2022_ML_Earth_Env_Sci/blob/09aecb52a71e60991207ee578e6c15ae7856ded5/Daphnia_Class.ipynb