On the use of morphological filtering to improve ConvNets predictions

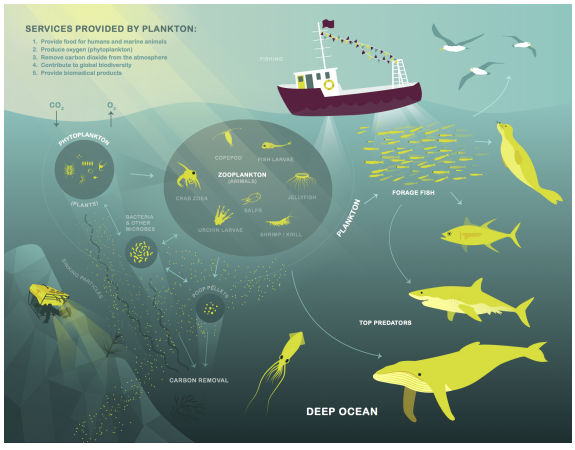
Simon Cazals
Stanford University
Department of Statistics
scazals@stanford.edu

Abstract

Planktons have crucial role in the global carbon cycle and aquatic food webs. For this reasons, their population levels an ideal measure of the health of world?s oceans. However, manual analysis of the underwater imagery is very time consuming and we need to find an alternative to this manual approach. The National Data Science Bowl is a Kaggle competition which challenges us to build an machine learning algorithm to automate the image identification process.

1. Introduction

Planktons are critically important to our ecosystem, accounting for more than half of the primary productivity on earth and nearly half the otal carbon fixed in the global carbon cycle. They also form the foundation of aquatic food webs including those of large, important fisheries. Loss of plankton populations could result in ecological upheaval as well as negative societal impacts, particularly in indigenous cultures and the developing world. Plankton?s global significance makes their population levels an ideal measure of the health of the world?s oceans and ecosystems.



We use an underwater camera system to capture microscopic images over large study areas. In total, Oregon State University's Hatfield Marine Science Center has captured nearly 50 million plankton images over an 18-day period. Manual analysis is unfeasible on this kind of dataset, and we have to find an alternative to this manual approach.

The National Data Science Bowl is a competition organized by Hatfield Marine Science Center and the website Kaggle, that challenges us to build a machine learning algorithm to automate the image identification process.

First, we will train simple ConvNets on the training set, and we will then try to use supplementary information to improve our prediction, using morphological filtering and size considerations.

2. Approach

2.1. Datasets and Evaluation

The training set is composed of 30336 pictures of plankton divided into 121 classes. Some classes have a very small number of images, like the *hydromedusae solmundella* (**9 pictures**), the *shrimp zoea* (**10**) or *siphonophore calycophoran rocketship adult* (**10**). 33 classes have less than 50 pictures. By contrast, 4

classes have more than 1000 images, two of them have more than 1900 pictures : *invertebrate larvae other B* (**1934**) and *appendicularian slight curve* (**1979**).

The testing set is composed of 130400 images that we must classify.

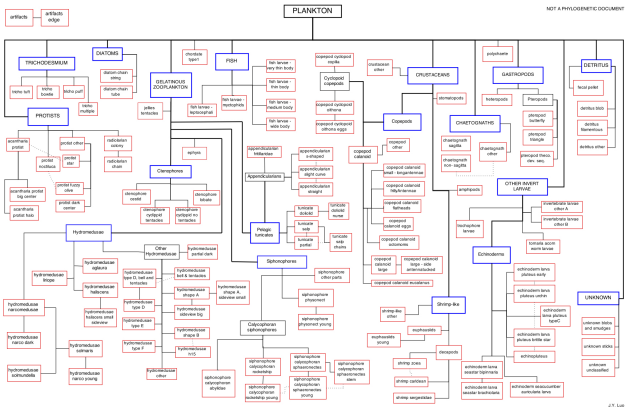
The submissions are evaluated using the multi-class logarithmic loss. Each image has been labeled with one true class. For each image, me must give a set of predicted probabilities (one for every class). The formula is then :

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{121} y_{ij} log(p_{ij})$$

with y_{ij} is 1 if observation i is in class j and 0 otherwise.

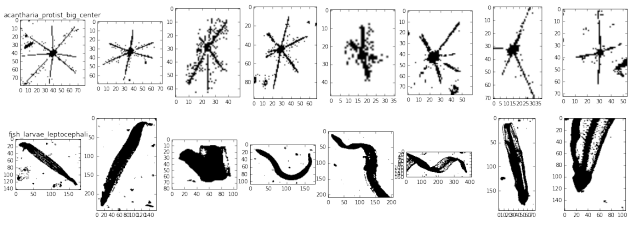
The submitted probabilities for a given image are not required to sum to one because they are rescaled prior to being scored (each row is divided by the row sum). In order to avoid the extremes of the log function, predicted probabilities are replaced with $max(min(p, 1-10^{-15}), 10^{-15})$. The purpose of using this score function is to heavily penalize a very low probability for the real class.

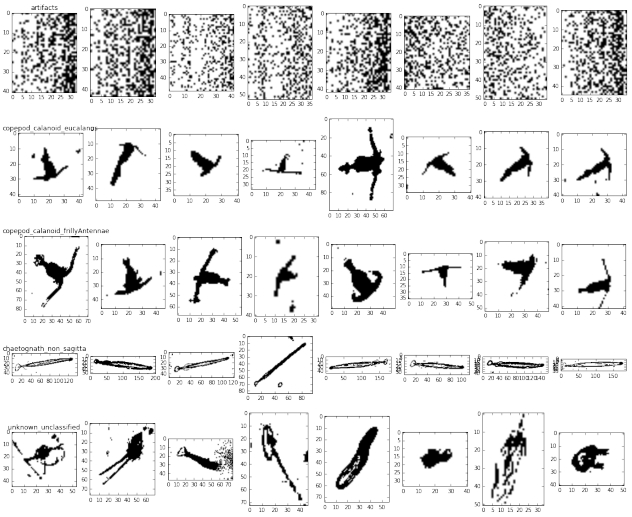
Here is provided a rough guide to understand relationships between the 121 classes. The tree-like diagram indicates morphological and biological connections between groups. Dashed lines indicate a weak(er) relationship and solid lines a stronger relationship.



2.2. Insight of the data

First, we must have a look at some of the classes in the dataset :





Remarks from the visualization of those classes :

• Some classes have very specific pattern, like the *acantharia portist big center* (the first), the *artifacts* (3rd), or the *chaetognath non sagitta* (6th).

• Some classes can have different shapes because of the 2D pictures of a 3D objects. Like the fish larvae leptocephali (2nd), which must make them harder to classify.

• Some different classes are very similar. For example, the copepod cloned eucalanus, copepod cloned frilly Antennae (4th and 5th). Because those classes are very similar, it make the classification harder.

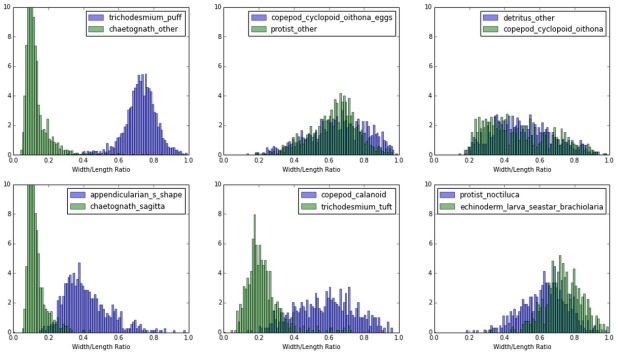
• The unknown classified class composed of very different objects, which is very hard to classify.

Other points make the classification very hard, the most obvious are :

• Some classes have a very very small number of pictures, 3 classes have less than 10 images, and 33 classes have less than 50, while other classes have nearly 2000 pictures... It's seems pretty hard to train a neural network with 10 pictures in a class.

• Some images are so noisy and ambiguous that experts have a difficult time labeling them. Noise in the ground truth in inevitable.

However, when we look at the ratio between the length and the width axis, we realize that the distributions of classes for this value are very different (as illustrated on the graph on the next page). This lead us to consider adding some global features that could help us to determine the class

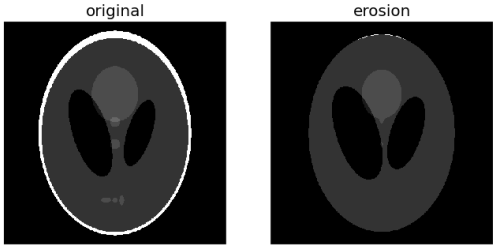


2.3. Morphological Filtering

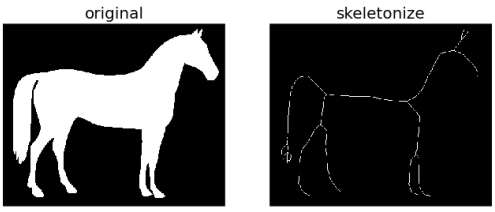
Morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image, such as boundaries, skeletons, etc. Our goal is to use this transformation, to obtain global information of the image. In fact, the ConvNets find some local properties on shape, but we could try to add some global component, like the number of pixel, the fact that the shape is convex or not, etc... And morphological filtering from sickit-image can help us for that.

Before doing this transformation, we decide to threshold the image and to select the principal shape, because sometimes, we have noise around the shape that we want to study.
Here are the morphological filterings than we will use :

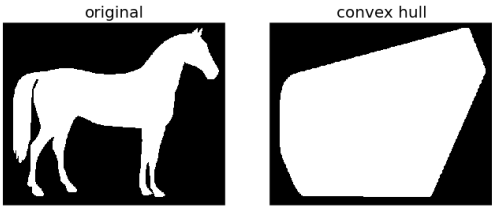
• Morphological **erosion** sets a pixel at (i, j) to the minimum over all pixels in the neighborhood centered at (i, j).



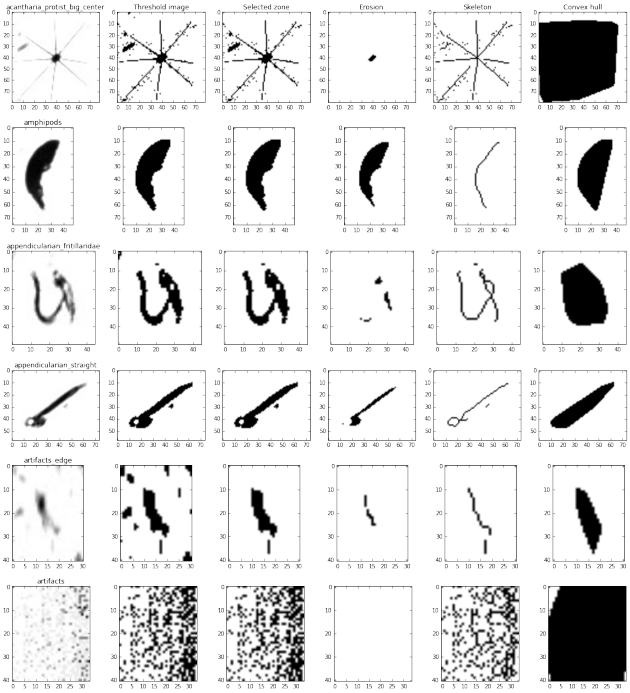
• **Thinning** is used to reduce each connected component in a binary image to a single-pixel wide **skeleton**. This can only be performed on binary images, which explain why we threshold images.



• The **convex hull** of an image is the set of pixels included in the smallest convex polygon that surround all white pixels in the input image. Again note that this is also performed on binary images.



We apply these operations to some classes, to see the impact on the images :



We notice that those operations have different impact on the classes :
• For some thin shape like the *acantharia protist big center*,

or the *appendicularian fritillaridae*, the erosion almost kill the plankton. For some compact shape like *amphipods*, the erosion does not change the shape that much. We notice also that erosion completely kill *artifacts*.

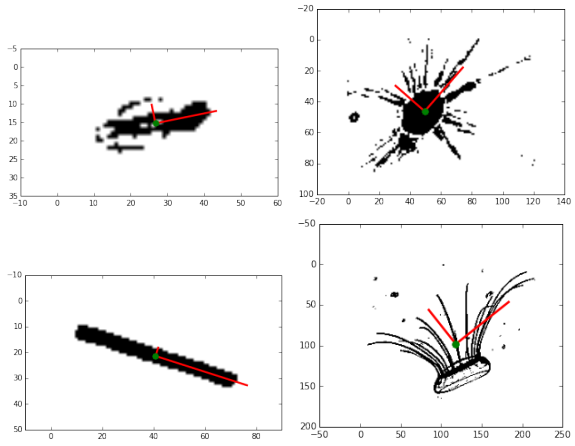
• Some skeletonize shape are very different from the original image, this is the case for compact images like the *amphipods*, while for thin shape (*acantharia protist big center* for example), it does not really change.

• The convex hull are adding a lot of pixels for plankton that take a large surface but with a thin structure. Typically, the *acantharia protist big center* have a huge difference of pixel between its convex hull and the original image, while for compact shape like *amphipods*, it does not change that much.

2.4. Other global features

An other approach of analyzing the shape of an image, is to consider the smallest ellipse that contain the shape. For example, the ratio between axis is close to 1 for a shape that has a circular general shape, where as a thin shape like the *appendicularian straight* will have a ratio between the large and the small axis very large. For this reason, we may change or probability given this features obtained before rescaling the image.

Here are some examples :



We can see here the axis for 4 classes, as expected, we see the important difference of length of axis for the thin shape, while they have similar length for a circular shape. Other features that we may consider are the eccentricity of the ellipse , or its area.

$$e = \sqrt{1 - \frac{b^2}{a^2}}$$

$$A = \pi ab$$

3. Experiment

3.1. Results with random forest with morphological filtering

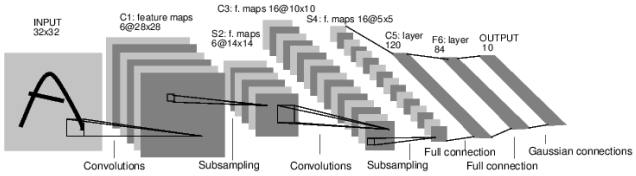
In the tutorial proposed by Kaggle, we build a classifier with a random forrest taking as features each of the pixel from the rescaled image, and a size (length and width) of the image before rescaling. This techniques give a score function of 3.72 which is already way better than the score obtained with the uniform distribution 4.79.

Into the features on which we build our random forrest classifier, we can features obtain from the morphological filtering discussed previously :

- The x and y axis from the ellipse
- The ratio x/y from the ellipse
- The number of black pixel in the eroded, the skeleton and the convex hull versions of each image
- The ratio between the previous number and the number of black pixel in the original image

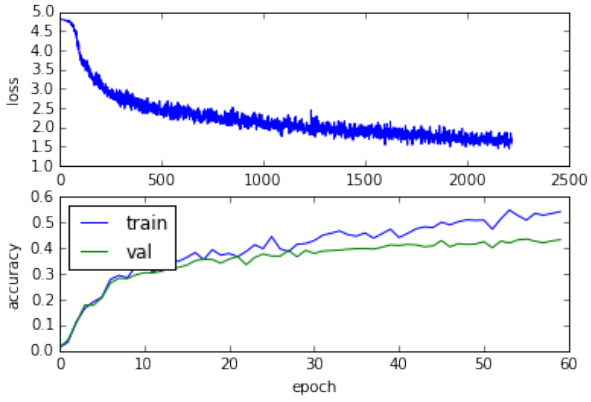
Adding all this features in the predictors in the random forest we obtain a model that give a score on the test set of 2.01 which is a big improvement from the original random forrest. We will try to do this analogy and to use the global features obtained to improve our prediction from a ConvNets.

3.2. Results with ConvNets



We mainly used two models of convolutional neural networks.

The first model is a 3-layer ConvNet with the following architecture : **conv - relu - pool - affine - relu - dropout - affine - softmax**. We train this model on reshaped 48x48 images, with a filter size of 3, and 64 & 128 filters. This model is training pretty fast (few hours), we obtain a misclassification ratio of about 46% (on 121 classes) and **a score close to 1.8 on the test set** but training the model longer does not improve our prediction a lot.



The second model is a more complex architecture, publicly posted by a participant : **conv - relu - pool - conv - relu - con - relu - pool - affine - relu - dropout - affine - relu - dropout - affine - softmax**, with 60x60 images, same score ($\sim$ 1.8) with 10 epochs in $\sim$**24 hours** on a CPU. With a GPU, participants declare that we can obtain a **score of 0.9 with 50 epochs, in about 1 hour.**

These methods are simple methods (simple model in the first case, and few iterations of a more complex model in the second one), but are already beating the prediction made by the random forest.

3.3. Adding global components

Since we obtained a matrix of probabilities from our model with ConvNets, we now try to add some features to improve our score without training a new model. We explain the process to use one feature with the ratio between the small and the large axis of the ellipse.
First, we compute the mean of this ratio for each class. We then consider an image from the testing set, and the probability vector obtained from ConvNets. We compute the ratio for this image. Now the problem is too find a way to penalize the class that have a very different mean ratio than the ratio for this image. Our hypothesis is that we must penalize equally when the ratio is 2 times greater than the mean ratio of one class, and when it is half of the mean ratio of another class. For that we use the function :

$$\phi(r) = \frac{r}{1+r^2}$$

This function is maximal for $x = 1$ and has the same value for any x and $1/x$. We then multiply the probability of the class, by the $\phi(r)$ where r is the ratio divided by the mean of the ratio for the corresponding class. This way, the closer the ratio is to the mean of the ratio in one class, the greater in the new probability of appartenance. Then, we scale the new vector to get a vector of probability (with $\sum p_i = 1$).
Here we used the ratio between axis, but we can replace that by any feature. However, the function ϕ may penalize the probability too much or not enough, so we have to tune the weight of the penalization of ϕ. For that, we can simply, use $\phi + C$ where C is a constant, this should reduce the penalization.
As a basis, we used the results that we obtain from the first model, with a score of 1.789134. Our expectation about the behaviors of the score function, depending on C, is that if C is big, we pretty much destroy the effect of ϕ, and we obtain the initial probability that we obtained from ConvNets. However the question is too know if the score is lower than the initial score with any C.

We've tried without adding the penalization term C, and we obtain a very bad result, a score of 3.03. Adding the C, the score is getting closer to the original score, but we tried many value and the score is continuously going from 3.03 to the initial score, without getting under the initial score 1.8.

We observe the same behaviors when we add the other features.

4. Conclusion

In conclusion, it it strange to notice that modifying the prediction matrix generated by ConvNets, using features extracted from morphological transformations of the image does not improve the score function, since this features improved a lot the prediction with random forest. We should try to add this features directly into the ConvNets, but the goal of my project was to add some information after training the ConvNets.

I was expecting some improvement in my results since ConvNets are acting locally and my new features had global properties, but it does not seem to work, the convolutional properties may propagate local property, which may leads to global aspects that must be more efficient than my simple computations.

Even if this approach did not improved the results as expected, I learnt a lot about image filtering. I would like to thank all the teaching team, I came in this class because I was curious about CNN but I had no idea about what it was. The class was amazingly well organized, we could not guess that it was the first year this course was taught, and I am pretty sure that everybody enrolled in this class loved it.

[UPDATE before submitting] : I have just found that somebody released a code that leads to a score of 0.77, based, according to him, on some "basic Morphological operations like tophat and bottomhat". This involves the addition of an extra convolutional layer and enhancing the grayscale images with two extra channels. This extra-layer leads to

a longer computational time, approximately 4 hours on a GRID K520 GPU.

5. References

Bibliography

- Public Start Guide of Deep Network with a score of 1.382285, Bing Xu, http://www.kaggle.com/c/datasciencebowl/forums/t/11279/
- My solution for the Galaxy Zoo Challenge, Sander Dieleman, http://benanne.github.io/
- Morphological Filtering with scikit-learn, http://scikit-image.org/
- MathWorks tutorial on morphological processing, http://www.mathworks.com
- Code sample for score of 0.93, Srikrishna Sridhar, http://www.kaggle.com/c/datasciencebowl/forums/t/12750/
- Implementation for 0.775 score with Basic morphology enhancement, Unas, http://www.kaggle.com/c/datasciencebowl/forums/t/12816/

Tools and libraries

- **CXXNET** to build the more complex ConvNet in the second model : https://github.com/antinucleon/cxxnet
- **Numpy, pandas, pylab, scipy**
- **skimage** for the morphological filtering : http://scikit-image.org/

6. Annexe

My code is in 2 notebooks in my dropbox :

- Notebook_cnn is the code for the 3-layer convolutional network used to train my 1st model.
- Notebook_morph is the code where I tried to use the morphological filtering to improve my results