Predicting ocean health One plankton at a time

Abhilash Kumar (12014)
Peeyush Agarwal (12475)

April 18, 2015

Indian Institute of Technology, Kanpur
Kanpur 208016

Contents

Abstract

This project is an attempt to apply state of the art machine learning algorithms in computer vision on the plankton dataset. More specifically, we have used convolutional neural networks for image classification as they have shown excellent performance on some of the most difficult computer vision tasks. We experimented with two different convnet architectures. The first one is inspired by Hinton's work and the second one is inspired by Bengio's work. The winning team in Data Science Bowl Competition, which required participants to work on the same dataset, also used convnets to achieve the best classification results.

1 Introduction

The objective of the project was build an algorithm to automate the plankton image identification process. So, given an image, we had to classify it into one of the 121 plankton classes.

2 Motivation

Plankton are vitally important to our ecosystem. They represent the bottom few levels of a food chain. They play an important role in the bio geochemical cycles of many important chemical elements, including the ocean's carbon cycle. 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 [1].

Traditional methods for measuring and monitoring plankton populations are time consuming and cannot scale to the granularity or scope necessary for large-scale studies. A better approach is to use underwater imagery sensors for capturing microscopic, high-resolution images over large study areas. These images can then be analyzed to assess species populations and distributions.

Manual analysis of the imagery is infeasible as it could take a year or more to manually analyze the imagery volume captured in a single day. Automated image classification using machine learning tools will allow analysis at speeds and scales previously thought impossible.

3 Dataset

We have used the plankton image dataset provided as a part of Data Science Bowl Competition.

The images in the dataset were captured using ISIIS and were processed by a segmentation algorithm to isolate individual organisms. They were then labelled by a trained team at the Hatfield Marine Science Center. Approximately 30,000 of these images were provided as a training set. A test test of approximately 130,000 images was also provided for the competition.

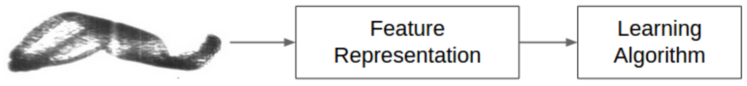
4 Challenges

As mentioned on the Data Science Bowl competition page[1], several characteristics of this problem make this classification difficult:

1. There are many different species, ranging from the smallest single-celled protists to copepods, larval fish, and larger jellies.
2. Representatives from each taxon can have any orientation within 3-D space.
3. The ocean is replete with detritus (often decomposing plant or animal matter that scientists like to call "whale snot") and fecal pellets that have no taxonomic identification but are important in other marine processes.
4. Some images are so noisy or ambiguous that experts have a difficult time labeling them. Some amount of noise in the ground truth is thus inevitable.
5. The presence of "unknown" classes require models to handle the special cases of unidentifiable objects.

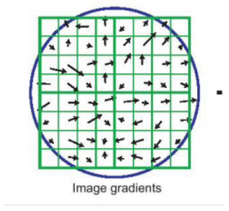
5 Previous Work and Approaches

While applying machine learning to image recognition, the image is first processed and converted to a feature representation using various feature extraction techniques. After that, various multi class classification algorithms can be applied on the extracted features to train a model.

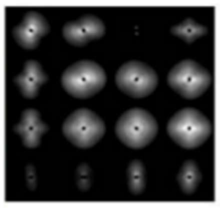


Common feature extraction methods include:

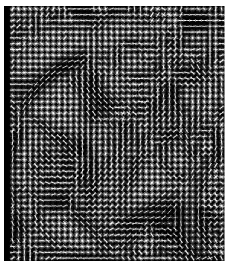
- Features for vision like Scale Invariant Feature Transform (SIFT), Gist and Histogram of Oriented Gradients (HOG) etc.



SIFT [3]



GIST [4]



HOG [5]

- Domain dependent hand crafted features
- Feature Learning Approaches like Convolutional Neural Networks that learn features from the image

Some of the commonly used algorithms for multi class classification are

- SVM
- SoftMax

Deep learning approaches have shown good results for computer vision and object recognition on a variety of datasets. These approaches are inspired by the "One Learning Algorithm" hypothesis according to which any part of the brain is capable of learning the functionality of any other part of the brain.

One of the first successful implementation of Deep Learning for image recognition was shown by Hinton and others in their paper "ImageNet Classification with Deep Convolutional Neural Networks" [10]. They showed an improvement of around 8% on the ImageNet dataset [11] over the previous best results. Since then, deep learning has been put to use successfully on a variety of image datasets including MNIST, CIFAR-10, CIFAR-100, SVHN and ImageNet. They have outperformed the traditional methods by large amounts for a variety of computer vision tasks.

This inspired us to use Convolutional Neural Networks for plankton image classification.

6 Methodology

We preprocessed the data to make it more suitable for our use case. Then, we performed data augmentation to increase the size of dataset artificially. As the first step, we applied a Random Forest approach with hand crafted feature and image pixels as training features. After that, we implemented a Convolutional Neural Network and a Dropout Network for image classification. The methods are described in more detail below.

6.1 Data Preprocessing

Data preprocessing was essential before training a model as the images had varying sizes and therefore they could not be used directly to learn a model. Also, we had to scale down the high resolution images since training with large images was infeasible. Also, some plankton classes had very few labelled images and so data augmentation was required to generate sufficient data for these classes.

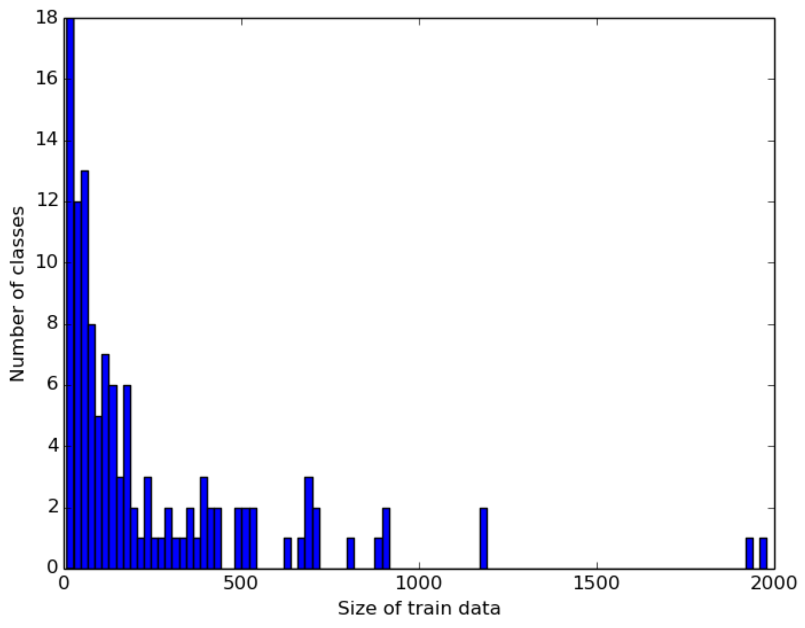


Figure 1: Data size among various classes

- We added padding to images thus converting them to a square shape. This helped in preserving the width to height ratio. It also helped prevent possible irregularities in plankton shapes which may arise during rotation and resizing of images.

- We scaled down the size of all images (25x25 for random forest and 48x48 for convnets) as training a model with larger images would have been infeasible.

- We added artificial data by performing data augmentation since some classes had very few training images (Figure 1). For data augmentation, images were rotated by random angles to produce new images. Apart from creating more data, is also helped decrease model overfitting at the cost of longer training period.

6.2 Random Forest

We observed that different plankton classes differed a lot on the basis of their shape. In particular, we found that while some classes were easily separable, some other classes had very similar length to width ratio.

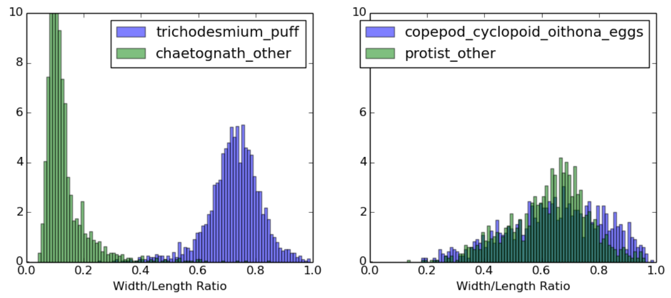


Figure 2: Class Separation based on width to height ratio

Given an image, the algorithm for finding the ratio first identifies the region where plankton is present. It then extracts this glob from the image and finds the axis along which the plankton is aligned. The major axis is taken in the direction of maximum variation and minor axis is taken perpendicular to it. These axis are then used to calculate the length to width ratio.

For the Random Forest algorithm, the images were first resized to 25x25. The 625 pixel values and length to width ratio was used for training the algorithm. The number of trees was set to 100. It did not perform well and the logloss value obtained by it was 3.72 which was only a minor improvement over the baseline solution of 4.79.

Our Random Forest code is a modified and tweaked version of the benchmark code [2] provided by Kaggle.

6.3 Convolutional Neural Network

Conv1	Conv2	Conv3	Conv4	Conv5	Full6	Full7	Full8
4x4x48	3x3x96	3x3x96	2x2x128	3x3x128	256	256	121
Stride 1	Stride 1	Stride 1	Stride 1	Stride 1	Dropout	Dropout	SoftMax
Pad 2	Pad 1	Pad 1					
ReLU	ReLU	ReLU	ReLU				

Figure 3: CNN Architecture

6.3.1 Network Description

The network used by us is similar to Hinton's ImageNet architecture[11]. It contains 8 weight layers (5 convolutional and 3 fully connected layers). The output of last fully-connected layer is fed to 121 way softmax thus producing a probabilistic distribution over the class labels.

We used Rectified Linear Unit (ReLU) activation function because networks with ReLUs consistently learn several times faster than equivalent networks with saturating neurons as shown in "ImageNet Classification with Deep Convolutional Neural Network"[11]. The authors showed that a four layer convolutional neural network with ReLUs reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons.

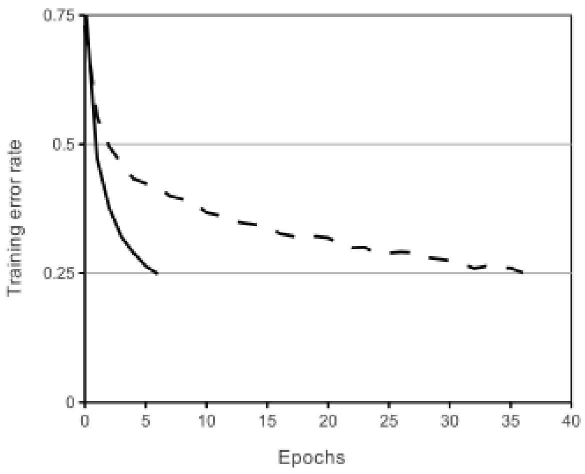


Figure 4: Training error rate with ReLU(solid line) and tanh(dashed line) neurons [11]

We also added Dropout in two layers to prevent overfitting and learn more robust features. While training, dropout removes neurons from the network with a probability of 0.5. These "dropped" neurons do not participate in forward pass and backpropagation. Thus, different architecture is presented to different input images thus reducing complex co-adaptations since neurons can no longer rely on each other's presence.

6.3.2 Training

The network is fed scaled down images of size 48x48. We used offline data augmentation for reasons highlighted in the data preprocessing subsection. The training time on Amazon Web Services (AWS) instance with NVIDIA GPU (having 1536 CUDA cores) was 20 minutes when run for 45 epochs. Our convnet code is a modified version of the examples provided by CXXNET[6].

We stopped training after 45 epochs since we stopped observing accuracy improvements with each epoch. The graph for training error started to flatten and the network became more prone to overfitting.

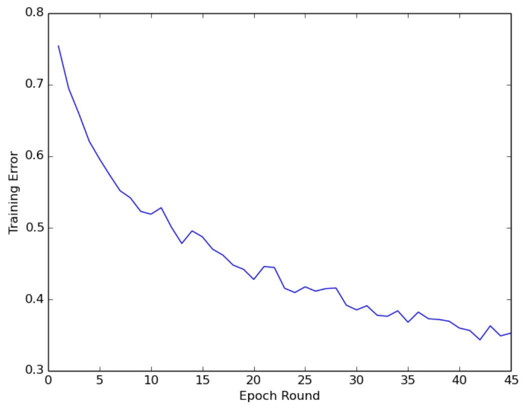


Figure 5: Training error v/s number of epochs

6.4 Maxout Network

Conv1	Conv2	Conv3	Conv4	Conv5	Conv6	Full7	Full8	Full9
4x4x32	4x4x64	3x3x128	3x3x128	3x3x256	3x3x256	1024	2048	121
Stride 2	Stride 2	Stride 2	Stride 3	Stride 2	Stride 2	Maxout	Maxout	SoftMax
Pad 0	Pad 3	Pad 3	Pad 3	Pad 2	Pad 2			

Figure 6: Maxout Network Architecture

6.4.1 Network Description

The network used by us is similar to Bengio's Dropout Network [12]. It contains 9 weight layers (6 convolutional and 3 fully connected layers).

It uses Maxout activation function which has been shown to leverage dropout more effectively[12]. The output of a maxout unit which is the max of a set of inputs can approximate arbitrary convex functions.

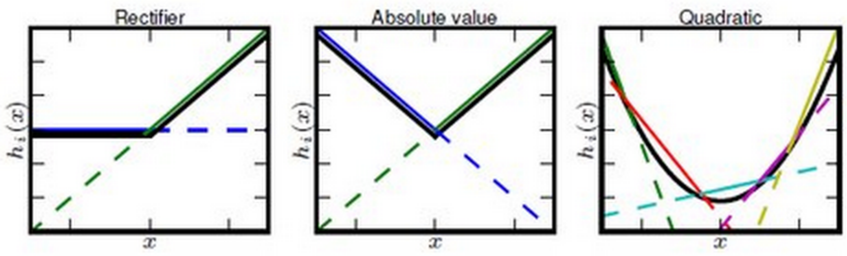


Figure 7: Graphical depiction of how maxout activation function can implement ReLU, absolute value rectifier, and approximate the quadratic activation function [12]

6.4.2 Training

The network is fed scaled down images of size 48x48. We also used real time data augmentation other than offline data augmentation. In real time data augmentation, the image is transformed slightly (rotation, translation etc.) before being fed to the network for training. Thus, the network sees a different image each time while training thus reducing overfitting. The training time on Amazon Web Services (AWS) instance with NVIDIA GPU (having 1536 CUDA cores) was 9 hours when run for 40 epochs. Our convnet code is a modified version of dkaylor's code written using pylearn2 for the same competition [7].

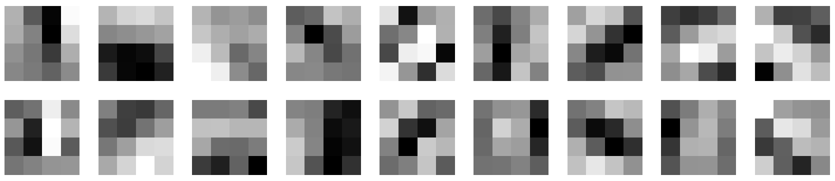


Figure 8: Visualization of weights learned in Convolutional Layer

The figure above shows some of the weights learned in the Convolutional Layer of the Dropout Network. On observing closely, it appears that the network is able to learn concepts like edge detectors, corner detectors etc. in addition to plankton specific filters.

7 Results

Being a classification problem with 121 different classes, the task of image classification was not easy. Also, accuracy is not a good measure of the effectiveness of a method. This is because as shown in Fig 1, some classes have large number of images compared to other classes and hence predicting more frequently occurring classes will give good cross validation accuracies.

A better performance measure is the multi-class logarithmic loss function [1] which heavily penalizes if the algorithm wrongly predict one class with either very high or very low probability.

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} Y_{ij} \log p_{ij}$$

Where N is the number of images in the test set (or cross validation set) , M is the number of class labels, log is the natural logarithm, y_{ij} is 1 if observation i is in class j and 0 otherwise, and p_{ij} is the predicted probability that observation i belongs to class j. Hence, a lower logloss value means better accuracy.

Method	**Accuracy**	**Logloss Error**
Random Forest	44%	3.721
CNN + Dropout	65%	1.303
Maxout	74%	0.687
Winning Team	81%	0.566

Table 1: Accuracy and logloss error for various methods

All evidences seem to suggest that training a deeper network with more data could improve our accuracy even further. The winning team in this contest obtained a classification accuracy of 81.52%. However, their winning model took 70 hours to train on an approximately 3 times more powerful GPU. Training on such large scales was infeasible for us.

References

[1] *Data Science Bowl Challenge*
`http://www.datasciencebowl.com/`
`www.kaggle.com/c/datasciencebowl`

[2] *Data Science Bowl Challenge*
`www.kaggle.com/c/datasciencebowl/details/tutorial`

[3] *Tombone's computer vision blog*
`http://quantombone.blogspot.in/2015/01/from-feature-descriptors-to-deep.html`

[4] *Andrew Ng's deep learning talk*
`http://cs229.stanford.edu/materials/CS229-DeepLearning.pdf`

[5] *Visualizing Object Detection Features*
`http://web.mit.edu/vondrick/ihog/`

[6] *CXXNET a fast, concise, distributed deep learning framework*
`https://github.com/dmlc/cxxnet`

[7] *Using dropout in pylearn2*
`https://github.com/dkaylor/datasciencebowl`

[8] *Theano, a Python library for fast and efficient evaluation of mathematical expressions involving multi-dimensional arrays over GPUs*
`http://deeplearning.net/software/theano/`

[9] Ciresan, Meier, Masci, Gambardella, Schmidhuber *Flexible, High Performance Convolutional Neural Networks for Image Classification.* IJCAI Proceedings-International Joint Conference on Artificial Intelligence. Vol. 22. No. 1. 2011.

[10] Lecun Y. , Bottou L. , Bengio Y. , Haffner P. *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 86(11),2278 - 2324,1998

[11] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. *Imagenet classification with deep convolutional neural networks.* Advances in neural information processing systems. 2012.

[12] Goodfellow, Ian J and Warde-Farley, David and Mirza, Mehdi and Courville, Aaron and Bengio, Yoshua, *Maxout networks* preprint 1302.4389,2013