CS231N Course Project Report
Classifying Shadowgraph Images of Planktons Using Convolutional Neural Networks

Shane Soh
Stanford University
shanesoh@stanford.edu

1. Introduction

In this final project report, we will document our solution to the National Data Science Bowl hosted by Kaggle [5]. The Data Science Bowl is an image classification task that involves classifying shadowgraph images of planktons into their various taxonomic classes.

Our solution uses convolutional neural networks and achieved a logloss score of **0.885346** and a position of 165 out of 1049 teams at this time of writing.

1.1. Motivation

Planktons are critical to the health of marine ecosystems. They form the foundation of aquatic food webs and the loss of plankton populations often result in devastating ecological consequences. Measuring plankton population levels around the world allows us to better understand the health of our worlds oceans.

A current method of measuring plankton populations involves towing an underwater camera system that captures microscopic images over large study areas. However, manual analysis of these images is infeasible as it would take more than a year to manually label the images captured in a single day. Therefore, there is a great need to automate the analysis of these images.

1.2. Related Work

Current leading results in the area of plankton classification uses support vector machines [1], shallow neural nets [2] and random forest-based approaches [3] that have 67% - 83% classification accuracies. This, along with one of the earliest plankton imaging systems - the Video Plankton Recorder (VPR) [4], remains among the most widely used approaches for plankton imaging today. However, there have been little to no documented attempts of using deep learning for automated plankton classification to date. This Kaggle competition also sets a precedent by having one of the largest labeled dataset of plankton shadowgraph images. These reasons in turn provide a great motivation for using convolutional neural networks (which have demonstrated high performances in various visual recognition tasks) in this domain of automated plankton classification which has otherwise seen little applications of deep learning.

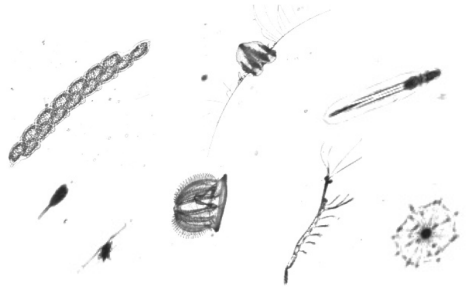

Figure 1. Some examples of images in the training set

2. Problem Statement

2.1. Dataset

For this competition, 30,336 labeled training images and 130,400 unlabeled test images were made available by the organizers. These images are greyscale shadowgraph images of planktons which were extracted through an automatic segmentation and cropping process. As a result, each image contains a single organism/entity and vary in dimensions. There is a total of 121 different labels, i.e. taxonomic classes, for these images. Figure 1 shows an example of some of the training images.

There are several characteristics of the data that make this task difficult:

- Training examples are limited and overfitting is a real concern, especially considering that there are 121 different possible labels.
- Representatives from each class can have any orienta-

tion within the 3D space, resulting in very different-looking images within the same class.

- Some of the classes look very similar to each other that experts have a difficult time labeling them.
- The automatic segmentation process introduced certain block-like artifacts which complicates the feature learning process.

2.2. Competition Details and Evaluation Metric

The competition task is to build a model that assigns all 121 class probabilities to the given test images. Evaluations are done via online submissions to the Kaggle servers, after which each submission is scored and ranked on a public leaderboard. Submissions are evaluated using the multi-class logarithmic loss as such

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

where N is the number of images in the test set, M is the number of class labels, y_{ij} is 1 if the observation i is in class j and 0 otherwise, and p_{ij} is the predicted probability that observation i belongs to class j.

The log-loss scoring metric penalizes heavily for wildly wrong guesses and suggests a need to be conservative in our predictions. This also means that an overfitted model is particularly detrimental to logloss than it is to classification accuracy. For instance, we had models that performed better in terms of validation accuracy but performed much worse in terms of logloss as a result of overfitting.

3. Overall Approach

Our solution utilizes Caffe [6], a deep learning framework developed by UC Berkeley, and scikit-image for preprocessing and augmentation.

We also adhered to the rules of the competition which prevents the use of external data, i.e. no transfer learning and finetuning of existing models trained on other datasets. Additionally, the benefits of transfer learning are quite limited in this case as it would inevitably involve training the model from activations very early in the network, given that the image dataset for this task is highly niche and domain specific.

Our overall strategy for the competition is as follows: build successively larger and deeper networks until we can overfit the training data, then subsequently employ various techniques to reduce overfitting. Various evaluation methods were also used (e.g. plotting loss curves, training/validation accuracies, visualizing first-layer filters, etc) to diagnose each model and to decide how to best improve each model.

4. Data Preprocessing and Augmentation

Data augmentation was crucial in achieving good results in this competition given the relatively small training set. We found that, on average, data augmentation increased validation accuracies by approximately 10% and dropped logloss by approximately 0.2.

4.1. Resizing and Random Cropping

The original images are of various dimensions, typically less than 100 pixels on the longer edge. The images of long planktons are usually rectangular while the images of blob-like planktons were more square.

We experimented with both preserving and forgoing aspect ratio when resizing images and found that both methods rendered very similar performances. This is largely because preserving aspect ratios led to images with large amount of empty space which hindered learning, while forgoing aspect ratios led to class confusions between classes that became similar after resizing (e.g. a long plankton looking very similar to another blob-like plankton after resizing).

We eventually chose to resize the images to 64x64 while forgoing aspect ratios. The resized images were then randomly cropped to 48x48 for training.

4.2. Mean Subtraction and Scaling

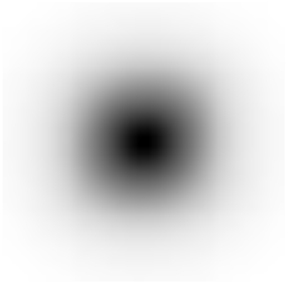

Figure 2. Calculated mean image from all training images

We calculated and subtracted the mean of all pixels across each image. We verified the correctness of the mean image by plotting it as shown in Figure 2. This result makes sense as most, if not all, of the input images had the object of interest centered in the image with whitespaces around it.

We also scaled the pixel values from [0, 255] to [0, 1].

4.3. Affine Transformations and Jittering

We augmented the data with various transformations:

- Scaling: uniformly sampled from $\frac{1}{0.8}$ to $\frac{1}{1.2}$; we were conservative with scaling as shadowgraph imagery generally produces images that are invariant to scale

- Rotation: uniformly sampled from 0 to 360 degrees
- Shearing: uniformly sampled from -15 to 15 degrees
- Flipping: Bernoulli probabilities of flipping up/down and left/right
- Gamma jittering: gamma value uniformly sampled from 0.8 to 1.6; we were conservative with changing gamma as increasing it too much exacerbated the block-like artifacts in the images

The data augmentation was performed offline by randomly generating 3 augmented images per training image, resulting in an augmented training set that has 121,344 images, i.e. four times the given training set. This dataset is then divided into a 75/25 training/validation split, where the training set consists entirely of the augmented images and the validation set consists of all the original training images.

5. Early Attempts

The first architecture that we explored is very similar to LeNet as used to classify the MNIST digit handwriting dataset [7] to great success. This was motivated by the similarities between our dataset and the MNIST dataset, in that they both consist of grayscale domain-specific images (and not "natural" images as in ImageNet).

The network is made up of four layers: the first two are convolutional layers and the last two are fully-connected layers as such: `[conv-ReLU-pool]x2 - [fully-connected] - [fully-connected] - [softmax]`. The convolutional layers have 20 and 32 number of filters respectively. The two fully-connected layers have 500 and 121 neurons respectively. The network was trained using stochastic gradient descent (SGD) with momentum. This model achieved approximately 65% training accuracy and 55% validation accuracy and a relatively high logloss of approximately 1.5. The model clearly lacked to capacity to overfit the training set and indicated the need for a network with greater capacity.

Our subsequent larger and deeper networks achieved higher training and validation accuracies (approximately 85% training accuracy and 70% validation accuracy). However, these networks were generally overfitted and resulted in lower but still undesirably high loglosses of 1.3.

We will now discuss two of our best performing models and the various techniques we used to reduce overfitting and achieve a 0.8 logloss.

6. Best Performing Models

Overview Our two best performing networks were of very different architectures and performed in the low 0.9 logloss range after controlling for overfitting. The first of the two is an AlexNet-like network with 7 layers and incorporates many features of AlexNet [8], including local response normalization layers and dropouts. The second best performing model is a "Tiny VGGNet [9]" that utilizes a homogeneous architecture and consecutive convolutional layers – however, we significantly reduced the number of filters and layers to make training tractable. We also see later that averaging across an ensemble of these two different network architectures led to significant improvement in results.

Training Both models and their variants were trained using SGD with Nesterov momentum (fixed at 0.9). All models were trained on a learning rate schedule starting at 0.01 with step decreases by factor of 10 every 20,000 iterations. All models were trained to convergence, i.e. once changes in validation accuracies fall within a threshold for repeated iterations.

Initialization All weights were initialized using the Xavier algorithm as described by Glorot and Bengio [10]. This initialization method automatically determines the scale of initialization based on the number of input and output neurons by drawing them from a Gaussian or uniform distribution with zero mean and a specific variance of $Var(W) = \frac{1}{n_{in}}$ where W is the distribution of the neuron and n_{in} is the number of neurons feeding into it. Intuitively, this initialization method allows the signal to propagate deep into the network.

6.1. AlexNet-like Model

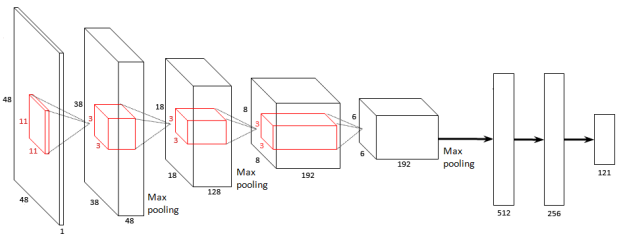

Figure 3. AlexNet-like model that has one less convolutional layer, less filters and no grouping

The first of our two best models is an AlexNet-like network with 7 layers: 4 convolutional layers and 3 fully-connected layers as shown in Figure 3. This model is similar to AlexNet except for one less convolutional layer, half the number of filters, no grouping and the use of smaller filters in the second layer. These changes in architecture were made as our input images are far smaller (i.e. 48x48) compared to 256x256 that AlexNet was designed for. It also

made training less computationally intensive given limited hardware and resources.

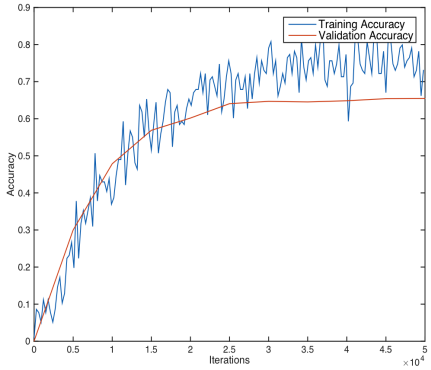

Figure 4. Training and validation accuracies for AlexNet-like model across training iterations

All convolutional layers include a ReLU nonlinearity, followed by 2x2 max-pooling and local response normalization (as detailed in Krizshevsky et al's 2012 paper [8]). Local response normalization has been shown to be useful when used with neurons with unbounded activation (e.g. ReLU) [8]. It effectively performs a type of regularization that encourages "competition" for big activities among local regions of neurons, and models a form of lateral inhibition found in real neurons. The fully-connected layers feature dropout with a probability of 0.5 and the loss function includes L2 regularization of 0.0005. As shown in Figure 4, these techniques combined substantially reduced overfitting and resulted in a model that has approximately 75% training accuracy and 65% validation accuracy. This model gave a logloss of 0.993999.

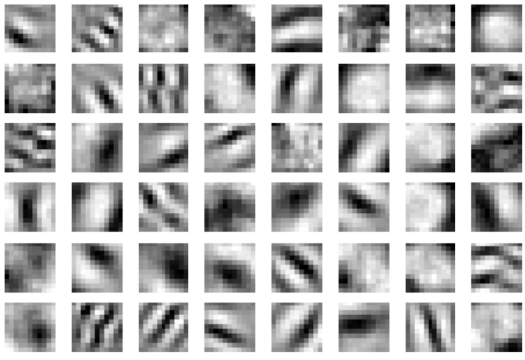

Figure 5. Visualization of the filters in the first convolutional layer

Visualizing the filters in the first convolutional layer (see Figure 5) shows that the network learned various frequency and orientation-specific kernels, as well as certain blob-like kernels. These learned filters are very similar to the color-agnostic filters learned in the first group (i.e. the first 48 filters) in AlexNet [8]. The existence of many more blob-like filters is expected given the nature of the training images and suggests that the network has successfully learned these more discriminative features.

6.2. "Tiny VGGNet" Model

We hypothesized that the aggressive pooling in our first model might have eroded too many features too quickly given the small dimensions and the fine features in our input images. Furthermore, we wanted to experiment with an architecture with greater depth. We chose to build a model that has a similar architecture to VGGNet [9] in that it contains several consecutive convolutional layers separated by ReLU's and has a homogeneous structure containing only 3x3 convolutions and 2x2 max-pooling from the beginning to the end. However, we significantly reduced the number of filters and layers (while still making it deeper and with more filters than our previously described model) to make training the model tractable.

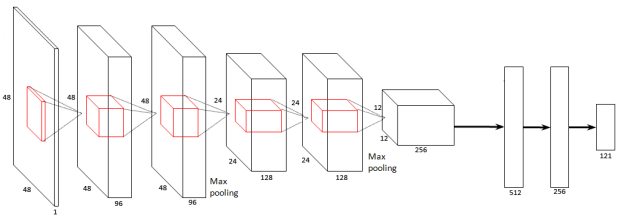

Figure 6. VGGNet-inspired architecture featuring consecutive convolutional layers and homogeneous structure

As illustrated in Figure 6, this model contains 8 layers: 5 convolutional layers and 3 fully-connected layers. The biggest difference between the two models are the consecutive convolutional layers separated by minimal max-pooling. The biggest motivation behind this architecture is so that the finer features (which typically are the most discriminative features) in the training images can be better preserved with more conservative pooling. We also doubled the number of filters in all the convolutional layers.

All convolutional layers have 3x3 kernel size with stride 1 and pad 1, and include ReLU nonlinearities. All max-pooling layers are 2x2 with stride 2 and no padding.

As shown in Figure 7, this model achieved a training accuracy of approximately 80% and a validation accuracy of approximately 65%. The figure also shows an acceptable but higher level of overfitting compared to the previous model. The best logloss achieved using this model is 0.949342.

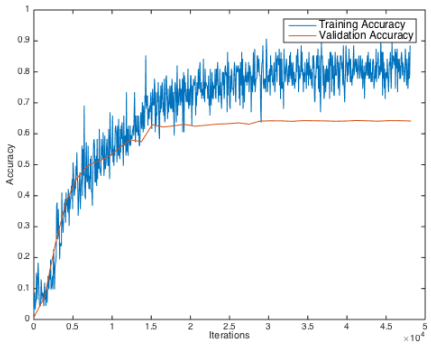

Figure 7. Training and validation accuracies for tiny VGGNet model across training iterations

7. Model Averaging

As commonly practiced in various visual recognition tasks, one reliable approach to improving model performance is to average across augmented images as well as across model architectures. Intuitively, this can be thought of as getting a "second opinion" from various other models so as to reduce the variances in the predictions. The improvement as a result of averaging is especially dramatic when using the logloss metric as logloss penalizes very heavily for predictions that are highly inaccurate; averaging hedges against the risk of any one model making particularly bad predictions.

7.1. Averaging Across Augmented Test Images

For each individual model, we made our predictions based on 10 random crops and flips of each test image. The predictions are then uniformly averaged across these 10 predictions. By averaging across augmented test images, we reduced logloss by close to 0.2.

7.2. Averaging Across Ensemble of Models

Table 1. Comparison of Ensembles of Top-k Models

Top-k Models	Logloss
5	0.903371
4	0.900150
3	**0.885346**
2	0.891029

We also averaged the predictions across an ensemble of models consisting of variants of our two best performing models (as described in the previous section). In particular, our best performing ensemble consists of uniformly averaging across the three best performing models:

- AlexNet-like model as described above
- AlexNet-like model without random cropping and using 64x64 input images
- Tiny VGGNet as described above

This ensemble gave our final logloss of 0.885346, about 0.5 to 1.0 logloss better than any individual model. This relatively high improvement can be attributed to the averaging across two very different models and their variants, i.e. AlexNet-like and Tiny VGGNet.

7.3. Other Possible Averaging Strategies

We also came up with several other averaging strategies but unfortunately did not have the time to implement them. For instance, we considered using the most confident predictions for each test image, which is a risky approach but can potentially increase our score drastically. We also considered using a weighted average of the models based on a linear regression to reduce validation error.

8. Additional Analyses and Observations

In this section we document some of the more interesting observations we made. Many of these observations are a result of various experiments we did in our attempt to improve our existing models.

8.1. Grouping

We originally trained our AlexNet-like model from scratch using a grouping of 2 as per Krizshevsky et al's 2012 paper [8]. We found from our own experiences that the use of grouping led to slightly deteriorated performance. We however have not done a thorough experiment to quantify this differences in performance.

Our understanding of Krizshevsky et al's use of grouping is so that they could fit their model into the memory constraints of the GPUs at that time. As such, we believe that there is little use for grouping for our purpose (or when using modern GPUs with sufficient memory to fit most models).

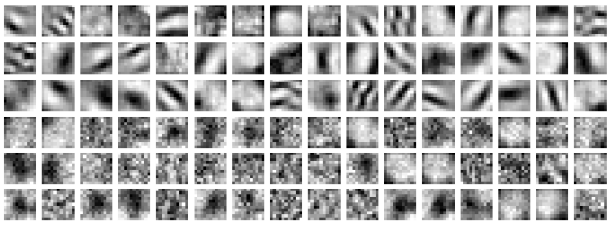

Figure 8. Visualization of all 96 filters in an AlexNet-like model with groups. Notice that only half the filters have converged.

We however noted an interesting observation where exactly half the filters, i.e. one of two groups of filters, converged properly while the other half did not (see Figure 8). This is despite training the model for several epochs beyond the convergence of its validation accuracy. This is very likely the result of setting the grouping parameter to 2, resulting in only the first group learning. When we tried using the same number of filters (i.e. 96 first-layer filters) without grouping, we found that all 96 filters converged properly.

8.2. Using Smaller Filters with Deeper Models

We found that our AlexNet-like model with 11x11 first-layer filters performed better than the same model with 3x3 first-layer filters and two additional convolutional layers. This result contradicts the intuition that smaller filters with deeper models exploit more non-linearities and give better performance. However, it should be noted that the documented performance gains in Simonyan et al's paper [9] was from the use of small filters together with very deep (16 to 19 layers) architectures and hence may not always be applicable to much shallower models.

8.3. Using Data Augmentation

We found that data augmentation was crucial for good performance in this task. The use of data augmentation helped to reduce logloss by about 0.2 across all the models we have trained so far. We suspect that more elaborate data augmentation schemes can further reduce logloss. However, we did not have the time to test this.

9. Conclusions

Given more time and resources (computing power, in particular), we would like to have experimented with deeper and bigger models (specifically smaller filters and deeper architectures a la VGGNet). We would also like to have experimented with transfer learning outside the context of this competition (i.e. disregard the competition rules that prohibit use of external data). However, given the limitations, we found that this Kaggle competition provided a great framework for practicing the various concepts taught in this course. In particular, the competition organizers provided a relatively clean dataset so that we could focus on applying the concepts learned in the course (and not with data collection and sanitation). The images also lent themselves well to the use of convolutional neural networks, but were still challenging enough such that we had to be careful with the architectures and techniques used to combat overfitting.

References

[1] Q. Hu, C. Davis *Automatic plankton image recognition with co-occurrence matrices and Support Vector Machine*. Marine Ecology Progress Series, 2015

[2] Q. Hu, C. Davis *Accurate automatic quantification of taxa-specific plankton abundance using dual classification with correction*. Marine Ecology Progress Series, 2016

[3] G. Gorsky, M. D. Ohman, M. Picheral, S. Gasparini, L. Stemmann, J. Romagnan, A. Cawood, S. Pesant, C. Garca-Comas, F. Prejger, *Digital zooplankton image analysis using the ZooScan integrated system*, Journal of Plankton Research, 2010

[4] C. S. Davis, S. M. Gallager, A. R. Solow, *Microaggregations of Oceanic Plankton Observed by Towed Video Microscopy*, Science 10 July 1992

[5] Kaggle *National Data Science Bowl*. [online] http://www.kaggle.com/c/datasciencebowl/ [accessed Feb 15, 2015].

[6] Jia, Yangqing and Shelhamer, et al. *Caffe: Convolutional Architecture for Fast Feature Embedding*. arXiv preprint arXiv:1408.5093, 2014.

[7] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, *Gradient-Based Learning Applied to Document Recognition*, Proceedings of the IEEE, November 1998

[8] A. Krizhevsky, I. Sutskever, G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, Advances in Neural Information Processing Systems (NIPS), 2012

[9] K. Simonyan, A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, arXiv technical report, 2014

[10] X. Glorot, Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, International Conference on Artificial Intelligence and Statistics, 2010