Alex Zamoshchin (alexzam), Jonathan Gold (johngold)

Convolutional Neural Networks for Plankton Classification: Transfer Learning, Data Augmentation, and Ensemble Models

1. ABSTRACT

We designed multiple Convolutional Neural Networks (CNNs) to enter a competition on *Kaggle.com* for the *National Data Science Bowl*. More concretely, we built a system that automates image identification for plankton and other marine organisms/entities. Using a training dataset of 30,000 labeled plankton images, we attempted to classify a test dataset of 130,400 images into one of the 121 classes. We analyzed and interpreted the success of our CNNs and implemented several different techniques including transfer learning, data augmentation, and ensembles in an effort to improve results. Transfer learning from two separate models, AlexNet trained on ImageNet and LeNet trained on MNIST, were examined, with AlexNet's deeper architecture and larger original dataset proving more beneficial for the task. Data augmentation, including constructing new images derived from random rotations of each training image, supplemented the existing training set and lowered loss. After introducing a larger training dataset, reducing dropout and regularization helped add expressive power to a network previously underfitting. Finally, building ensembles of multiple different models provided a final boost by reducing variance. In sum, our models were able to achieve a final multiclass log-loss of 1.1946 on the *Kaggle.com* submission leaderboards.

2. INTRODUCTION

The purpose of this investigation was to tackle the problem of Plankton Classification as described in the *National Data Science Bowl* competition on *Kaggle.com*. More concretely, the challenge was to build a system that automated image identification for marine organisms/entities. This final system could become extremely useful for monitoring and measuring plankton populations, which are an ideal measure of the health of the world's ecosystems and oceans.

Typically, there are many technical challenges associated with computer vision classification, especially in an area as challenging as ocean imagery. Some of these challenges are the sheer quantity of species, the fact that all entities are oriented in 3D space, the fact that the ocean is full of detritus and fecal pellets, the amount of noise present in some images, and the presence of unidentifiable species, and therefore the "unknown" classes.

In this paper, we examined the role of CNNs for plankton classification and implemented several techniques including transfer learning, data augmentation, and ensembles in an effort to improve our efforts. The reasoning, implementation, and results of these efforts are discussed below.

3. BACKGROUND

The CNNs used in this investigation build on two different models developed previously, in particular: AlexNet trained on the ImageNet dataset and LeNet trained on the MNIST dataset. According to *ImageNet Classification with Deep Convolutional Neural Networks* (2012), AlexNet is an eight layer CNN, of which five are convolutional and three are fully-connected, trained on a dataset of over 15 million high-resolution images belonging to approximately 22,000 classes. According to *Gradient-Based*

Learning Applied to Document Recognition (1998), the version of LeNet we used is a four layer CNN, of which two are convolution and two are fully-connected, trained on a dataset of 60,000 training examples. Applying transfer learning from these two models to the task of plankton classification was a major component of this investigation.

Additionally, an examination of the state-of-the-art in plankton classification gave us a useful metric on which to judge results. In *Automatic plankton image recognition with co-occurrence matrices and Support Vector Machine* (2005), Hu and David describe an acceptable accuracy of 67-83% for the task of plankton classification. Since then, this range has stuck and was one of the metrics to which we compared our accuracy results.

4. DATASET

The training dataset consisted of 30,000 labeled examples of plankton images, used to classify a held-out test dataset of 130,400 images into one of 121 classes. More concretely, each image was to be classified into one of 121 plankton species, including several classes for unknown entities and artifacts.

5. PREPROCESSING AND PRELIMARNY RESULTS

After preliminary inspection of the data, we counted the number of classes and the number of training examples per class. A histogram of the number of examples each class contained is displayed below:

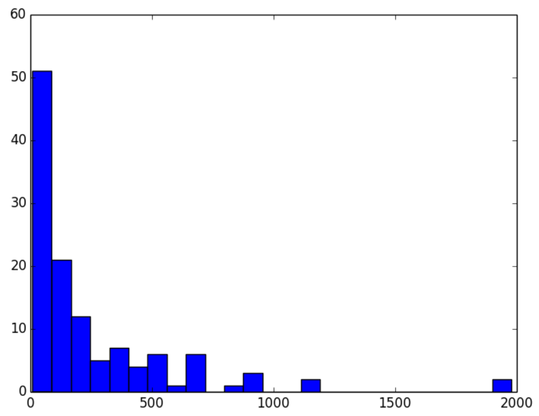

Fig 1: Histogram of class sizes

Next, we performed a number of initial preprocessing steps. The images were of different sizes, so the first step was to resize them all to the same size. We chose to resize images to either 256x256 or 48x48 pixels, experimenting with the trade-off between data loss and computation time. Next, we inspected the mean images of a few different classes, and two of the more interesting ones are displayed below:

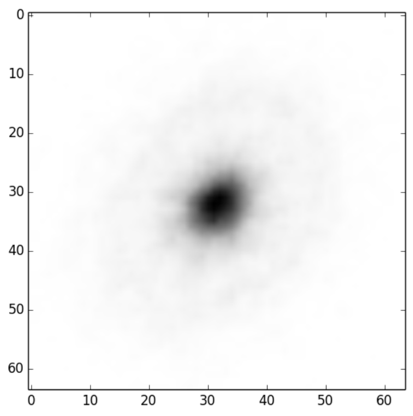

Fig 2: Mean image of class 'acantharia protist halo'

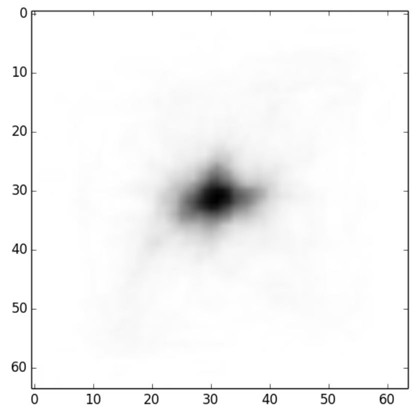

Fig 3: Mean image of class 'copepod calanoid frilly antennae'

The first class's mean image contained a faint ring around the center and the second class's mean image contained lines streaking from the middle blob, both of which were already indicative of the types of plankton images in these classes. In addition, we inspected the mean image itself, which can be seen below:

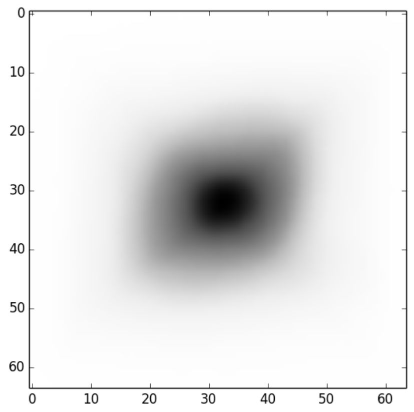

Fig 4: Mean image of full dataset

The full dataset mean image contained a large dark blob in the middle, which was to be expected since most of the plankton were located in the center.

6. EVALUATION

The evaluation method by which we evaluated our results was multiclass log-loss. Meaning, given a list of predicted probabilities for each image, the final loss was determined as:

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

We compared our results using this evaluation metric to a baseline of the uniform probability distribution across all classes, resulting in a log-loss of 4.795791, as well as to other results in the leaderboard.

7. TECHNICAL APPROACH AND EXPERIMENTS

After preprocessing, the technical approach consisted of several stages: transfer learning from various pre-trained architectures, data augmentation, hyper-parameter tuning of dropout and learning parameters, and ensemble techniques, all in an effort to decrease our loss on the test dataset.

7.1. Transfer Learning from AlexNet on the ImageNet Dataset

Our first approach consisted of using an identical architecture as AlexNet and fine-tuning from the weights trained on ImageNet. The motivation behind such an approach was that insight gained from AlexNet on the ImageNet dataset, especially the accumulated knowledge on shapes, could be beneficial to training on our dataset. Although weights were initialized from AlexNet, the final fully-connected layer was not taken and instead replaced with a fully-connected layer of size 121, to predict plankton classes. Since our dataset was not extremely similar to the original ImageNet data, none of the weights were held constant and instead we continued training all weights using significantly reduced learning rates. One problem with this approach was that ImageNet was a dataset of color images, and therefore contained data input dimensions different from those of the plankton images, which were in grayscale. Moreover although transfer learning in general is a good option, knowledge learned from the RGB dataset likely contained representations of color that were not useful for our task. Nevertheless, we decided transfer learning from ImageNet was still beneficial, not only because all accumulated knowledge of lines and shapes was still applicable, but also because ImageNet itself contained many grayscale images. Two different approaches were considered for tackling the challenge of varying input dimensions. Our initial approach consisted of treating our images as grayscale and cutting of the first convolutional layer of AlexNet, replacing it with our own so that it was compatible with the dimensionality of grayscale images. Although this approach seemed practical at first, the loss of trained weights early on in the network lost much of the insight that the network originally contained. Hence, a more sensible approach was to treat all images as colored images, effectively duplicating the depth dimension three times into RGB values. Such a technique allowed us to continue to use AlexNet in its original form, and provided the best results of the two methods. A comparison is seen below:

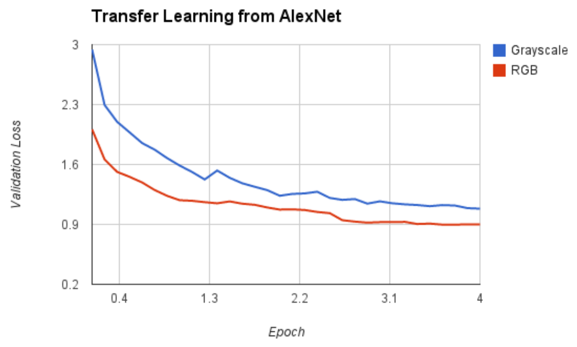

Fig 5: Validation loss vs. epochs for AlexNet trained on grayscale or colored images

From the graph it is evident that using the first layer's trained weights was beneficial not only for the initial loss, but also for the eventual convergence loss. For this reason, we performed transfer learning from the unmodified AlexNet, barring the deepest layer, and continued to build on the performance of this model.

7.2. Transfer Learning from LeNet on the MNIST Dataset

Another approach was to perform transfer learning from a network trained on the MNIST dataset. The intuition behind such an approach was simple: not only was this dataset in grayscale, reducing the differences between our dataset and the original dataset, but also many lines and curved shapes present in digits were also evident in plankton. Hence, perhaps the insight gained in differentiating different digits could be extended to the classification of plankton. Similarly to as when fine-tuning from AlexNet, none of the weights were held constant and instead a reduced learning rate was used. This decision was made because of the number of differences that still existed between our dataset and MNIST. Moreover, since LeNet was relatively shallow, at least relative to AlexNet discussed above, we also tried extending the network by adding several large fully-connected layers at the end of the network. This still allowed for transfer learning but also increased the depth of LeNet. This change also agreed with our intuition that recognizing straight and curved lines, an ability usually captured by early layers in a network, could be built-upon in deeper layers to understand more complicated plankton shapes.

After implementing these experiments, it was deemed that, barring a significant investment in hyper-parameter tuning, a substantial improvement would not be achieved via transfer learning from LeNet. Even at various learning rates, the model was not able to achieve training and validations losses similar to those achieved by AlexNet, and seemed stuck at a multiclass log-loss of approximately 4.5. The reason for this was hypothesized to be the extremely shallow nature of the network, even after addition of the addition of more fully-connected layers.

7.3. Data Augmentation

The primary motivation behind data augmentation was the relatively small set of training images. With only approximately 30,000 examples of labeled plankton, the dataset was very small relative to other image classification datasets that have been trained on in the past. Moreover, since we used a portion of this training set for validation, effectively only 80%, or 24,000 examples were available for use for training. A deep network like AlexNet, without substantial regularization or dropout, would likely overfit any such training set. In an effort to bolster the number of training examples, and hence reducing variance, we implemented data augmentation in two stages: preprocessing and in real-time.

Our most substantial data augmentation efforts occurred in preprocessing, primarily due to the limitations of Caffe. Since every image was an instance of a marine organism floating in the ocean, it likely could be considered from any camera angle. Meaning for any plankton image, any rotation of that image was a viable training image for that class. In order to reduce errors associated with plankton orientation, and to bolster our training set, for each training image we created twelve supplementary training images using the following procedure: for each 15-degree increment between 0 and 180, a random rotation within that range was taken and an image was generated by rotating the original image by that amount. Special care was taken to extend images such that no part of the original image was lost, and so the background remained white. Rotations were only taken between 0 and 180 degrees because a random mirror flip was performed in the real time data augmentation stage, as discussed below.

The second state of data augmentation was in real-time. During this stage, a number of augmentations were implemented including a random mirror flip of the image, and a random crop. Two different random crops were used: from a reshape size of 256x256 to 227x227 and from a reshape size of 48x48 to 47x47, such that the sizes of the resultant images match the architectures of the networks. A random mirror flip was implemented so that with a probability 50%, a flip of the image across the vertical axis was taken instead of the original image itself.

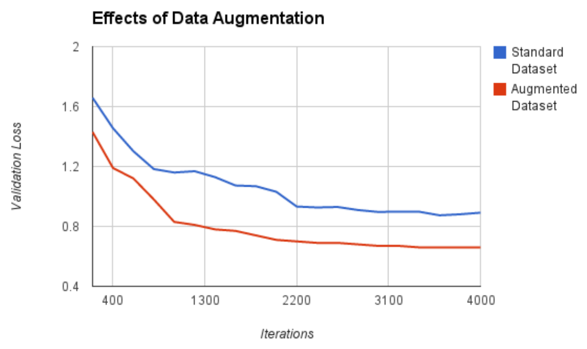

Fig 6: Validation loss vs. iterations for training on standard and augmented datasets. Note we plot number of iterations because the datasets are of different sizes.

Hence, training on the augmented dataset was able to achieve a lower final validation loss than training on the original dataset.

7.4. Augmenting Test Images

The original submission script involved simply submitting the predicting probabilities for each test image, and the next attempt implemented to improve performance was the averaging of probabilities for the predictions of several different augmentations for each test image. Meaning, the new method averaged together the predicted probabilities for a number of random rotations of each test image, including the original image itself. Such a technique was implemented for no rotations, two rotations, and ten rotations, displayed below:

	No rotations	2 rotations	10 rotations
Multiclass log-loss	1.3322	1.54567	2.1809

Unfortunately this method did not show any boost in performance, and in fact seemed to significantly diminish results. It is not immediately clear why no improvement was seen, although many reasons could exist including an improper implementation of the strategy, or a strange property in the dataset.

7.5. Tuning Learning Methods

A number of different learning methods and associated hyper-parameters were evaluated in tuning our networks. The first hyper parameter we tuned was the learning rate. The standard model started with a learning rate of 0.001, but we found that a slightly higher starting rate caused a more rapid descent. From there we tested the step-size in conjunction with the learning rate decay parameter, which annealed the learning rate after a certain number of iterations. When starting from ImageNet trained weights, it worked best to have a high learning rate decay, causing the learning rate to decrease rapidly. However, after multiple epochs the learning rate was too small, so we would start from the most recent snapshot and decrease the learning rate decay, causing the learning rate to again anneal slower.

One discovery we made was that different types of gradient descent methods were able to converge on final losses lower than that of stochastic gradient descent. We experimented with both Adagrad and Nesterov, for which the results are plotted below.

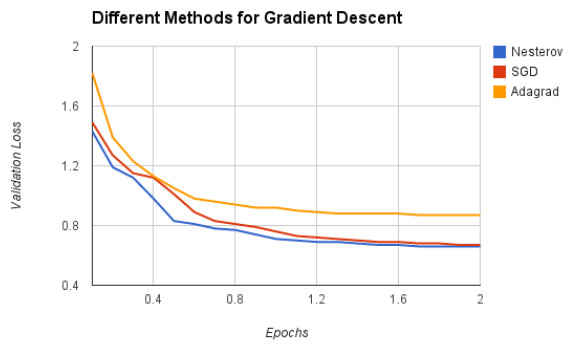

Fig 7: Validation loss vs. epochs for different learning methods.

As you can see, Nesterov's Accelerated Gradient Descent worked the best for us. It was not immediately clear why Nestrov's method was able to reach a final convergence loss lower than those of other methods, but perhaps the properties inherent in this method most accurately match the descent curve of our model.

7.6. Tuning Regularization and Dropout

After evaluation of our results thus far, examination of training and validation losses led to the hypothesis that some of our existing models were underfitting. This is evident in the models discussed above: in many cases training and validation losses are approximately equal. Though not necessarily a negative sign, this could be indicative of an underfitting network that would benefit from more expressive power. Since the network was already fairly deep, the two primary strategies for improving an underfitting network were to reduce dropout or to reduce regularization via the weight decay parameter in Caffe. In fact, preliminary analysis of tuning the dropout parameters for all dropout layers in the network showed that simply decreasing these parameters could improve results.

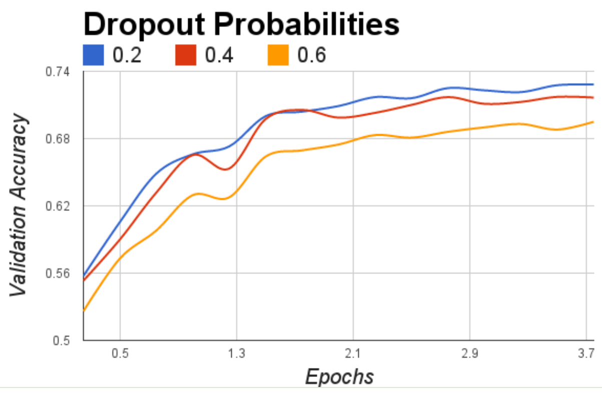

Fig 8: Validation accuracy vs. epochs for different dropout parameters. Note we plot validation accuracy because the difference is more prominent.

Implementation of these changes led to a loss reduction of about 0.153.

7.7. Ensemble Methods

As a final step in optimizing our performance we created ensemble models consisting of averages of our previously submitted results. Meaning, taking two lists of predicted probabilities, an ensemble list of predicted probabilities was constructed by averaging each probability from all of the input models. As seen in previous investigations, an ensemble of multiple models is able to improve performance by reducing variance, in effect hedging the bets made by the model's predictions. Such a technique provided the last substantial improvements in our test loss, and these results are displayed below:

	Model 1	Model 2	Model 3	Ensemble
Multiclass log-loss	1.3512	1.3334	1.3163	1.1946

Hence, we created an ensemble model from several networks initiated with AlexNet's weights trained on ImageNet, on augmented training sets, with a dropout probability of 0.2, and trained with Nestrov's Accelerated Gradient Descent. Clearly, the ensemble model was able to outperform any one of the existing models taken by themselves.

8. CONCLUSION

As discussed above, this investigation examined various transfer learning, data augmentation, and ensemble techniques. The ultimate CNN selected was an implementation of AlexNet's original architecture with weight initialization from weights trained on ImageNet. This deep network, in combination with insight learned on a data corpus of 15 million images, proved the most successful strategy. Data augmentation, including random rotation, mirroring, and cropping, also proved beneficial to our results, especially due to the small size of the original train dataset. Finally, creating ensembles of our various models provided another boost in performance, either because of the sensitive nature of our evaluation metric, multiclass log-loss, or because of the high variance present in our original models. In sum, our best model was able to achieve an eventual train loss of 0.22, validation loss of 0.61, and validation accuracy of 77%. When submitting on *Kaggle.com*, we were able to achieve a multiclass log-loss of 1.1946, placing us in the top 30% of competition.

Among the contributions made in this paper, the most notable was the surprising success of transfer learning from ImageNet, a dataset substantially different from ours involving only marine organisms. Moreover, the necessity for data augmentation when working with small training datasets, and our work with ensemble models, show the continued success of those techniques. Finally, it was interesting to note that almost all leaders in the *Kaggle.com* competition were implementations of Convolutional Neural Networks.

9. FUTURE WORK

Many further experiments could be implemented to further improve our results. Ideas for future work include further data augmentation, more intelligent methods for generating test predictions, and further hyper-parameter tuning. Additional data augmentation could be done to generate additional training examples via scaling, zooming, or modifying the contrast of the training images. Although averaging predictions for random rotations did not prove beneficial, a similar technique could be implemented for different augmentations including random cropping, mirror flips, or contrast filtering. As always, further hyper-parameter tuning would prove beneficial since, despite the extensive nature of this investigation, some hyper-parameters still remain only coarsely-tuned.

10. NOTES

Use of Terminal.com was a major hindrance to our efforts. In sum, around 5-6 hours were wasted vying with other students for GPU instances, and on one occasion an entire instance containing computed weights and augmentations was lost. We hope the teaching staff in evaluating our results acknowledges this difficulty.

11. REFERENCES

Alex, K., Stskever, I., & Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25,* 1097-1105. Retrieved March 11, 2015, from http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Bubeck, S. (2013, April 1). Nesterov's Accelerated Gradient Descent. Retrieved March 17, 2015, from https://blogs.princeton.edu/imabandit/2013/04/01/acceleratedgradientdescent

Hu, Q., & Davis, C. (n.d.). Automatic plankton image recognition with co-occurrence matrices and Support Vector Machine. *Marine Ecology Progress Series,* 21-31.

Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998