Convolutional Neural Networks-based Plankton Image Classification System*

An Zheng
A53095104
anz023@eng.ucsd.edu

Mingyang Wang
A53100579
miw092@eng.ucsd.edu

ABSTRACT

To the marine ecosystem, plankton are vitally significant for they produce more than half of the primary productivity on earth and play an irreplaceable role in the global carbon cycle. However, most of traditional plankton population measuring and monitoring methods become gradually impractical for they are time consuming and cannot scale to the granularity required in large-scale studies. Thus we introduced a machine learning model, convolutional neural networks, and utilized it to implement an image classification system and automate the plankton image identification process. In the experiment, our classification system's logarithmic loss in test set was 0.7736, performing much better than systems using linear model and multi-layer perceptron model. In addition, the noise resistance ability of our system was examined by introducing Gaussian noise.

Keywords

Plankton Classification, Convolutional Neural Network, Parameter Training

1. INTRODUCTION

The plankton population is one of the most important evaluation factor of marine ecological environment. And the observation of plankton' living condition highly relies on the classification of these plankton. With the observation tool becoming digitalized and the underwater images becoming increasingly easier to capture, the image dataset size in marine research projects grows in an extremely rapid speed and traditional manual classification methods cannot satisfy most of these experiments' requirements anymore. Thus, in our experiment, we introduced convolutional neural networks (CNN) into the plankton classification work, and used a CNN-based classification system to increase the classification speed.

*Inspired from Kaggle competition *National Data Science Bowl* https://www.kaggle.com/c/datasciencebowl

To guarantee the effectiveness and efficiency of our classifier, we did three works: (1) Using the invariability of convolutional neural networks, we augmented our dataset by rotating and scaling the original pictures to guarantee that the model is fully trained. (2) We examined the performance of 3-layer, 4-layer and 5-layer convolutional neural networks respectively and concluded the most suitable structure for our plankton image dataset. (3) We set comparison experiments to tune the model parameters such as learning rate and batch size. In addition, we compared the performance of CNN with linear model and multi-layer perceptron (MLP) model, and proved that our method was the best one among them, achieving 75.726% accuracy in classification. At last, our model's ability of noise resistance was examined by adding Gaussian noise in the dataset.

In all, we proved that our image classification system was suitable for the classification of plankton images. It could help researcher to study larger population of plankton and enable them to evaluate local environment timely.

2. LITERATURE

Image classification is a complex process including determination of a suitable classification system, selection of training samples, image preprocessing, feature extraction, selection of suitable classification approaches, post-classification processing, and accuracy assessment[8]. Traditionally, support vector machine classifier[9], maximum likelihood classifier, clustering classifier[3], single layer neural network classifier[5] are widely used.

In 2012 ImageNet Competition[10], the winner team obtained an error of 0.15315 by deploying convolutional network model[6], while the team in second place used a combination of traditional models and only obtained an error of 0.26172. The results show the convolutional networks have huge advantages over other traditional models.

According to Lecun[7], multilayer neural networks are good candidates for image recognition tasks, because they are designed to learn complex, high-dimensional, nonlinear mappings from large collections of examples. Moreover, the special architecture of shared weights gives convolutional network the ability to outperform the general Multilayer perceptron. With this idea, convolutional networks can model the real neural network better, the whole complexity of the network can be reduced, and the number of weight parameters in between layers is also reduced. When input data are in high dimensions, say, images, these advantages are extraordinary. For example, convolutional networks could take images as direct input and avoid the process of compli-

cated feature extraction and data reconstruction, as required by traditional image classification models. Besides, convolutional networks can ensure some degree of shift, scale, and distortion in variance.

After the Kaggle competition – *National Data Science Bowl*[2], in which we obtained our dataset, ended, it turned out that the winner team defeated other teams by deploying convolutional networks with many optimizations, as they shared their solution afterwards[1].

3. DATASET EXPLORATORY ANALYSIS

3.1 Dataset Description

Our dataset contains 160736 plankton images. These image data are collected by Oregon State University's Hatfield Marine Science Center and the dataset is available as part of the Kaggle challenge[2]. In this dataset, there are 121 different species, ranging from the smallest single-celled protists to copepods, larval fish, and larger jellies, and images are taken from different orientations within 3-D space. Each image in the train dataset is manually labeled by a trained team, and the classification results have been cross-validated. According to Culverhouse [4], experts are able to maintain 84-95% self-consistency in labeling difficult images. In each image, it is guaranteed that there is only one plankton creature in it.

The sizes of image are various, ranging from 37 x 41 pixels to 323 x 297 pixels. Hence, it is necessary to pre-process these images and scale them into the same size so that they can be processed by CNN model. Here is a sample image:

Figure 1: A fish larvae with medium body.

Take this sample image as an example. It is a gray-scale image with 96 x 96 pixels, and in each pixel, the value ranges from 0 to 127. Instead of abstracting features from the image, we take all pixels as features. Namely, for this input data, there are 9126(=96 x 96) features in all, and every feature has a value from 0 to 127.

After thoroughly examining the dataset, we find that there are some images too ambiguous to identify, and labeled as "unknown" by the experts. Images in this category could be a source of noise in the model training process and it requires the model applied to have the ability to handle special cases of unidentifiable objects. Also, not every "plankton creature" in the images is actual creature. Some of them are sticks, and some are blobs, which are also needed to be classified separately. In addition, according to the dataset introduction, because of the special shadowgraph imagery technique, organisms are at the same size regardless of distance to the camera, which means creatures can be roughly differentiated according to their sizes in the images.

In our experiment, we utilized 30336 images to train our CNN classifier (because Kaggle only provides 30336 images with labels, and the rest served as test cases), including 27238 images in the training set and 3098 images in the validation set. The rest images were taken as test set.

3.2 Image Pre-process Method

Since the number of training data we had was comparatively limited, it was very necessary to enlarge the training set with some data pre-process methods, which could help the model trained better. Specifically, the original data were processed in the following steps:

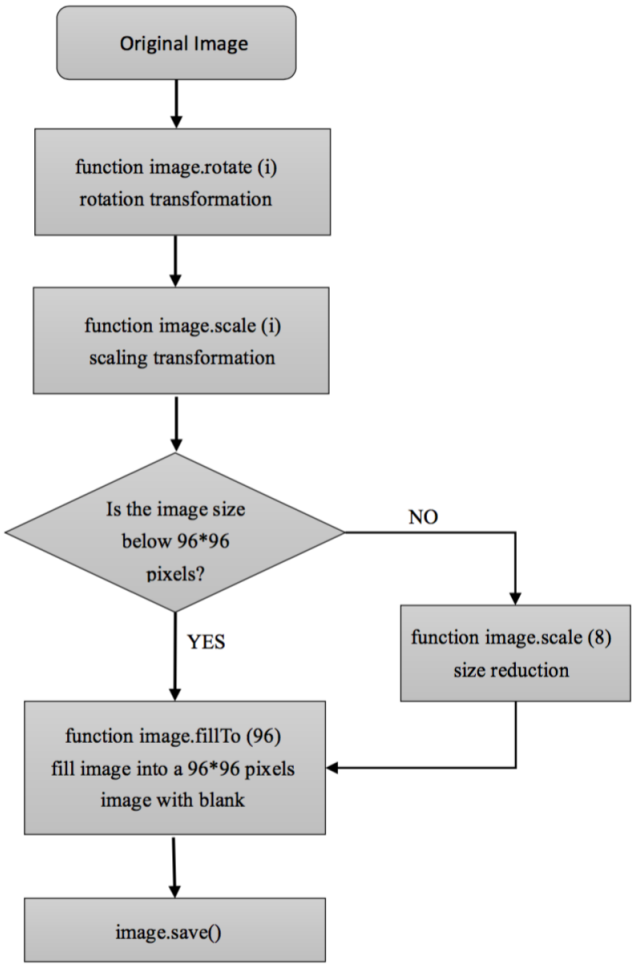

Figure 2: Image pre-process procedure.

Firstly, an original image was rotated by 0, 90, 180 and 270 degree in series to generate four new different angle images. Then each rotated image was filled into a square with blank, and the geometrical center of this square was the same as the original image.

Secondly, each image was randomly scaled by 1-1.6 times. This process was repeated four times. It is noteworthy that all the parameters in the rotation and scaling transformations were derived from Sander Dieleman's expertiment[1]. Here though the size of creatures in image changed slightly, their classes still could be roughly differentiated according to their sizes in the images because the scaling range was limited and corresponded to real size difference of the creatures in the same species.

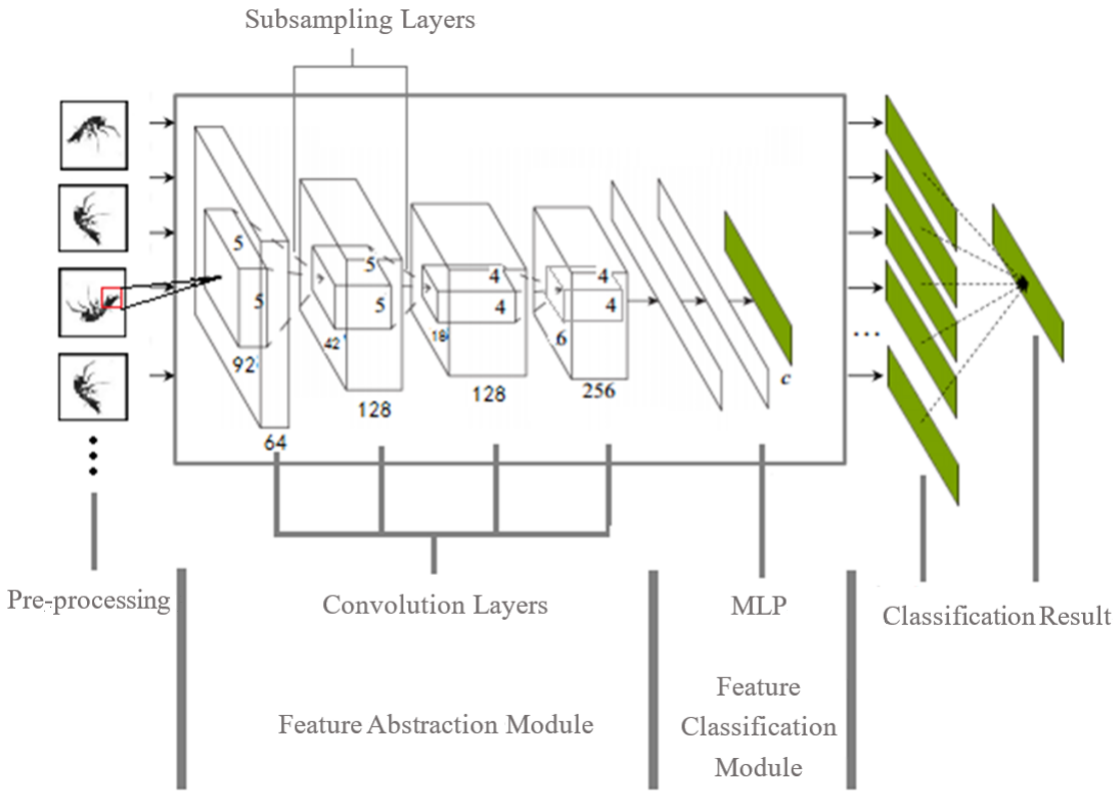

Figure 3: CNN-based classification system.

Finally, each image processed after above transformations was filled into a 96*96 pixels square images. However, if the image was originally larger than 96*96, its size would be reduced eight times and then filled into 96*96 pixels.

In addition, another detail was that the gray-scale value in each pixel was converted from [0,127] into [0,1], in order to satisfy the requirement of the CNN model we used.

3.3 Evaluation Method

We evaluated the classification results using the following logarithmic loss equation:

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

In this equation, N is the number of images in the test set, M is the number of categories of data in test set. When the i-th image is classified into the j-th category, y_{ij} equals 1. Otherwise, y_{ij} equals 0. p_{ij} is the probability of the i-th image being classified into the j-th category. It is noteworthy that, to prevent p_{ij} being too large in log domain, the value of p_{ij} is processed using the following equation to restrict the log-value:

$$p_{ij} = \max\{\min\{p_{ij}, 1-10^{-15}\}, 10^{-15}\}$$

4. MODEL DESCRIPTION

CNN model is a kind of artificial neural network. Compared with other basic artificial neural networks, it is highly suitable in dealing with image data. It can take original image data as input without complex feature abstraction and data reconstruction process which are required in the traditional image classification algorithms. More importantly, it ensures shift, scale and distortion invariance, enabling this algorithm performs better than MLP model[7].

A typical CNN model contains three main parts: the convolution layer, the sub-sampling layer, and the MLP classifier. The convolution layer and sub-sampling layer are stacked alternatively, and the combination of one convolution layer and one sub-sampling layer is called a CNN layer. When building a CNN model as an image classifier, one or multiple CNN layers are needed and connected in series. The number of CNN layers is highly significant to the classification accuracy: if the number is too small, the data features will not be fully abstracted and utilized in the classification process. But when the number of layers is too large, the run time will increase exponentially and, what is worse, cause some undesirable training problems such as gradient vanishing or exploding.

In our experiment, we constructed our CNN-based classification system as shown in Figure 3 and the functions of a single CNN layer is described in Figure 4. Image data were input in the convolution layer after pre-processed. Then after the convolution, data were sampled and thus the data dimension was reduced. It is noteworthy that, since the feature maps were already small enough in layer 3 and layer 4, there was no need to add subsampling layer in these two layers. After the process above, data generated were input into a MLP classifier. The parameters in the whole model were then tuned according to the data labels.

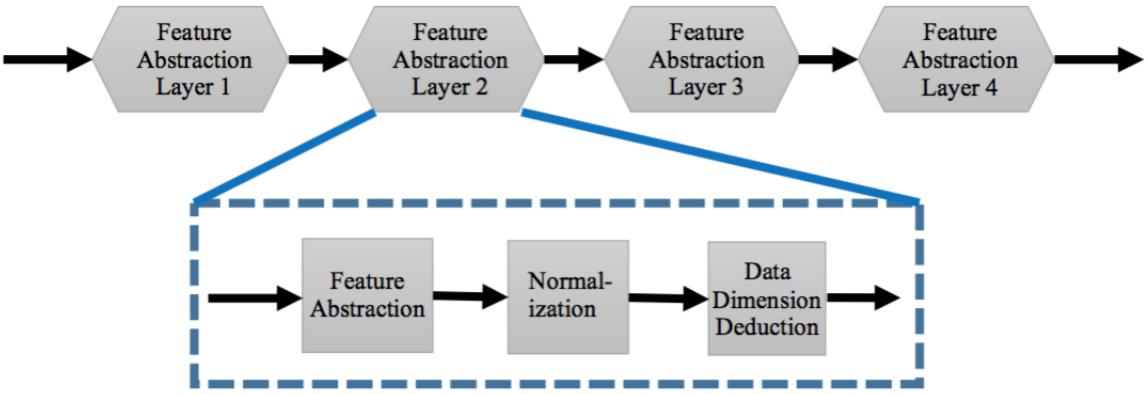

Figure 4: Functions of a single layer.

5. RESULTS

5.1 Runtime Environment

Our CNN model was established using Lua, an efficient scripting programming language. Torch7 was used as runtime environment in the experiment. Torch7 is a Matlab-like environment, containing a large number of machine learning libraries for users to utilize, including elemental methods of convolutional neural networks.

5.2 Number of CNN Layers

As mentioned in the previous part, the number of CNN layer can influence the classification accuracy and running time on a large scale. Thus, it was essential to decide how many CNN layer needed when building a CNN model. In this experiment, we constructed a 3-layer, a 4-layer and a 5-layer CNN model respectively, and used the same dataset to evaluate their performance. The result is shown in Table 1.

Table 1: Performance of CNNs against number of layers

	Logarithmic Loss	Running Time/min
3-layer CNN	1.0644	346
4-layer CNN	0.7736	431
5-layer CNN	1.2901	503

The 3-layer CNN did not perform as well as 4-layer CNN, because features in data were not fully abstracted and utilized in the 3-layer CNN. However, it seemed a little bit strange that the 5-layer CNN did not perform as well as 4-layer CNN model neither. After examining the model parameters in the trained 5-layer CNN model, we found that, the model suffered from gradient vanishing problem in the back propagation process, which caused the model was trained insufficiently. This problem happened because parameters in each layer decreased exponentially level by level. And when the parameters in the front layers was too small, the training process became very slow and finally stopped before fully trained because of a mechanism, "early stopping" we set beforehand. And thus, for our dataset, a 4-layer CNN model was the most suitable one.

5.3 Batch Size

In the experiment, the mini-batch technique was introduced to prevent that accidental factors were brought in and then led to the output logarithmic loss fluctuating rapidly. Instead of tuning the model parameters for every input image, we input a batch of images in the model every time, calculating the mean value and then tuning the model parameters using the mean value. The number of images in a batch was called batch size. An experiment was conducted to select the most suitable batch size, the result is shown in Table 2.

Table 2: Performances of CNNs against batch size

Batch Size	Logarithmic Loss	Running Time/min
32	0.7509	1136
128	0.7736	431
512	0.8164	153

From the experiment result, we can see that 32 images in a batch was the optimal for our dataset, but it would cost too much time to train the model if we used this batch size. So we actually used 128 images in each batch whose logarithmic loss was only slightly higher than 32 but enabled our system to complete the classification work in an appropriate and acceptable time period.

5.4 Learning Rate

Learning rate is another important parameter in model training. If the learning rate is too large, the pace in every update will be too large, and can hardly reach the minimum value. But if the learning rate is too small, the model can be stuck in a local minimum fast and cannot find more desirable point anymore.

We conducted an experiment to decide the most suitable learning rate. The result is shown as in Table 3.

The table shows that 10^{-3} was the most suitable value for our experiment. In fact, in our experiment, we applied a dynamic learning rate. The initial learning rate was 5×10^{-3}. Each time when the gradient descent did not reach the expectation we set, the learning rate would be reduced to one fifth of original value.

Table 3: Performances of CNNs against learning rate

Learning Rate	Logarithmic Loss	Running Time/min
10^{-1}	1.4536	229
10^{-2}	0.8062	306
10^{-3}	0.7833	410
10^{-4}	0.8898	348

5.5 Model Comparison

We compared the performance of three models in total, including CNN and two basic models, linear model and MLP model. The comparison result is shown in Table 4.

Table 4: Performance comparison between models

Algorithm	Logarithmic Loss	Running Time/min
CNN	0.7736	431
MLP	1.9213	63
Linear	2.4854	19

From the result, we can see that classification system based on CNN remarkably outperformed linear model and MLP model, though it took more time to complete the classification.

5.6 Noise Resistance Test

Finally we tested our model's ability of the noise resistance: we set four comparison groups, and added 300 (1% noise), 1500 (5% noise), 3000 (10% noise), 6000 (20% noise) randomly generated and labeled images into each group. The test result is shown in Table 5.

Table 5: Noise Resistance Test

Noise Ratio	Logarithmic Loss
0%	0.7736
1%	0.7730
5%	0.8809
10%	1.9023
20%	4.8524

From the result we can conclude that, our model was noise resistant when small amount of noise was added into dataset. But its performance would suffer a lot when the ratio of noise was more than 5%.

6. CONCLUSION

In our experiment, we implemented a CNN-based classification system to classify the images of plankton, which was much more efficient than traditional methods. Through experiment and analysis, we can conclude that: (1) Compared with traditional manual methods and basic machine learning algorithms such as linear model and MLP model, CNN model is a more suitable and desirable model in image classification for our dataset, achieving 0.7736 . (2) The number of CNN layer cannot be too large or small. For our dataset and model configuration, four CNN layers is the optimal. (3) Our system can resist noise to a certain degree, though when the ratio of noise in the dataset is larger than 5%, the influence of added noise will gradually become notable in model performance.

7. REFERENCES

[1] The deep sea. http://benanne.github.io/2015/03/17/plankton.html. Accessed: 2015-11-26.

[2] National data science bowl. https://www.kaggle.com/c/datasciencebowl/data. Accessed: 2015-11-26.

[3] S. Bandyopadhyay and U. Maulik. Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern Recognition*, 35(6):1197–1208, 2002.

[4] P. F. Culverhouse, R. Williams, B. Reguera, V. Herry, and S. González-Gil. Do experts make mistakes? a comparison of human and machine identification of dinoflagellates. *Marine Ecology Progress Series*, 247(17-25):5, 2003.

[5] J. A. Fernandes, X. Irigoien, G. Boyra, J. A. Lozano, and I. Inza. Optimizing the number of classes in automated zooplankton classification. *Journal of Plankton Research*, 31(1):19–29, 2009.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

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

[8] D. Lu and Q. Weng. A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, 28(5):823–870, 2007.

[9] T. Luo, K. Kramer, S. Samson, A. Remsen, D. B. Goldgof, L. O. Hall, and T. Hopkins. Active learning to recognize multiple types of plankton. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 478–481. IEEE, 2004.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.