Plankton Classification with Deep Convolutional Neural Networks

Ouyang py, Hu Hong, Shi zhongzhi
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
{ouyangpy,shizz}@ics.ict.ac.cn, huhong@ict.ac.cn

***Abstract*—Traditional methods for measuring and monitoring plankton populations are time consuming and can not scale to the granularity or scope necessary for large-scale studies. Improved approaches are needed. Manual analysis of the imagery captured by underwater camera system is infeasible. Automated image classification using machine learning tools is an alternative to the manual approach. In this paper, we present a deep neural network model for plankton classification which exploits translational and rotational symmetry. In this work, we propose two constrains in the design of deep convolutional neural network structure to guarantee the performance gain when going deep. Firstly, for each convolutional layer, its capacity of learning more complex patterns should be guaranteed; Secondly, the receptive field of the topmost layer should be no larger than the image region. We also developed a "inception layer" like structure to deal with multi-size imagery input with convolutional neural network. The experimental result on Plankton Set 1.0 imagery data set show the feasibility and effectiveness of the proposed method.**

***Keywords*—deep learning, image classification, convolutional neural network**

I. Introduction

Traditional methods for measuring and monitoring plankton populations are time consuming and can not scale to the granularity or scope necessary for large-scale studies. Manual analysis of the imagery captured by underwater camera system is infeasible. Automated image classification using machine learning tools is an alternative to the manual approach. Deep learning and convolutional networks (ConvNets) have recently enjoyed a great success in large-scale image and video recognition ([1], [2], [3], [4]) which has become possible due to the large public image repositories, such as ImageNet [5] or Plankton imagery data [6], and high-performance computing systems, such as GPUs or large-scale distributed clusters (Dean et al., 2012). In this paper we classifying plankton with deep convolutional neural network implemented with Caffe [7], in this work, we proposed a practical theory for designing very deep convolutional neural network structure. In addition, we developed a inception module for multi-scale image input. Based on property of the plankton data set, we designed rotational invariant data augmentation in training and testing stage.

978-1-4673-9194-8/16/$31.00 ©2016 IEEE

A. Planktons Classification Dataset

Plankton imagery data [6] collected from F.G. Walton Smith in Straits of Florida from 2014-06-03 to 2014-06-06 and used in the 2015 National Data Science Bowl, There are around 30k training data and 130k testing data. The goal is to classify gray scale images of plankton into one of 121 classes. They were created using underwater camera that is towed through an area. The images obtained using the camera were already processed by a segmentation algorithm to identify and isolate individual organisms, and then cropped accordingly. Interestingly, the size of an organism in the resulting images is proportional to its actual size and does not depend on the distance to the lens of the camera. This means that size all the images in the data set have different sizes. In our work we developed a inception module with convolutional neural network to deal with the multi-scale imagery input and the classification result outperforms single fixed size imagery input.

B. Related Work

Mainly due to the advances of deep learning, more concretely convolutional networks [8], the quality of image recognition and object detection has been progressing at a dramatic pace. Starting with LeNet-5 [8], convolutional neural networks (CNN) have typically had a standard structure C stacked convolutional layers (optionally followed by contrast normalization and max pooling) are followed by one or more fully-connected layers.

Network-in-Network is an approach proposed by Lin et al. (2014) [9] in order to increase the representational power of neural networks. Network in network build a micro multi-layer perception with more complex structures to abstract the data within the receptive field. The feature maps are obtained by sliding the micro networks over the input in a similar manner as CNN. When applied to convolutional layers, the method could be viewed as additional 1×1 convolutional layers followed typically by the rectified linear activation proposed in AlexNet [1]. This enables it to be easily integrated in the current CNN pipelines and allows for increasing the depth without significant performance penalty.

With ConvNets becoming more of a commodity in the computer vision field, a number of attempts have been made to improve the original architecture of Krizhevsky et al. (2012)[1]

in a bid to achieve better accuracy. For instance, K Simonyan and A Zisserman (2015) [10] proposed VGGNet [10] investigating the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. VGGNet evaluated networks of increasing depth using an architecture with very small (3×3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to very deep (16-19) weight layers. GooLeNet [11] is a deep convolutional neural network architecture codenamed inception, the main hallmark of this architecture is the improved utilization of the computing resources inside the network, that allows for increasing the depth and width of the network.

C. Motivations

Going deep is the essential of deep learning and has demonstrated a great success in many tasks. Both GooLeNet and VGGNet using very deep convolutional neural networks. Going deep greatly improves the learning ability of the network. Although it has become a common sense to going deep, it is still unclear how to design a very deep convolution neural network effectively. Most times arbitrarily adding more layers does not help, or even worsen the performance. In this work, first we propose a practical theory for designing very deep convolutional neural network effectively. The design of deep convolutional neural network is considered as a constrained optimization problem. The objective is maximizing the depth of the target convolutional neural network, subjecting to two constraints:

1) the c-value of each layer should not be too small, c-value is a metric for measuring the capacity of learning more complex patterns. Which gives a lower bound and it means the learning capacity of each layer should be guaranteed.
2) the receptive field of the topmost convolutional layer in the feature-level should no larger than image size. Which presents a upper bound and it means neurons have seen the entire image region, it stops emerging new and more complex patterns, the driven force for adding new layers no longer exists.

Secondly, we designed a inception module for multi-scale imagery input and it shows advantage of single-scale imagery input. Thirdly, we developed various affine transforms to do data augmentation, includes rotation, in order to gain the rotational invariant property. Data augmentation in training and testing stage helped decreasing over fitting risk and improve predict performance.

II. Design a deep convolutional neural network

In this section we present a overview of our model for classifying plankton with convolutional neural networks. First we propose a theory for design very deep convolutional neural network; Then we present the inception module for multi-scale imagery input; In addition, we introduce the data augmentation in our training and testing stage; Finally, we show the representative network architectures following our structure design theory.

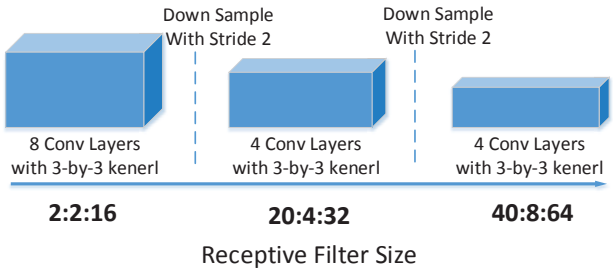


Fig. 1. An illustration of receptive field size. The receptive field sizes of all convolutional layers are represented in matlab-style arrays for clarity. For example, 40: 8: 64 represents [40, 48, 56, 64]. The size of input image is 64-by-64. We subtract an annoying small constant 1 when calculating the receptive field sizes in order to make the description and subsequent derivation more concise.

A. Going deep with convolutions

we propose two constrains in the design of deep structure to guarantee the performance gain when going deep. Firstly, the c-value of each layer should not be too small, c-value is a metric for measuring the capacity of learning more complex patterns. Secondly, the receptive field of the topmost convolutional layer in the feature-level should no larger than image size.

1) Capacity of learning - the first constraint: When convolutional neural network goes deep without down sampling, the possibility of the learning failure grows. because the sizes of the detected patterns and their meaningful spatial relationships grows layer by layer. Once the spatial relationships of the majority input patterns exceed the filter size of a convolutional layer, this convolutional layer will lose its capacity of learning more complex patterns. To quantitatively measure the learning capacity of a convolutional layer we define the c-value of a convolutional layer as follows.

$$\text{c-value} = \frac{\text{Real Filter Size}}{\text{Receptive Field Size}} \tag{1}$$

Where the **Real Filter Size** in a k×k convolutional layer is k if there is no down sampling. It doubles after each down sampling if the pooling stride is 2. The **Receptive Field Size** is defined as the maximum size of a neuron can see on the raw image. It grows proportionally as the convolutional neural network goes deep. Fig.1 shows how the receptive fields grows in an exemplar convolutional neural network. It is worth noting that in the definition, we implicitly assume that the receptive field size of a convolutional layer is proportional to the typical size of the spatial relationships of the input patterns. Presented the discussion above, The first constraint:

the c-value of each convolutional layer should be larger than a minimum value t. We empirically found $t = 1/6$ *is a good lower bound of c-value for all convolutional layers.*

2) Necessity of learning - the second constraint: In designing deep convolutional neural network, some times we find arbitrarily add more layers does not help or even worsen

the performance. As the receptive field size grows, new and complex patterns are constantly emerging. However, when the receptive field size reach the image size i.e. neural have seen the entire image region, it stops emerging new and more complex patterns, the driven force for add more layers is no longer exists. Base on the above analysis, we propose the second constraint for designing very deep convolutional neural network:

The receptive field size of the topmost convolutional layer should be no larger than the image size.

This constraint implies that the receptive field of the topmost convolutional layer must be around the entire image region. It shows an upper bound for designing a very deep convolutional neural network.If the receptive field of the topmost layer is much smaller the image region, we can add one more layer to improve our objective (i.e. increase the depth) without violating the constraint. From another perspective, if the receptive field is much smaller the image size, the network will lose the opportunity to learn the high-level patterns or features, which is suboptimal to the performance.

3) The mathematical formulation: We define the notations of the input parameters, herein the image size is z, the filter size is k and the minimum c-value t. The architecture of a deep model can be determined by the total number of stages n and the number of layers in various stage is $\{a_i\}$. Various stages are divided by a down sampling (with stride 2). For example, $n = 3$ and $a_1, a_2, a_3 = 6, 3, 2$ represent a model with 3 stages and the number of convolutional layers in each stage are $6, 3, 2$ layers. In between are down sampling with pooling stride equals 2.

The goal of going deep essentially is to maximize the total number of layers i.e. $\sum_i a_i$, given the two constrains proposed above. **The first constrain** requires the c-values of all layers are no smaller than the minimum c-value t. The real filter size keeps the same in one stage while the receptive field size grows, so the last layer of one stage get the smallest c-value t. Therefore the first constrain is equivalent to ensuring the c-value of the last layer in each stage no smaller than the minimum c-value t:

$$\frac{2^s k}{\sum_i^s 2^{i-1}(k-1)a_i} \geq t \text{ where } s = 1, 2, 3...n \tag{2}$$

Where $2^s k$ is the real filter size at stage s, $\sum_i^s 2^{i-1}(k-1)a_i$ is the receptive field size of the last layer at stage s. t represent the minimum c-value and we set $t = 1/6$ for all tasks. **The second constrain** requires the receptive field size of the topmost convolutional layer is no larger than the entire image region:

$$\sum_i 2^{i-1}(k-1)a_i \leq z \tag{3}$$

Where the left term represent the receptive field of the topmost convolutional layer, $2^{i-1}(k-1)$ is the receptive filed size increment of a layer at i-th stage. $2^{i-1}(k-1)a_i$ is the total receptive increment at i-th stage.

The objective function of the formulation can be formally represented by maximizing the total number of layers, subjecting to the two constraints in eq.(2) and eq.(3):

$$\max_{n,\{a_i\}} \sum a_i \tag{4}$$

$$\sum_i^s 2^{i-1}a_i \leq \frac{2^s k}{t(k-1)} \text{ where } s = 1, 2, 3...n \tag{5}$$

$$\sum_i^s 2^{i-1}a_i \leq \frac{z}{k-1} \tag{6}$$

Where both n and $\{a_i\}$ are integers.

An optimal solution under certain conditions is from eq.(5) and eq.(6), assume the image size is $2^{(m-1)}k/t$, and each stage layers $\{a_i\}$ are relaxed from integers to positive integers. The optimal solution is:

$$n = m \tag{7}$$

$$a_1 = \frac{k}{(k-1)t}, a_2 = ... = a_n = 1/2a_1 \tag{8}$$

Although this optimal solution is obtained under certain conditions, it provides great insights about how to design effective deep architecture under general conditions. First, it guides how many times of down samplings should we choose given the input parameters. Second, it shows the number of layers should be as evenly distributed as possible in all stages, except the first stage. Third, it shows the maximum depth could be achieved by various filter size, based on which we can make better tradeoff between various filter sizes.

Proof of the optimal solution: First, we can easily verify the solution satisfies the constrains in the inequations 5 and 3. Second, assume this is a solution l and $b_1, ..., b_l$, and $l \leq m$. By linearly composing the inequations in 5 and 3, we can show that:

$$\sum_{i=1}^{l} b_i \leq (l+1)C/2 = \sum_{i=1}^{m} a_i \tag{9}$$

where $C = \frac{k}{(k-1)t}$. This inequations implies there is no better solution when $l \leq m$. Third, assume there is a solution l and $b_1, ..., b_l$, and $l \geq m$. By linearly composing the inequations in 5 and 3, we can have

$$\sum_{i=1}^{m} b_i + 2\sum_{i=m+1}^{l} b_i \leq (l+1)C/2 \tag{10}$$

By reorganize the left term, we have

$$\sum_{i=1}^{l} b_i + \sum_{i=m+1}^{l} b_i \leq (l+1)C/2 = \sum_{i=1}^{m} a_i \tag{11}$$

From the inequations above we tell that there is no better solution when $l > m$.

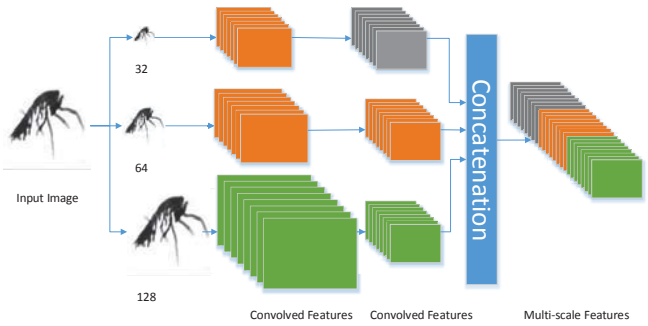


Fig. 2. An illustration of inception module for multi-scale image input architecture.

B. Inception layer for multiscale architecture

The size of all the images in the plankton data set have different sizes. In a convolutional neural network model, it is necessary to scale all the images to a fixed size i.e. 64×64. But it is difficult to decide which size of the input image feature, 32×32 or 64×64 , 96×96, 128×128, etc. In practice we can choose one of the fixed input image size in our deep neural network model, training a deep model individually for different scaled image. While in any case we select the fixed input image size in each model, there are always some image distortion or information loss. For instance, it is possible a image scaled to 64×64 can be correct classified, while wrong classified when scaled to 128×128 in training a deep convolutional neural network, because of information loss or distortion.

Inspired by GooLeNet[11], in order to make good use of the information of the image and minimizing the distortion, we developed a inception module for multi-scale image in one model with a convolutional layer. The architecture of the inception module show in Fig.2. Convolution can change the spatial size of the output volume (size of feature map). We can compute the spatial size of the output volume as a function of the input volume size (W and H represent the with and the hight of the image feature respectively), herein we define the convolutional kernel size (filter size) is K, it represent a K×K filter is applied. The stride is S and the amount of zero padding is P. So the spatial size of the output volume F_w and F_h is:

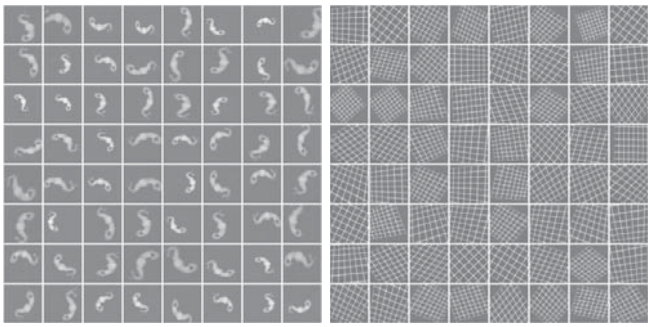
$$F_w = (W - K + 2P)/S + 1 \quad (12)$$

$$F_h = (H - K + 2P)/S + 1 \quad (13)$$

In Fig.2 we set the stride value in first convolutional layer to 1 and 2 for 64 input source and 128 input source respectively, from eq.12 and eq.13, the design of parameters of convolutional layer, we can control the output volume size, so we make the output volume sizes from different sources to the same size. According to this property, it is convenient to deal with times relationship input sources, Table. I gives an example of how to configure a inception module. The last layer of the module we concatenate the output feature maps. The inception module applied to the classification of plankton outperforms single scaled imagery input model, we will discuss the experimental results in section

parameter	Input image size 32×32 and 64×64	Input image size 48×48 and 96×96
filter size	3×3, 2×2	3×3, 5×5
zero-padding	1, 0	1, 2
stride	1, 2	1, 2

TABLE I
CONFIGURATION OF THE CONVOLUTION LAYER IN AN INCEPTION MODULE.



(a) An example of random data augmentation applied to a plankton image (b) Transformation grid

Fig. 3. random transformation, transformation method includes translation, mirror, scaling, rotation, cropping.

C. Data augmentation

We augmented the data for translational and rotational invariant and increased the size of the dataset. Various affine transformation are used and gradually increase intensity of the augmentation is an effective method of reducing over fitting. In plankton dataset, images were created using underwater camera that is towed through an area, the direction of the object in an image can be in any angle. So rotational invariant is necessary, we applied rotation affine to the data augmentation. Data augmentation is applied during training and testing. In testing stage, the output of one image is the average of the augmented results. Fig.3 shows a example of data augmentation and present the transformation methods.

D. Network architecture

In this section, we introduce network architectures designed according to our proposed theory in section II. In this work we divide a deep convolutional neural network model into two parts: feature and classifier parts. when design a deep convolutional neural network, it is common with a stack of convolutional layers (feature part, which has a different depth and width in different models) and followed by 2 or 3 Fully-Connected layers (classifier part) with dropout [12] applied. But we found this design of classifier part is prone to over-fitting if the training set is not sufficiently large. Inspired by Network in Network [9] and GooLeNet [11], we found that it is better to replace the last two fully-connected layers with two convolutional layers with small kernels (6×6, 7×7, or

Input image size 48×48 (model48_5)	Input image size 128×128 (model128_5)
layer id, feature, filter, channel	layer id, feature, filter, channel
conv11, 48×48, 5×5, 16 conv12, 48×48, 5×5, 16 conv13, 48×48, 5×5, 32 conv14, 48×48, 5×5, 64 conv15, 48×48, 5×5, 128 conv16, 48×48, 5×5, 128	conv11, 64×64, 5×5, 16 conv12, 64×64, 5×5, 16 conv13, 64×64, 5×5, 32 conv14, 64×64, 5×5, 64 conv15, 64×64, 5×5, 128 conv16, 64×64, 5×5, 128
max pooling with stride 2	max pooling with stride 2
conv21, 24×24, 5×5, 256 conv22, 24×24, 5×5, 256 conv23, 24×24, 5×5, 256	conv21, 32×32, 5×5, 256 conv22, 32×32, 5×5, 256 conv23, 32×32, 5×5, 256
max pooling with stride 4	max pooling with stride 2
	conv31, 16×16, 5×5, 512 conv32, 16×16, 5×5, 512 conv33, 16×16, 5×5, 512
	max pooling with stride 2
conv31, 6×6, 5×5, 256 conv32, 6×6, 5×5, 256	conv41, 8×8, 5×5, 512 conv42, 8×8, 5×5, 512
fully-connected layer, 121×1	fully-connected layer, 121×1
softmax layer	softmax layer

TABLE II
NETWORK ARCHITECTURE FOR PLANKTON CLASSIFICATION WITH FILTER SIZE 5×5.

Method	Softmax loss	Input image size
VGGNet-16	**0.635**	64
VGGNet-16	0.647	128
VGGNet-16	0.639	96
GooLeNet	0.642	128
GooLeNet	0.651	64
Netwok in Network	0.689	96
model48_5	0.631	48
model64_5	0.633	64
model128_5	0.627	128
Model128_3	**0.613**	128
model112_2	0.621	112
model12864_5	**0.617**	128 & 64
model9648_3	**0.625**	96 & 48

TABLE III
COMPARISON WITH STATE OF THE ART IN PLANKTON CLASSIFICATION (EVALUATED USING SOFTMAX LOSS VALUE). METHOD REPRESENT THE MODEL NAME I.E. MODEL48_5 STANDS FOR PROPOSED MODEL TAKE 48 × 48 IMAGES WITH KERNEL SIZE 5. IMAGE SIZE I.E. 48 MEANS 48×48.

8×8). As kernel size is very large relative to the feature map, the convolutional layer is more like fully-connected layer. One perspective of understanding this design is it conduct dense sliding window test [10]. In Tab. II we present network architectures for plankton classification task and our proposed theory guide us going deep with convolutions.

III. EXPERIMENTAL RESULTS

A. Configuration and Implementation

The implementations are based on the publicly available C++ Caffe toolbox [7] and runs on a GeForce GTX GPU with 5G memory. Weight initialization all set to *xavier* with *std* 0.05.

B. Classification Results on Plankton Classification

Experimental results in Table.III shows the comparison of our proposed models and state-of-the-art models in Plankton classification. From Table. III our proposed model ourperforms state of the art methods in plankton classification. In addition, our proposed *inception module* for multi-scale input proved to be effective i.e. *model12864_5* outperforms *model64_5* and *model128_5*.

IV. CONCLUSION

In this work we proposed a practical theory for designing very deep convolutional neural network and developed a *inception module* for multi-scale input. We cast the design of deep convolutional neural network into a constrained optimization problem and found a optimal solution under certain condition. Our proposed method applied to plankton classification shows the feasibility and effectiveness, in addition, more experiment on cifar10 confirmed our theory is effective. This work provide us a guidance for designing very deep convolutional neural networks. In the future, more experiments will be carried out on large data set e.g. ImageNet [5].

ACKNOWLEDGMENT

This work was supported by National Program on Key Basic Research Project (973 Program, No. 2013CB329502).

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
[2] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 818–833.
[3] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.
[4] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, 2014, pp. 568–576.
[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.
[6] R. K. Cowen, S. Sponaugle, K. Robinson, and J. Luo, "Planktonset 1.0: Plankton imagery data collected from f.g. walton smith in straits of florida from 2014-06-03 to 2014-06-06 and used in the 2015 national data science bowl (ncei accession 0127422)," *Oregon State University; Hatfield Marine Science Center*, vol. Dataset. doi:10.7289/V5D21VJD, 2015.
[7] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.
[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.
[9] C. Q. Lin, M. and S. Yan, "Network in network," *ICLR*, 2014.
[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *arXiv preprint arXiv:1409.4842*, 2014.
[12] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.