An Incremental Deep Learning Network for On-line Unsupervised Feature Extraction

Yu Liang, Yi Yang, Furao Shen(✉), Jinxi Zhao, and Tao Zhu

National Key Laboratory for Novel Software Technology,
Department of Computer Science and Technology,
Collaborative Innovation Center of Novel Software Technology and Industrialization,
Nanjing University, Nanjing, China
frshen@nju.edu.cn

Abstract. In this paper, we propose an incremental deep learning network for on-line unsupervised feature extraction. This deep learning network is based on 3 data processing components: (1) cascaded incremental orthogonal component analysis network (IOCANet); (2) binary hashing; and (3) blockwise histograms. In this architecture, IOCANet can process online data and get filters to do convolutions. Binary hashing is used to enhance the nonlinearity of IOCANet and reduce the quantity of the data. Eventually, the data is encoded by blockwise histograms. Experiments demonstrate that the proposed architecture has potential results for on-line unsupervised feature extraction.

Keywords: Deep learning · On-line unsupervised feature extraction

1 Introduction

Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. Many significant fields such as artificial intelligence, neurobiology are closely related to computer vision. However, similar images may have different lighting conditions, misalignment, non-rigid deformations, occlusion and corruptions, which brings great difficulties to computer vision. Hence, scientists want to extract features from images to overcome the intra-class variability. Representative examples are Gabor features and local binary patterns (LBP) for texture and face classification and SIFT and HOG features for object recognition [3]. These artificial features have achieved great success in corresponding tasks. However, these features can not be adapted to the new conditions.

Deep neural networks (DNNs) are proposed to overcome the weakness of hand-crafted features. The main idea of DNNs is that higher level features can represent more abstract semantics of the data, which means that the intra-class variability will only have little affect upon the features. Therefore these features can achieve great results on image classification. One key ingredient to the success of deep learning in image classification is the use of convolutional

D. Liu et al. (Eds.): ICONIP 2017, Part II, LNCS 10635, pp. 383–392, 2017.
https://doi.org/10.1007/978-3-319-70096-0_40

architectures [1], which is called ConvNet. A convolution operation on small regions of input is introduced to reduce the number of free parameters and improve generalization. One major advantage of convolutional networks is the use of shared weight in convolutional layers, which means that the same filters (weights bank) are used for each pixel in the layer; this both reduce memory footprint and improve performance [4].

Although ConvNet has achieved great success in different vision tasks, it still can not process the online and incremental data. Besides, although ConvNet adopts convolution operation to reduce the number of parameters, it still needs to train a mass of parameters, which takes a lot of time and computing resources.

The initial motivation of our study is to resolve the problem that current deep learning has no way to process online incremental data, which means that if new data comes, our only way is to train all the data again. This would waste a lot of computing resources. What's more, we want to design a simple deep learning network. This network does not need to train a large amount of parameters.

Hence, we propose an Online Incremental Orthogonal Component Analysis Network (IOCANet). The core of our method is to use online incremental orthogonal component analysis method to generate the convolution kernels. After that, like ConvNet, we will do convolution operation to extract features. However, different from ConvNet, IOCANet is simple, but efficient and effective. Compared with ConvNet, it has lower time complexity and both mass data and small amount of data are suitable for IOCANet.

2 Related Work

In recent years, incremental learning has attracted great attention due to the increasing demand for systems have the ability of learning and evolving. When new data is input, incremental learning methods updated the learned model without recalculating the whole model repeatedly. Obviously, these methods enjoy a great advantage: their computational and storage cost is greatly reduced while the performance is improved [2].

Hence, we propose online incremental orthogonal component analysis (IOCA) to deal with incremental learning problem. And in IOCANet, IOCA algorithm is used to generate the convolution kernels (filters), which are the basis in the feature subspace of IOCA. The main principle of IOCA is "entities should not be multiplied unnecessarily". As is shown in Fig. 1 and we give the detailed incremental orthogonal component analysis algorithm in Algorithm 1. In the beginning, feature subspace S is initialized as a zero-dimensional space. Suppose when the t-th data x_t is input, $b_1, b_2, ..., b_k$ are the k basis vectors have been learned, IOCA tries to update S by extracting candidate basis vectors b_{k+1} from x_t and outputs $x_t's$ low-dimensional representation y_t. Then, IOCA continues to process the t+1th data until there is no new data.

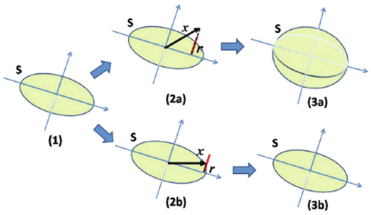


Fig. 1. S is the feature subspace. Input vector x is projected onto S and its complemented subspace $S^{\perp}$. $||r||_2$ measures the linear dependence between x and S. The adaptive threshold T is represented by the red line. (2a) If $||r||_2$ is larger than T, (3a) IOCA will extract a new base vector and enlarge S. (2b) Otherwise, (3b) no new base vector will be extracted and S remains unchanged.

Algorithm 1. Incremental Orthogonal Component Analysis

Initialize basis $B = \varnothing$ and its dimension k = 0.
Initialize $L_{MAX} = 0$
for each input x_i **do**
  if $\|x_i\|_2 > L_{max}$ **then**
    $L_{max}^t = \|x_i\|_2$
  end if
  Let $r_t = x_t$
  for i=1:k **do**
    Compute $y_t^i = r_t^T b_i$, let $y_{t,i}$ be the i-th entry of y_t.
    Compute $r_t^i = r_t - y_{t,i} b_i$
  end for
  Compute $b_{k+1} = \frac{r_t}{\|r_t\|_2}$
  if $\frac{\|r_t\|_2}{L_{max}^{(t)}} \geq f(\frac{f}{d})$ **then**
    Accepted b_{k+1} as a component and let $B^{(k+1)} = [B^{(k)}, b_{k+1}]$.
    Let $y_{t,k+1} = \|r_t\|_2$ be the $(k+1)^{th}$ entry of y_t
    Update basis dimension $k = k + 1$
  end if
end for

The time complexity of IOCA is $O(Ndk)$, N is the training set size, d is the dimension of original data, and k is the number of basis eventually learned by the algorithm. Note that the algorithm of IOCA is concise and its time complexity is low. Therefore, the proposed method enjoys a low computational load and high numerical stability. Because of its simplicity, IOCA has few limits in applications and has the potential to be a universal approach in feature extraction.

PCANet [3] is a simple deep learning network that shares various similarities with IOCANet. PCANet uses PCA algorithm to generate the convolution kernels. PCANet needs all the data input at once and process them together, besides, the number of convolution kernels should be given before the training, which may require a priori information about the data. However, presumably, it

is exactly in the absence of such information. And we always need to attempt some values to decide which value is suitable. For IOCANet, we don't have to worry about this situation. IOCANet will tell you the appropriate number of convolution kernels. Experiment in the fourth section demonstrates this.

3 Proposed Method

As shown in Fig. 2, we use two stages IOCANet as an example to describe the proposed incremental deep learning architecture. In the first stage, we mainly introduce the structural characteristic of each layer in IOCANet. In the second stage, for the reason that layers are similar to each other, we mainly put the focus on the structure between layers. In the output stage, we introduce binary hashing and blockwise histograms to realize non-linear feature extraction.

3.1 Structures of the IOCA Network (IOCANet)

Suppose that we are given N input training images $\{I_i\}_{i=1}^{N}$ of size $m \times n$, and we assume that the patch size (or 2D filter size) is $k_1 \times k_2$ at all stages. Because our algorithm is online and incremental, only one image will be input at a time. Here we assume that I_i is input.

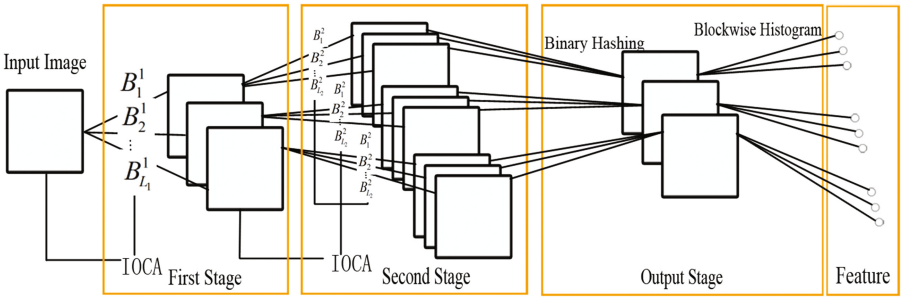


Fig. 2. The overall view of two-stage IOCANet

The First Stage. For input image I_i, around each pixel, we take a $k_1 \times k_2$ patch and we collect all (overlapping) patches, i.e., $x_{i,1}, x_{i,2}, ...x_{i,\widetilde{m}\widetilde{n}}$, where each $x_{i,j}$ denotes the j-th vectorized patch in I_i, $\widetilde{m} = m - k_1 + 1, \widetilde{n} = n - k_2 + 1$. After that we can obtain matrix X

$$X = [x_{i,1}, x_{i,2}, ...x_{i,\widetilde{m}\widetilde{n}}] \in R^{k_1 k_2 \times \widetilde{m}\widetilde{n}} \tag{1}$$

With matrix X, we use incremental orthogonal component analysis to generate the convolution kernel. After that, the basis B captures the main variation of all of the training patches and that is the convolution kernels (or filters) we want to get. Of course, similar to ConvNet, we can stack multiple stages of IOCANet filters to extract higher level features.

The Second Stage. The operation in the second stage is the same as the first stage. Let the l-th filter output of the first stage be:

$$I_i^l = I_i * B_l^1, i = 1, 2, ..., N \tag{2}$$

where * denotes 2D convolution, and the boundary of I_i is zero-padded before convolving with B_l^1, so as to make I_i^l have the same size as I_i. As in the first stage, we collect all of the overlapping patches of I_i^l, and form $Y_i^l = [y_{i,l,1}, y_{i,l,2}, ...y_{i,l,\widetilde{m}\widetilde{n}}] \in R^{k_1 k_2 \times \widetilde{m}\widetilde{n}}$, where each $y_{i,l,j}$ denotes the j-th vectorized patch in I_i^l. Furtherly, we collect all the l-th filters output and define

$$Y = [Y_i^1, Y_i^2, ..., Y_i^{L1}] \tag{3}$$

where L_1 is the num of filters in the first stage. After that, with the IOCA algorithm, the filters of the second stage B_l^2 are then obtained. We set the number of the filters in the second stage to L_2. For each input I_i^l of the second stage, one will output L_2 images of size $m \times n$, and each convolves I_i^l with B_l^2 for $l = 1, 2, 3, ..., L_2$

$$O_i^l = \{I_i^l * B_l^2\}_{l=1}^2 \tag{4}$$

The number of output images at the second stage is $L_1 \times L_2$. If more stages are helpful for us to extract suitable feature, we can repeat the above process to build more stages.

Output Stage (Hashing and Histograms). For the input image I_i, we get L_1 images with the process of the first stage, then repeat the process, we get $L_1 \times L_2$ images with the process of the second stage. In this section, we will reduce the number of images and extract features in an unsupervised manner.

For the images in O_i^l, we binarise these images and obtain $H(O_i^l)$, where $H()$ is a Heaviside step (like) function, whose value is one for positive entries and zero otherwise.

Around each pixel, we view the vector of L_2 binary bits as a decimal number. This converts the L_2 outputs in O_l^l back into a single integer-valued image:

$$T_i^l = \sum_{i=1}^{L2} 2^{l-1} H(I_i^l * B_l^2) \tag{5}$$

Hence, every pixel is an integer in the range $[0, 2^{L_2-1}]$. Besides, the value of the pixel is different from the common number. We treat the L_2 outputs equally and the order and weights are irrelevant, which means that the distance between 0 and 2^{L_2-1} equals to the distance between 0 and 1.

Each of the L_1 images $T_i^l, l = 1, 2, ..., L_1$, is partitioned into N blocks, for each blocks, we compute the histogram (with 2^{L_2} bins) of the decimal values and then concatenate all N blocks into one vector and denote as $hist(T_i^l)$. After that, we will concatenate all L_1 images into one vector:

$$f_i = [hist(T_i^1), hist(T_i^2), ..., hist(T_i^{L^1})] \tag{6}$$

The local blocks can be either overlapping or non-overlapping, it depends on the input data, we will walk through this in detail in an upcoming section.

The parameters of the IOCANet include the filter size k_1, k_2, the number of stages, and the block size for local histograms in the output layer. The number of filters in each stage L_1, L_2 can be decided by IOCA automatically.

3.2 Computational Complexity

In this section, we give you the computational complexity of IOCANet. We take two-stage IOCANet-2 as an example. For an input image, it has $\widetilde{m} \times \widetilde{n}$ patches and the patches size is $k_1 \times k_2$. Hence, in the first stage, the complexity of IOCA is $O(k_1 k_2 \widetilde{m}\widetilde{n} L_1)$. After that, the complexity of convolution operation is $O(L_1 k_1 k_2 \widetilde{m}\widetilde{n})$, therefore, the overall computational complexity of first stage is $O(L_1 k_1 k_2 \widetilde{m}\widetilde{n})$. In the second stage, the number of input images is L_1 and the computational complexity of second stage is $O(L_1 L_2 k_1 k_2 \widetilde{m}\widetilde{n})$. In the output stage, the complexity of binary hashing is $O(L_2 mn)$, and the naive histogram operation is of complexity $mnL_2 \frac{1}{1-BOR} log2$, where BOR is block overlap ratio and the range of BOR is 0 to 1. The overall computational complexity of IOCANet is

$$O(mnk_1 k_2 L_1 L_2 N)$$

where $m \times n$ is the size of input images, $k_1 \times k_2$ is the size of patches, L_1 is the number of filters in the first stages and L_2 is the number of filters in the second stages. N is the number of images in input to the network in an on-line way.

Beyond that, the space complexity of IOCANet is low. After we extract features from an input image, we throw the input image away to make room for the next image, which means that the space complexity of IOCANet is independent of the number of images in the data set. That is particulary suited for big data.

4 Experiment

In this section, we first explore how the proposed IOCANet performs in handwritten digit recognition tasks. Then we do the experiment on face recognition, which would explain the performance of IOCANet on different tasks. Here we introduce RandNet to illustrate the effectiveness of IOCANet. Compared with IOCANet, RandNet replaces the IOCA filters with completely random filters.

4.1 Digit Recognition on MNIST Datasets

The MNIST database (Mixed National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. We use this database to measure the effectiveness of IOCANet on mass data.

Databases. The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. These examples are 28×28 grayscale images of handwritten digits 0–9.

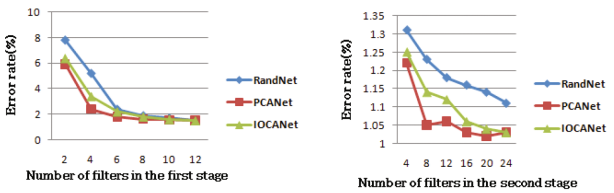
Classifiers. A linear SVM classifier is used in this section.

Number of Filters. In this section, experiments will tell whether the number of filters decided by IOCANet is suitable. The block size in this experiment is 7×7, the block overlap ratio is 0.5, and the filter size is 7×7.

Firstly, for the one-stage network, we get the result that $L_1 = 8$ automatically. Regarding the two-stage networks, we can get the result that $L_1 = 8, L_2 = 16$.

After that, we remove the threshold $f(\frac{f}{d})$, with this threshold, we can determine whether a new filter is needed and get the number of filters automatically. Then we vary the number of filters in the one-stage networks IOCANet-1 from 2 to 12. Regarding the two-stage networks IOCANet-2, we set $L_1 = 8$ and change L_2 from 4 to 24. The result is shown in Fig. 3.

We can see that in IOCANet-1, $L_1 = 8$ is suitable for the reason that more filters will only bring little benefit, which is in tune with the number of filters determined automatically by our algorithm. Regarding the two-stage networks, $L_1 = 8, L_2 = 16$ is appropriate too. What's more, Fig. 3 illustrates that the number of filters decided by IOCANet also is reasonable for RandNet and PCANet, which validates the effectiveness of our algorithm.



(a) (b)

Fig. 3. Error rate of RandNet, PCANet and IOCANet on MNIST test set. For (a), we vary the number of filters in the first stage from 2 to 12. For (b), we set $L_1 = 8$ and vary the number of filters in the second stage from 2 to 12

Comparison with State of the Art. We compare IOCANet with RandNet, PCANet, ConvNet and other state of the art methods. Regarding the parameters of RandNet, we set the block size to 7×7. In PCANet, we set the filter size 7×7 and the number of PCA filters $L_1 = L_2 = 8$. The block overlap ratio is set to 0.5.

Table 1. Comparison of error rates(%) of the methods on MNIST test set

Methods	MNIST	Methods	MNIST
HSC [5]	0.77	RandNet-1	1.32
K-NN-SCM [6]	0.63	RandNet-2	0.63
K-NN-IDM [7]	0.54	PCANet-1	0.94
CDBN [8]	0.82	PCANet-2	0.66
Stochastic pooling ConvNet [9]	0.47	IOCANet-1	2.98
Conv. Maxout+Dropout [10]	0.45	IOCANet-2	0.92

The testing error rates of the various methods on MNIST are shown in Table 1. We know that the best result is 0.23% [4]. We see from the table that IOCANet is comparable with the state-of-the-art methods on this standard MNIST task. However, IOCANet does not need to set the number of filters on every stage, and if new training images input, PCANet needs to combine original training set and the new training set, then redo the experiment. For IOCANet, we only need to input the images one by one, then we can get the feature and do the classification.

4.2 Face Recognition on ORL Dataset

In this section, we do experiment on ORL dataset to explore how IOCANet performs in face verification task. And for the reason that ORL dataset has 400 images, we use this dataset to check the performance of IOCANet on small amount of data.

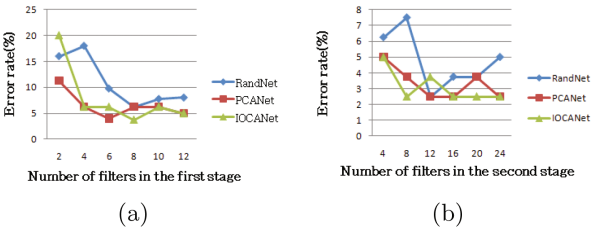
Databases. ORL dataset contains 40 people and per person has 10 images. The image size is 112×92. All the images are frontal and slight tilt of the head. For every person, we randomly choose 8 images as training images and the remaining images as testing images.

Classifiers. A nearest neighbor (NN) classifier is employed in this section. Linear SVM are not selected because every person only has few images, which may not be suitable for Linear SVM classifier.

Number of Filters. The effectiveness of the number of filters decided by IOCANet is studied here. The filter size of the network is 17×17. The block size is 19×19 and the block overlap ratio is 0.

For one stage network IOCANet-1, with out algorithm, we can get the number of filters in the first stage $L_1 = 8$. Regarding the two-stage networks, we can get that $L_1 = 8, L_2 = 18$.

After that, we vary number of filters in the first stage L_1 from 2 to 12. Regarding two stage network IOCANet-2, we set $L_1 = 8$, change L_2 from 4 to



(a) (b)

Fig. 4. Error rates of RandNet, PCANet and IOCANet on ORL test set. For (a), we vary the number of filters in the first stage from 2 to 12. For (b), we set $L_1 = 8$ and vary the number of filters in the second stage from 4 to 24

24. The result is shown in Fig. 4. We can see that $L_1 = 8$ is suitable for IOCANet-1, which equals the number of filters decided by our method. For IOCANet-2, $L_1 = 8, L_2 = 18$ be slightly more than the best result, but it still is appropriate for the reason that it has almost no effect on the performance of IOCANet.

Comparison with State of the Art. We compare IOCANet with RandNet and PCANet. We set the parameters of PCANet to the filter size 17×17, the number of filters $L_1 = 8, L_2 = 12$ and 19×19 block size. The filters size of IOCANet is 17×17. Besides, the block size of IOCANet is 19×19.

The performances of all methods are given in Table 2. One can observe that PCANet and IOCANet achieve similar result, but both PCANet and IOCANet perform better than RandNet. What's more, two-stages network also performs better than one-stage network.

Table 2. Comparison of error rates(%) of the methods on ORL test set

Methods	Orl	Methods	Orl
RandNet-1	6.25	PCANet-2	2.5
RandNet-2	2.5	IOCANet-1	3.75
PCANet-1	5	IOCANet-2	2.5

5 Conclusion

In this paper, we proposed an incremental deep learning network for online unsupervised feature extraction – IOCANet. IOCANet is a simple deep learning network, it only has few parameters must be given to and some of parameters can be decided by itself. When the parameters are set, this algorithm is simple and effective. IOCANet has low time complexity, which means that IOCA has few limits in applications and has the potential to be a universal approach. What's more, IOCANet is able to keep learning from on-line and incremental

data, which means that if new training data input, IOCANet can extract features directly. For classical feature extraction algorithms such as ConvNet, they need to combine original training set and the new training set, then redo the training process, which wastes large amount of computing resources. Besides, both mass data and small amount of data are suitable for IOCANet.

The experiments demonstrate that IOCANet can achieve reasonable result for digit recognition and face recognition. Although the performance of the other methods sometimes is slightly better in certain aspect, IOCANet fulfills simple architecture, reasonable classification result and low time complexity. When the learning is online and incremental, the performance of IOCA is outstanding.

Acknowledgments. This work is supported in part by the National Science Foundation of China under Grant Nos. (61373130, 61375064, 61373001), and Jiangsu NSF grant (BK20141319).

References

1. Kavukcuoglu, K., Sermanet, P., Boureau, Y.L., et al.: Learning convolutional feature hierarchies for visual recognition. In: International Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, pp. 1090–1098. Curran Associates Inc. (2010)
2. Tao, Z., Ye, X., Furao, S., et al.: An online incremental orthogonal component analysis method for dimensionality reduction. Neural Netw. **85**, 33–50 (2016)
3. Chan, T.H., Jia, K., Gao, S., et al.: PCANet: a simple deep learning baseline for image classification? IEEE Trans. Image Process. **24**(12), 5017–5032 (2015). A Publication of the IEEE Signal Processing Society
4. Dan, C., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 3642–3649. IEEE Computer Society (2012)
5. Yu, K., Lin, Y., Lafferty, J.: Learning image representations from the pixel level via hierarchical sparse coding. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1713–1720. IEEE Computer Society (2011)
6. Salve, S.G., Jondhale, K.C.: Shape matching and object recognition using shape contexts. In: IEEE International Conference on Computer Science and Information Technology, pp. 471–474. IEEE (2010)
7. Keysers, D., Deselaers, T., Gollan, C., et al.: Deformation models for image recognition. IEEE Trans. Pattern Anal. Mach. Intell. **29**, 1422–1435 (2007)
8. Lee, H., Grosse, R., Ranganath, R., et al.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: International Conference on Machine Learning, pp. 609–616. ACM (2009)
9. Zeiler, M.D., Fergus, R.: Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. Eprint Arxiv (2013)
10. Goodfellow, I.J., Wardefarley, D., Mirza, M., et al.: Maxout networks. Comput. Sci. **28**, 1319–1327 (2013)