Stacked Local Autocorrelation Features

Hideki NAKAYAMA[1,a)]

1. Introduction

Extracting informative image features has always been one of the most important topics in visual recognition. Since it was proposed in 1988, higher-order local autocorrelation (HLAC) features [20] have been known for their exceptional balance between performance and computational efficiency and have been successfully used for many visual recognition tasks including face and gesture recognition [11], [23], anomaly detection [19], and image annotation [18]. However, to improve the recognition performance of HLAC, it is necessary to use correlations with a much higher order as well as large masks, making the number of features exponentially large and implementation infeasible. For this reason, despite its name, HLAC has been used with relatively low-order correlations. This problem comes from the inherent "shallow" architecture of HLAC. In recent years, HLAC has generally been outperformed by more sophisticated hand-crafted features encoding high-level nonlinearities such as bag-of-visual-words variants [4], [21].

We propose here an efficient method to exploit higher-order correlation information by hierarchically stacking low-order local autocorrelations, which we named the SLAC method. The method was motivated by the recent remarkable success achieved in deep learning [9]. The key idea in our method is to decompose the computation of higher-order correlations into a multi-layer network rather than computing everything in a single input layer as the original HLAC does. Experimental results show that SLAC represents powerful discriminative information with a moderately sized feature vector compared to traditional HLAC. Moreover, it is shown that the SLAC scheme can be applied not only to raw pixels but also to generic image descriptors.

2. Related Work and Our Approach

2.1 Higher-order Local Autocorrelation (HLAC)

HLAC features [20] consist of the autocorrelations of neighboring pixels within an image. The d-th order HLAC features can be represented as followed.

$$f_d(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_d) = \int I(\boldsymbol{r}) \cdots I(\boldsymbol{r} + \boldsymbol{a}_d) d\boldsymbol{r}, \qquad (1)$$

where $\boldsymbol{r}$ is a reference point, $I(\boldsymbol{r})$ is its pixel value, and $\{\boldsymbol{a}_1, \ldots \boldsymbol{a}_d\}$ are displacement vectors defining neighbors. Figure 1(a) shows an example when $d = 2$. The pattern formed by $\boldsymbol{r}$ and $\boldsymbol{a}$ is called a "mask pattern" (Fig. 1(b)). For c-channel inputs (*e.g.*, c=3 for color images), we also consider the combination of channels in mask patterns without duplications [10]. The width (number of pixels) of the unit displacement vector is called the mask width; it is denoted by δ and should be tuned properly.

The HLAC features consisting of autocorrelations up to the d-th order are called "at most the d-th order HLAC" features. If we take a larger d, their representation ability increases; however, the computation cost becomes substantially larger. Whereas the dimensions of the first-order and second-order color-HLACs are 45 and 739, the third-order ones become 8023- and 153,115-dimensional with mask sizes of three and five, respectively. Generally, it is not easy to handle feature vectors of this size. Therefore, at most, the first- or second-order HLAC features with mask size three have generally been used in practice.

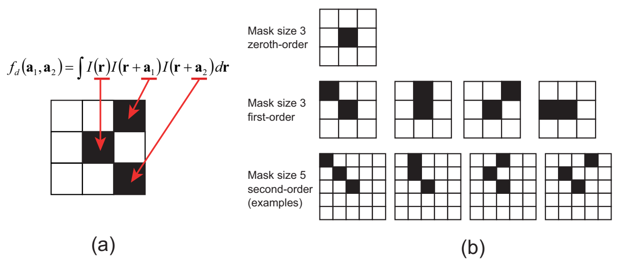
$f_d(\mathbf{a}_1,\mathbf{a}_2)=\int I(\mathbf{r})I(\mathbf{r}+\mathbf{a}_1)I(\mathbf{r}+\mathbf{a}_2)d\mathbf{r}$
Mask size 3 zeroth-order
Mask size 3 first-order
Mask size 5 second-order (examples)
(a)
(b)

Fig. 1 (a) Mask pattern corresponding to specific displacement vectors. (b) Examples of HLAC mask patterns.

2.2 Deep Learning by Stacking

Deep convolutional neural networks (DCNNs) have recently achieved surprisingly high performance and are now the state-of-the-art methodology used for visual recognition [3], [13], [14], [15]. These works have proved that deep architectures are fundamentally important to extract rich discriminative information and that traditional features are too shallow to do this. However, optimizing deep networks is not easy for non-experts because they are sensitive to numerous hyperparameters that should be carefully tuned [2].

Meanwhile, deep architectures have also been studied in traditional object recognition pipelines. For example, HyperFeatures [1] and Deep Fisher Networks [24] stack bag-of-words [4] based feature extraction layers multiple times

[1] Graduate School of Information Science and Technology, The University of Tokyo, Hongo 7–3–1, Bunkyo-ku, Tokyo, 113–8656 Japan
[a)] nakayama@ci.i.u-tokyo.ac.jp

to encode more discriminative information. These methods construct each layer one by one in a feed-forward manner, which is somewhat analogous to the unsupervised pretraining stage in deep learning [9]. Although their networks are thought to be suboptimal in the sense that they do not perform global optimization through the entire network, they achieve good performance and can be stably constructed with a significantly lower computational cost compared to DCNNs [24].

The success of these studies motivated us to develop SLAC; its specific objective is to ”stack” local autocorrelation layers in order to extract rich information related to truly higher-order correlations of inputs in a more efficient manner with the help of deep structures. Indeed, HLAC can be interpreted as an instance of a ”shallow” sum-product network (SPN) [22], which is a deep network consisting of sum layers and product layers where the sums and products of input variables are respectively passed to the next layer. Namely, responses of mask patterns at each local point are computed in the first product layer and then averaged over the entire image in the following sum layer (Figure 2). Unlike many deep learning methods, the theoretical basis of this network has been revealed relatively well. It has been pointed out that for representing a model with a fixed complexity, shallow SPN generally requires exponentially more hidden nodes compared to deep SPN [5]. This fact suggests the importance of deepness in sum-product models including HLAC and SLAC. While the original HLAC features, which are inherently shallow, become exponentially large to attain rich information related to higher-order correlations, SLAC features are expected to be efficiently approximated by repetitively computing low-order correlations.

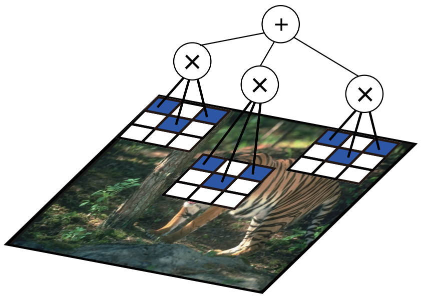

Fig. 2 Sum-product architecture of HLAC.

3. Stacking Local Autocorrelations

Figure 3 illustrates the overall architecture of SLAC. Our idea is quite simple and straightforward. SLAC is the iteration of the two basic operations below.

- Compute low-order (at most, the first- or second-order) local autocorrelations (LAC) for each small local region within inputs.
- Compress the obtained local feature vectors into a moderate size via principal component analysis (PCA) so that computation of correlations in the next layer is feasible while retaining the essential information in that layer.

This procedure can be repeated multiple times in order to gradually capture global higher-order correlation information in higher layers. Finally, we place a logistic regression classifier taking all responses of the final layer as the inputs.

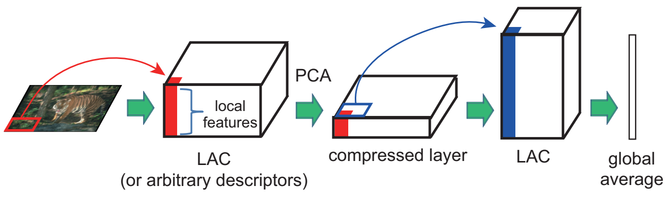

Fig. 3 SLAC architecture with two LAC layers.

4. Experiment

4.1 Experimental Setup

We use three datasets in our experiment: CIFAR-10 [12], MNIST [16], and Caltech-101 [6]. CIFAR-10 and MNIST have been flagship datasets for deep learning problems. CIFAR-10 is a dataset of tiny color images (32x32 pixels) spanning ten object classes. It contains 5,000 training and 1,000 testing samples per class. MNIST consists of tiny gray images (28x28 pixels) of handwritten digits (0 to 9), and has roughly 6,000 training and 1,000 testing samples specified per class. For the input layer, we used color ($c = 3$) (H)LAC for CIFAR-10 and gray ($c = 1$) for MNIST.

Caltech-101 has been a de-facto standard dataset for object recognition. We used 30 training samples per class following the standard experimental protocol. For this dataset, also tested dense SIFT features as the input to show that the SLAC framework is applicable to generic descriptors. Note that applying an HLAC-like scheme to generic descriptors was proposed by [8], [17] and is not new. The objective here is to show that this methodology can also be boosted by the same SLAC framework.

We describe our SLAC network by a concatenation of the operations starting from the bottom layer to the top layer, where each operation is defined as follows.

- (H)LAC(d,m,δ,c,n): At most the d-th order LAC features with mask size m and mask width δ extracted from a c-channel $n \times n$ patch. The letter ”G” in the n position means that features are extracted globally. For clarification, we use the notation ”HLAC” for the standard implementation of HLAC features [20] and ”LAC” for any single layer in SLAC.
- SIFT(n,s): SIFT descriptors of size $n \times n$ are densely extracted from raw images spacing s pixels.
- PCA(k): PCA compression into k variables.

For example, LAC(2,3,1,3,4)-PCA(16)-LAC(1,3,1,16,G) means that we extract at most the second-order ($d = 2$) LAC with mask size $m = 3$ and mask width $\delta = 1$ from each 4x4 patch of $c = 3$ images (raw color images), compress them into $k = 16$ dimensions using PCA, and further compute at most the first-order LAC globally.

4.2 Results

Table 1 summarizes the performance of SLAC compared to the original HLAC on CIFAR-10. For each method, we set optimal mask width δ, which was experimentally tuned. Among the baseline HLACs, the third-order one (1-c) showed the best performance. When we start from the first-order LAC, which is the same descriptor as that of (1-a), our method with the second LAC layer (1-e) achieved better performance, although the dimension of the final feature vector was substantially smaller. Similar results were observed with MNIST (Table 2). Our method (2-e) performed better than high-dimensional HLACs (2-d).

Further deepening the model might produce better results; however, we found this to be difficult with these datasets because the images are too small. Therefore, we tested deeper models on Caltech-101 (Table 3). We observed that increasing the number of LAC layers generally improved the performance while keeping the size of the resulting feature vectors the same, although adding the fourth LAC layer (3-f,i) did not improve the performance.

Table 4 presents a comparison using SIFT descriptors as inputs. (4-a) and (4-b) correspond to the SIFT and PCA based LAC features, the idea of which was proposed by [8], and they serve as the baseline here. We also show the performance of the standard bag-of-visual-words method [4] using the same SIFT descriptors and 4000 visual words (4-c). We successfully improved the performance from the single-LAC baselines ((4-a) to (4-e,g), (4-b) to (4-h)) as the number of LAC layers was increased and achieved better than the bag-of-words baseline. Because SIFT descriptors are high-dimensional ($c = 128$), it is almost impossible to compute higher-order correlations in one layer as the original HLAC does. In such cases, the SLAC method is more suitable and could be a powerful tool to model higher-order statistics.

Finally, we combined our methodology with the Fisher vector framework (Table 5). We implemeted Fisher vector with a GMM of 64 Gaussians. The baseline Fisher vector (5-a) achieves 58.3% which is much better than the best scores of SLAC. However, we observed that putting the Fisher feature extractor on top of the SLAC network result in a moderate performance improvement. Moreover, combining the original Fisher vector and the hybrid ones in late fusion substantially boosted performance. This result indicates that out method can extract different statistical properties of local features and is complementary to the standard Fisher vector.

5. Conclusion

The recent success in deep learning motivated us to propose a novel method called SLAC to efficiently exploit higher-order correlations of inputs in a moderately sized feature vector. SLAC builds a network in a deterministic feed-forward approach based on simple eigenvalue problems, so it does not require a substantial training cost; it can be easily trained on a single CPU. Moreover, once the network is built, we can achieve high-speed feature extraction inherited from HLAC. In addition, our method can also be applied to generic descriptors such as SIFT. The results obtained in experiments are encouraging and are thought to suggest a new direction in representation learning.

SLAC is interpreted as a greedy layer-wise pre-training of sum-product networks. Therefore, our future task is to experiment with a discriminative fine-tuning approach [7] to see if we can further improve the performance.

References

[1] Agarwal, A. and Triggs, B.: Hyperfeatures -Multilevel local coding for visual recognition, *Proc. ECCV* (2006).

[2] Bengio, Y.: Practical recommendations for gradient-based training of deep architectures, *Neural Networks: Tricks of the Trade* (2012).

[3] Ciresan, D., Meier, U. and Schmidhuber, J.: Multi-column deep neural networks for image classification, *Proc. IEEE CVPR* (2012).

[4] Csurka, G., Dance, C. R., Fan, L., Willamowski, J. and Bray, C.: Visual categorization with bags of keypoints, *Proc. ECCV Workshop on Statistical Learning in Computer Vision* (2004).

[5] Delalleau, O. and Bengio, Y.: Shallow vs. deep sum-product networks, *Proc. NIPS* (2011).

[6] Fei-Fei, L., Fergus, R. and Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories, *Journal of Computer Vision and Image Understanding*, Vol. 106, No. 1, pp. 59–70 (2007).

[7] Gens, R. and Domingos, P.: Discriminative learning of sum-product networks, *Proc. NIPS* (2012).

[8] Harada, T., Nakayama, H. and Kuniyoshi, Y.: Improving local descriptors by embedding global and local spatial information, *Proc. ECCV* (2010).

[9] Hinton, G. E. and Salakhutdinov, R. R.: Reducing the dimensionality of data with neural networks, *Science*, Vol. 313, pp. 504–507 (2006).

[10] Kato, T., Kurita, T., Otsu, N. and Hirata, K.: A sketch retrieval method for full color image database -query by visual example-, *Proc. ICPR*, pp. 213–216 (1992).

[11] Kobayashi, T. and Otsu, N.: Action and simultaneous multiple-person identification using cubic higher-order local auto-correlation, *Proc. ICPR*, pp. 741–744 (2004).

[12] Krizhevsky, A.: Learning multiple layers of features from tiny images, Master's thesis, Toronto University (2009).

[13] Krizhevsky, A., Sutskever, I. and Hinton, G. E.: ImageNet classification with deep convolutional neural networks, *Proc. NIPS* (2012).

[14] Le, Q. V., Ranzato, M. A., Monga, R., Devin, M., Chen, K., Corrado, G. S. and Ng, A. Y.: Building high-level features using large scale unsupervised learning, *Proc. ICML* (2012).

[15] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P.: Gradient-based learning applied to document recognition, *Proc. of the IEEE* (1998).

[16] LeCun, Y.: The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`.

[17] Nakayama, H., Harada, T. and Kuniyoshi, Y.: Dense sampling low-level statistics of local features, *Proc. CIVR* (2009).

[18] Nakayama, H., Harada, T. and Kuniyoshi, Y.: Image annotation and retrieval based on efficient learning of contextual latent space, *Proc. ICME*, pp. 858–861 (2009).

[19] Nanri, T. and Otsu, N.: Unsupervised abnormality detection in video surveillance, *Proc. MVA*, pp. 574–577 (2005).

[20] Otsu, N. and Kurita, T.: A new scheme for practical, flexible and intelligent vision systems, *Proc. IAPR Workshop on Computer Vision* (1988).

[21] Perronnin, F., Sánchez, J. and Mensink, T.: Improving the Fisher kernel for large-scale image classification, *Proc. ECCV* (2010).

[22] Poon, H. and Domingos, P.: Sum-product networks: A new deep architecture, *Proc. UAI* (2011).

[23] Shinohara, Y. and Otsu, N.: Facial expression recognition using Fisher weight maps, *IEEE FG* (2004).

[24] Simonyan, K., Vedaldi, A. and Zisserman, A.: Deep Fisher networks for large-scale image classification, *Proc. NIPS* (2012).

Table 1 Classification performance on CIFAR-10 dataset (%).

No.	Description of network	Dims.	Acc. (%)
(1-a)	HLAC(1,3,2,3,G)	45	42.47
(1-b)	HLAC(2,3,2,3,G)	739	55.25
(1-c)	HLAC(3,3,2,3,G)	8023	61.49
(1-d)	HLAC(2,5,2,3,G)	5419	59.82
	SLAC (Ours)		
(1-e)	LAC(1,3,2,3,4)-PCA(16)-LAC(1,3,2,16,G)	1176	**63.24**

Table 2 Classification performance on MNIST dataset (%).

No.	Description of network	Dims.	Acc. (%)
(2-a)	HLAC(2,3,3,1,G)	35	89.10
(2-b)	HLAC(3,3,3,1,G)	153	95.95
(2-c)	HLAC(2,5,2,1,G)	219	96.86
(2-d)	HLAC(3,5,3,1,G)	2245	98.61
	SLAC (Ours)		
(2-e)	LAC(2,3,2,2,4)-PCA(16)-LAC(1,3,2,16,G)	1176	**98.88**
(2-f)	LAC(2,3,1,2,4)-PCA(16)-LAC(1,3,1,16,3)-PCA(16)-LAC(1,3,1,16,G)	1176	98.49

Table 3 Classification performance on Caltech-101 dataset (%). Color (H)LAC features were used for inputs.

No.	Description of network	Dims.	Acc. (%)
(3-a)	HLAC(1,3,8,3,G)	45	21.3
(3-b)	HLAC(2,3,8,3,G)	739	29.5
(3-c)	HLAC(3,3,8,3,G)	8023	32.0
	SLAC (Ours)		
(3-d)	LAC(1,3,3,3,4)-PCA(16)-LAC(1,3,3,16,G)	1176	31.8
(3-e)	LAC(1,3,3,3,4)-PCA(16)-LAC(1,3,3,16,2)-PCA(16)-LAC(1,3,2,16,G)	1176	**35.2**
(3-f)	LAC(1,3,3,3,4)-PCA(16)-LAC(1,3,3,16,2)-PCA(16)-LAC(1,3,2,16,2)-PCA(16)-LAC(1,3,2,16,G)	1176	**35.2**
(3-g)	LAC(1,3,3,3,4)-PCA(24)-LAC(1,3,3,24,G)	2628	31.5
(3-h)	LAC(1,3,3,3,4)-PCA(24)-LAC(1,3,3,24,2)-PCA(24)-LAC(1,3,2,24,G)	2628	**35.0**
(3-i)	LAC(1,3,3,3,4)-PCA(24)-LAC(1,3,3,24,2)-PCA(24)-LAC(1,3,2,24,2)-PCA(24)-LAC(1,3,2,24,G)	2628	**35.4**

Table 4 Classification performance on Caltech-101 dataset (%). Dense SIFT descriptors were used for inputs.

No.	Description of network	Dims.	Acc. (%)
(4-a)	SIFT(24,4)-PCA(16)-LAC(1,3,4,16,G)	1176	46.3
(4-b)	SIFT(24,4)-PCA(24)-LAC(1,3,4,24,G)	2628	47.3
(4-c)	SIFT(24,4)-BoW(4000) (with histogram intersection kernel and SVM)	4000	47.4
	SLAC (Ours)		
(4-d)	SIFT(24,4)-PCA(16)-LAC(1,3,1,16,2)-PCA(16)-LAC(1,3,1,16,G)	1176	45.0
(4-e)	SIFT(24,4)-PCA(16)-LAC(1,3,1,16,2)-PCA(16)-LAC(1,3,1,16,2)-PCA(16)-LAC(1,3,1,16,G)	1176	48.4
(4-f)	SIFT(24,4)-PCA(24)-LAC(1,3,1,24,2)-PCA(24)-LAC(1,3,1,24,G)	2628	47.3
(4-g)	SIFT(24,4)-PCA(24)-LAC(1,3,1,24,2)-PCA(24)-LAC(1,3,1,24,2)-PCA(16)-LAC(1,3,1,16,G)	1176	**50.3**
(4-h)	SIFT(24,4)-PCA(24)-LAC(1,3,1,24,2)-PCA(24)-LAC(1,3,1,24,2)-PCA(24)-LAC(1,3,1,24,G)	2628	**52.1**

Table 5 Combining SLAC and Fisher vector frameworks. Classification performance on Caltech-101 dataset (%).

No.	Description of network	Acc. (%)
(5-a)	SIFT(24,4)-PCA(64)-Fisher (baseline)	58.6
(5-b)	SIFT(24,4)-PCA(16)-LAC(1,3,4,16,2)-PCA(64)-Fisher	58.9
(5-c)	SIFT(24,4)-PCA(16)-LAC(1,3,4,16,2)-PCA(16)-LAC(1,3,1,16,2)-PCA(64)-Fisher	54.3
	(5-a) + (5-b) late fusion	**63.7**
	(5-a) + (5-b) + (5-c) late fusion	**65.6**