Reversed Spectral Hashing

Qingshan Liu, Senior Member, IEEE, Guangcan Liu, Member, IEEE, Lai Li, Xiao-Tong Yuan, Member, IEEE, Meng Wang, and Wei Liu, Member, IEEE

Abstract—Hashing is emerging as a powerful tool for building highly efficient indices in large-scale search systems. In this paper, we study spectral hashing (SH), which is a classical method of unsupervised hashing. In general, SH solves for the hash codes by minimizing an objective function that tries to preserve the similarity structure of the data given. Although computationally simple, very often SH performs unsatisfactorily and lags distinctly behind the state-of-the-art methods. We observe that the inferior performance of SH is mainly due to its imperfect formulation; that is, the optimization of the minimization problem in SH actually cannot ensure that the similarity structure of the high-dimensional data is really preserved in the low-dimensional hash code space. In this paper, we, therefore, introduce reversed SH (ReSH), which is SH with its input and output interchanged. Unlike SH, which estimates the similarity structure from the given high-dimensional data, our ReSH defines the similarities between data points according to the unknown low-dimensional hash codes. Equipped with such a reversal mechanism, ReSH can seamlessly overcome the drawback of SH. More precisely, the minimization problem in our ReSH can be optimized if and only if similar data points are mapped to adjacent hash codes, and mostly important, dissimilar data points are considerably separated from each other in the code space. Finally, we solve the minimization problem in ReSH by multilayer neural networks and obtain state-of-the-art retrieval results on three benchmark data sets.

Index Terms—Neural networks, spectral methods, unsupervised hashing.

I. INTRODUCTION

IN THESE years, the volume of feature-rich data (e.g., photographs and videos) gets bigger and bigger, and thus, it is very crucial to develop the techniques of fast $k$-nearest neighbor (KNN) search in gigantic data sets, particularly the techniques of building similarity indices of high retrieval efficiency. Among various approaches for building efficient indices and performing fast KNN search, hashing is widely regarded as a promising one due to its advantages of fast query speed and small storage expenses [1]–[6]. According to how much label information is used, the existing hashing methods can be divided into three categories: unsupervised [7], [8], supervised [9], [10], and semisupervised [11]. Benefiting from the advancement of deep neural networks, the performance of supervised hashing methods has been boosted a lot in recent years [12], [13]. However, supervised methods are inapplicable to the environments where label information is lacking. In this paper, we shall, therefore, study unsupervised hashing, aiming to establish a desirable method that is useful for building indices of high retrieval efficiency in large-scale search systems.

A. Related Work

1) Spectral Hashing: We are particularly interested in spectral hashing (SH) [14]–[16], a classical method of unsupervised hashing. Given a real-valued data matrix $X = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^{d \times n}$, each column in which is a $d$-dimensional data point $x_j \in \mathbb{R}^d$, SH tries to learn a matrix of hash codes, denoted as $Y = \{y_1, y_2, \ldots, y_n\}$, by minimizing

$$\min_Y \sum_{i,j=1}^{n} a_{ij} \|y_i - y_j\|^2$$

s.t. $Y \in \{-1, 1\}^{k \times n}$, $YY^T = kI$, $Y1 = 0$ (1)

where $\| \cdot \|$ denotes the $\ell_2$ norm of a vector, $y_j \in \{-1, +1\}^k$, the $j$th column of $Y$, is the $k$-dimensional hash code one wants to learn for the $i$th data point, $x_j \in \mathbb{R}^d$, $1$ denotes a vector of all ones, and $A = [a_{ij}]_{n \times n} \in \mathbb{R}^{n \times n}$ is a predefined affinity matrix that encodes the similarity structure of the data. The last two constraints, $YY^T = kI$ and $Y1 = 0$, are to achieve uncorrelated and balanced hash bits, respectively. Usually, the affinity matrix $A$ is computed by

$$a_{ij} = \exp\left( -\frac{\|x_i - x_j\|^2}{2\sigma^2} \right)$$

(2)

where $\sigma > 0$ is a thresholding parameter. By discarding the binary constraint, $Y \in \{-1, 1\}^{k \times n}$, the minimization problem in (1) can be easily solved by performing an eigenvalue decomposition on the Laplacian matrix of $A$. To obtain hash codes of strictly binary, after the code matrix $Y$ is learned, SH applies a simple quantization operator $h(\cdot)$ on the elements of $y_j$s, where $h(v)$ takes value 1 if $v \geq 0$ and 0 otherwise.
Although computationally simple, very often SH performs unsatisfactorily and lags distinctly behind the recently established state-of-the-art methods, e.g., iterative quantization (ITQ) [17]. The main reason is that the formulation of SH is indeed imperfect: The minimization of $\sum_{i,j=1}^{n} a_{ij} \| y_i - y_j \|^2$, in general, tends to assign similar hash codes to all data points, including the dissimilar ones, resulting in inferior retrieval performance. This issue is rather striking when the binary constraint $Y \in \{-1, 1\}^{k \times n}$ is discarded during the spectral relaxation procedure.

2) Discrete Graph Hashing: To alleviate the drawback of SH, Liu et al. [18] and Shen et al. [19] show that it is helpful to keep the binary constraint $Y \in \{-1, 1\}^{k \times n}$, establishing a method called discrete graph hashing (DGH). However, DGH may not fully remedy the defect of mapping dissimilar data points to similar hash codes, which is essentially caused by the imperfect objective function in (1). Moreover, the discrete optimization problem in DGH is NP-hard in nature and hard to solve.

B. This Work

In order to fix the defect of mapping dissimilar points to adjacent hash codes, we argue that one just needs to change the position of the input $x_i$s with the output $y_i$s in (1), resulting in a novel formulation termed reversed SH (ReSH). Different from SH, which estimates the affinity matrix $A$ from the given high-dimensional data points, $x_i$s, our ReSH defines the similarities between data points according to the unknown low-dimensional hash codes, $y_i$s. Those code-dependent similarities take charge of evaluating the quality of the produced hash codes and, therefore, dominate the whole training process. Interestingly, unlike SH, which tends to assign similar hash codes to dissimilar data points, the minimization problem in our ReSH can be optimized if and only if similar data points are mapped to adjacent hash codes, and even more, dissimilar data points are considerably separated from each other in the hash code space.

In our ReSH, the hash codes $y_i$s and the similarities $a_{ij}$s are mutually dependent and both entirely unknown. As a consequence, not surprisingly, the optimization problem arising from ReSH is complex and not easy to solve. Fortunately, we find that the optimization problem could be solved approximately by making use of artificial neural networks (ANNs). Namely, we first simulate the underlying hash mapping by a multilayer feedforward neural network, and then use the resilient backpropagation algorithm [20] to train the network that approximately solves the optimization problem in ReSH. Experiments on three benchmark data sets, CIFAR-10 [21], 22K-LabelMe [22], and ALOI [23], show the superior performance of the proposed ReSH method. In summary, the contributions of this paper include the following.

1) We propose a novel unsupervised hashing method termed ReSH, which can seamlessly overcome the drawback of the classical SH method. Compared with DGH, our ReSH is more effective and more convenient to work with.

2) The formulation of ReSH results in a difficult minimization problem. We devise a neural network-based algorithm to approximately solve the optimization problem and obtain superior performance in our experiments.

3) The idea of ReSH is probably useful for improving many other graph-based learning methods that share a similar drawback as SH. In that sense, the significance of this paper is indeed not limited to hashing.

The rest of this paper is organized as follows. Section II presents the details of ReSH and the optimization algorithm. Section III implements different methods to experimentally verify the effectiveness of our proposed ReSH. Section IV concludes this paper.

II. SPECTRAL HASHING WITH INPUT AND OUTPUT INTERCHANGED

In general, the proposed ReSH method is the reversed form of SH. Different from SH, ReSH defines the similarities between data points according to the unknown low-dimensional hash codes rather than the given high-dimensional data. Those code-dependent similarities take charge of evaluating the quality of the produced hash codes. Due to this key difference, ReSH can seamlessly eliminate the drawback of assigning close hash codes to distant data points.

A. Problem

Suppose we are given with a real-valued data matrix $X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{d \times n}$, each column in which is a $d$-dimensional data point $x_i \in \mathbb{R}^d$. Without loss of generality, we assume that the data are normalized, such that the elements in $x_i$s are within the range from $-1$ to $1$. Our goal is to learn a hash mapping, denoted as $f$, that converts any data point $x$ to a binary string $y \in \{0, 1\}^k$ of length $k$

$$y = f(x) : \mathbb{R}^d \rightarrow \{0, 1\}^k.$$ (3)

The similarity structure of the original data is required to be preserved. Namely, similar points in the original data space $\mathbb{R}^d$ should be hashed into adjacent binary codes, and mostly important, dissimilar points must be considerably separated from each other in the hash code space. Also, the learned hash mapping $f$ is required to have generalization ability to well handle any fresh data.

B. Model

To remedy the defect of mapping dissimilar points to similar hash codes, we argue that one just needs to change the position of the input $x_i$s with the output $y_i$s in SH (1). More precisely, in this paper, we suggest to consider the following model:

$$\min_{f \in \mathcal{F}} \frac{1}{n^2} \sum_{i,j=1}^{n} a_{ij} \| x_i - x_j \|^p \quad \text{s.t.}\ y_i = f(x_i), \quad y_i \in \{0, 1\}^k, \quad i = 1, \ldots, n$$

$$a_{ij} = \begin{cases} 1, & \text{if } D(y_i, y_j) < \sigma \\ 0, & \text{otherwise} \end{cases}$$ (4)

where $\mathcal{F}$ generally denotes the continuous mapping family each member of which converts high-dimensional data to
low-dimensional codes, \(^1\) \(D(\cdot, \cdot)\) denotes the Hamming distance between hash codes, \(p > 0\) is a parameter that is responsible for rescaling the distances between the data points, and \(\sigma > 0\) is a thresholding parameter. Notice that the continuity of the mapping \(f\) can enforce similar data points to have adjacent hash codes. Also, mostly important, the objective function in (4) can be minimized only if distant data points are considerably separated from each other in the low-dimensional code space. So, the problem in (4) is optimized if and only if the produced hash codes can strictly preserve the similarity structure of the given high-dimensional data. That is, dissimilar data points are considerably separated from each other in the code space, while similar data points are mapped to adjacent hash codes.

### C. Relaxation

The optimization problem in (4) is not easy to solve, due to the discrete nature of the hash codes and, even more, the mutual dependence between the hash codes \(y_i's\) and the similarities \(a_{ij}''s\). In particular, the dependence between \(y_i's\) and \(a_{ij}''s\) is determined by a hard thresholding function, the output of which is discrete too.

For the ease of optimization, first, we relax the binary constraint, \(y_i \in \{0, 1\}^k\), by using ANN with “S”-shaped activation functions (e.g., the sigmoid) to simulate the hash mapping \(f\), i.e., we assume that

\[
\mathcal{F} = \{\text{ANN with the sigmoid activation function}\}. \quad (5)
\]

In the experiments of this paper, the used sigmoid function is simply \(s(t) = 1/(1 + \exp^{-t})\). It is worth noting that the binary constraint \(y_i \in \{0, 1\}^k\) is indeed well treated in our method, because the use of sigmoid (or any “S”-shaped) activation function can make the output of \(f\) be near binary, as will be shown later.

Second, to smooth the mutual dependence between the codes \(y_i's\) and the similarities \(a_{ij}''s\), we replace the hard thresholding operator, \(D(y_i, y_j) < \sigma\), with a soft one. To be more precise, the affinity matrix \(A\) in (4) is defined as

\[
a_{ij} = \exp\left(-\frac{\|y_i - y_j\|^2}{2\sigma^2}\right) \quad (6)
\]

where \(y_i\) is the real-valued, near binary codes of the \(i\)th data point \(x_i\), \(\| \cdot \|\) denotes the \(\ell_2\) norm of a vector, and \(\sigma\) is the thresholding parameter. In this way, the discrete nature of the problem in (4) has been removed, and thereby we reach the following optimization problem:

\[
\begin{aligned}
&\min_{f \in \mathcal{F}} \frac{1}{n^2} \sum_{i,j=1}^{n} a_{ij} \|x_i - x_j\|^p \\
&\text{s.t. } y_i = f(x_i), \quad i = 1, \ldots, n \\
&\quad a_{ij} = \exp\left(-\frac{\|y_i - y_j\|^2}{2\sigma^2}\right). \quad (7)
\end{aligned}
\]

1Since a hash mapping with discrete outputs cannot be meanwhile continuous, it is assumed in (4) that the output space of \(f\) is \(\mathbb{R}^k\) rather than \(\{0, 1\}^k\). This is not inconsistent with the binary constraint of \(f(x_i) \in \{0, 1\}^k\), because \(\{x_i\}_{i=1}^{n}\) is a discrete set.

In general, this problem can be solved by using backpropagation to train an ANN, as will be shown in Section II-C1.

**Connections to Existing Methods:** The objective function in (7) is somewhat similar to the second term of elastic embedding (EE) [24], whose formulation involves two terms. The first term puts the similar points near each other and the second term puts the dissimilar points far apart from each other. The idea of EE has been used in several hashing papers [25]. Different from EE-like methods, our proposed method utilizes the continuity of the hash mapping to achieve local smoothness and, therefore, involves only one term in the formulation. Moreover, please note that (7) is just a relaxation of our original formulation (4), which is essentially different from the second term of EE.

### D. Optimization

To solve the problem in (7), generally, there are many techniques, such as expectation maximization and deep learning, for neural networks to be used. For the sake of computational efficiency, we use a standard backpropagation algorithm proposed in [20] to train a flat feedforward neural network that solves (7) in an approximate fashion.

We first denote the objective function in (7) as \(E\)

\[
E = \frac{1}{n^2} \sum_{i,j=1}^{n} a_{ij} \|x_i - x_j\|^p \quad (8)
\]

\[
= \frac{1}{n^2} \sum_{i,j=1}^{n} \exp\left(-\frac{\|y_i - y_j\|^2}{2\sigma^2}\right) \|x_i - x_j\|^p.
\]

With this notation, it can be calculated that the gradient of the objective function value, \(E\), with respect to the output variable, \(y_i\), is given by

\[
\frac{\partial E}{\partial y_i} = \sum_{j=1}^{n} \frac{-\|x_i - x_j\|^p}{n^2} \exp\left(-\frac{\|y_i - y_j\|^2}{2\sigma^2}\right) \frac{y_i - y_j}{\sigma^2}. \quad (9)
\]

Given \(\partial E/\partial y_i\), the gradients with respect to the network weights, denoted as \(\partial E/\partial w_{ij}\), can be computed by backpropagation. So, the problem in (7) can be solved by the standard backpropagation algorithm. To obtain more reliable solutions, we use instead the resilient backpropagation algorithm [20], the pseudocode of which is shown in Algorithm 1.

The parameters of the resilient backpropagation algorithm do not need extensive adjustments and could be set as follows: \(\Delta_0 = 0.07\) (initial weight change), \(\Delta_{\text{max}} = 50\) (maximum weight change), \(\Delta_{\text{min}} = 10^{-6}\) (minimum weight change), \(\eta^+ = 1.2\) (increment to weight change), and \(\eta^- = 0.5\) (decrement to weight change). Empirically, the resilient backpropagation algorithm can converge within 70 epochs, as exemplified in Fig. 1 (left).

1) **Quantization:** The hash mapping \(f\), learned by solving the problem in (7), does not convert a data point into a hash code of strictly binary. Thus, we need a quantization scheme to postprocess the output of \(f\). As can be seen from Fig. 1 (right), the output of \(f\) is very close to be binary. Hence, we apply a simple operator \(h(\cdot)\) to quantize...
Fig. 1. Left: exemplifying the convergence properties of the optimization algorithm presented in Section II-D. Right: distribution of the output value of the learned hash mapping $f$. These statistics are collected from 1.4 million output values obtained in our experiments. The label “0.05” refers to the interval of $(0, 0.1]$, and similarly for “0.15,” “0.25,” . . . , “0.95.”

Fig. 2. Precision–recall and recall curves on CIFAR-10. (a) Precision–recall curve at 16 b. (b) Precision–recall curve at 24 b. (c) Precision–recall curve at 32 b. (d) Recall curve at 16 b. (e) Recall curve at 24 b. (f) Recall curve at 32 b.

the output of $f$, where $h(v)$ takes value 1 if $v \geq 0.5$ and 0 otherwise.

2) Reducing the Training Time: Since the gradient for a single output point is a sum over all points [see (9)], the computational cost for learning $f$ from $n$ data points is $O(n^2)$. This is expensive on big data sets where $n$ is very large. To speed up the training process, we utilize the idea of stochastic gradient descent. In each epoch, we randomly choose $r$ integers from 1 to $n$, denoted as $T = \{i_1, i_2, \ldots, i_r\}$, and then approximate the objective function $E$ in (8) by

$$E_r = \frac{1}{nr} \sum_{i \in T} \sum_{j=1}^{n} \exp \left(-\frac{\|y_i - y_j\|^2}{2\sigma^2}\right) \|x_i - x_j\|^p$$

(10)
where \( r \) is token as a parameter and referred to as “sampling rate.” In this way, the computational cost is reduced to \( O(nr) \) with \( r \ll n \). According to our experimental investigations, ReSH can work equally well when \( r \geq 0.05n \).

III. Experiments

A. Experimental Data

We evaluate our method on four benchmark data sets, CIFAR-10 [21], 22K-LabelMe [22], ALOI [23], and ANN-GIST1M [26], which are widely used as test beds in large-scale search.

1) CIFAR-10: The first data set, CIFAR-10, contains 60K RGB images of size 32 \( \times \) 32 and has been manually labeled into ten classes. Each class consists of 6K images. For each image, we extract a 128-D GIST [27] feature vector, resulting in a collection of 60K data points of 128-D for experiments. We randomly choose 59K images as training data for learning the underlying hash mapping \( f \) and the rest for testing.

2) 22K-LabelMe: The second data set we experiment with, 22K-LabelMe, consists of 22,019 images randomly sampled from the whole LabelMe database. We exact a 128-D GIST feature vector for each image, resulting in a collection of 22,019 data points of 128-D for experiments. Again, the data set is randomly divided into two parts: 20K data points for training and the rest 2019 points for testing.

3) ALOI: The third data set, ALOI, contains 108K 128-D data points. We randomly choose 100K points as training data and the rest for testing.

4) ANN-GIST1M: The fourth data set for experiment is the ANN-GIST1M [26], which contains a training set of one million 916-D GIST feature points and a testing set of 1K feature points. For convenience, the dimension of the feature points is reduced to 128 by principal component analysis.

B. Baselines

Since the proposed ReSH is closely related to SH, we adopt SH as a benchmark baseline for comparison. To verify the position of ReSH in the state-of-the-art (unsupervised) hashing methods, besides of SH, we also include for comparison eight prevalent methods, including DGH [18], semantic hashing [28], ITQ [17], isotropic hashing [29], sparse projection [30], locality sensitive hashing [2], unsupervised sequential projection learning for hashing [31], and spherical
hashing [32]. In addition, we also include, for comparison, the unsupervised kernel hashing, which is an unsupervised version of [9].

C. Evaluation Metrics

For each query point (i.e., test point), its ground-truth neighbors are defined as the points that lie within the top 0.01%∼1% points close to the query in terms of the original Euclidean distance. Then precision–recall, recall, and mean average precision (MAP) are used as the metrics for evaluating the performance of hashing methods.

D. Parameter Configuration

In the proposed ReSH method, the parameter \( p \) takes the role of rescaling the distances between the data points so as to refine the similarity structure of the given data. In our experiments, \( p \) is chosen as 8. Obviously, the thresholding parameter \( \sigma \) should increase as the number of hash bits, \( k \). Since the L2 norm of a nearly binary \( k \)-bit code is \( O((k)^{1/2}) \), the dependence of \( \sigma \) on \( k \) could be described as

\[
\sigma = \alpha \sqrt{k}
\]  

where \( \alpha \) is set as 0.2 in our experiments. Since the desired hashing mapping \( f \) is required to be continuous, the neural network for simulating \( f \) only needs one hidden layer. In all the experiments shown in this paper, the neural network structure is configured as 128-300-\( k \), where \( k \) is the number of hash bits we want to generate.

E. Results

Using a workstation of Intel Core i7-4790 CPU@3.60 GHz with 8G memory, the training time of our algorithm on two small data sets (CIFAR and LabelMe) is about 10 min \( (r = 0.05n) \). For training the network using 1 million data points, the runtime is a little longer than 1 h \( (r = 0.05n) \).

The comparison results are shown in Figs. 2–5 and Table I. It can be seen that our ReSH is distinctly better than the benchmark baseline, SH. To be more precise, in terms of MAP, Table I shows that ReSH is at most 132% and at least 33% better than SH. This verifies that it is very beneficial to change the position of the input with the output in SH, which arguably fixes the defect of assigning similar hash codes to dissimilar data points. Compared with the state-of-the-art methods, the performance of ReSH is also superior. For example, on ALOI,
The 32-D hash codes produced by ReSH get an MAP of 0.56, which is 7.5% higher than the most close baseline, ITQ. These results confirm the effectiveness of the proposed ReSH method.

**Influences of the Parameters:** In ReSH, the thresholding parameter $\sigma$ cannot be too large or too small, and similarly for the rescaling parameter $p$. Regarding the sampling rate, $r$, there is a tradeoff between efficiency and effectiveness: Using smaller $r$, the training process is faster but the retrieval results might be less accurate. The results in Fig. 6 (middle) show that it is adequate to set the thresholding parameter as $\sigma = \alpha(k)^{1/2}$, with $\alpha$ being around 0.2. The rescaling parameter $p$ could be chosen around 8, as can be seen from Fig. 6 (left). Empirically, as shown in Fig. 6 (right), our ReSH performs almost equally well when $r \geq 0.05n$.

**IV. CONCLUSION**

In this paper, we studied hashing that is a key technique for solving the efficiency issues in dealing with high-dimensional,
massive data. Particularly, we were interested in SH, which is a classical method of unsupervised hashing. While computationally simple, SH has a defect of mapping dissimilar data points to adjacent hash codes. To overcome this drawback, we proposed a novel method termed ReSH. We showed that the minimization problem in ReSH can be optimized if and only if the similarity structure of the high-dimensional data is strictly preserved in the low-dimensional hash code space. Experimental results have verified that ReSH is much better than SH and can achieve the state-of-the-art performance.

However, it is entirely possible to improve ReSH by using some more effective learning techniques (e.g., the deep learning methods) to solve the optimization problem in (7). We would leave this as future work.

REFERENCES


Qingshan Liu (M’05–SM’08) received the Ph.D. degree from the National Laboratory of Pattern Recognition, Chinese Academy of Sciences, Beijing, China, in 2003, and the M.S. degree from Southeast University, Nanjing, China, in 2000. He was an Assistant Research Professor with the Computational Biomedicine Imaging and Modeling Center, Department of Computer Science, Rutgers University, New Brunswick, NJ, USA. From 2010 to 2011, he was an Associate Professor with the National Laboratory of Pattern Recognition, Chinese Academy of Sciences. He is currently a Professor with the School of Information and Control Engineering, Nanjing University of Information Science and Technology, Nanjing. His current research interests include image and vision analysis, including face image analysis, graph- and hypergraph-based image and video understanding, medical image analysis, and event-based video analysis.

Dr. Liu was a recipient of the President Scholarship of the Chinese Academy of Sciences in 2003.

Guangcan Liu (M’11) received the bachelor’s degree in mathematics and the Ph.D. degree in computer science and engineering from Shanghai Jiao Tong University, Shanghai, China, in 2004 and 2010, respectively.

He was a Post-Doctoral Researcher with the National University of Singapore, Singapore, from 2011 to 2012, the University of Illinois at Urbana–Champaign, Champaign, IL, USA, from 2012 to 2013, Cornell University, Ithaca, NY, USA, from 2013 to 2014, and Rutgers University, New Brunswick, NJ, USA, in 2014. Since 2014, he has been a Professor with the School of Information and Control, Nanjing University of Information Science and Technology, Nanjing, China. His current research interests include the areas of machine learning, computer vision, and image processing.

Lai Li received the bachelor’s degree in electronic information and electrical engineering from the Nanjing University of Information Science and Technology, Nanjing, China, in 2014, where he is currently pursuing the master’s degree with the School of Information and Control.

His current research interests include machine learning and computer vision.

Xiao-Tong Yuan (M’14) received the B.A. degree in computer science from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2002, the M.E. degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2005, and the Ph.D. degree in pattern recognition from the Chinese Academy of Sciences, Beijing, China, in 2009.

He was a Post-Doctoral Research Associate with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, the Department of Statistics and Biostatistics, Rutgers University, New Brunswick, NJ, USA, and the Department of Statistical Science, Cornell University, Ithaca, NY, USA. In 2013, he joined the Nanjing University of Information Science and Technology, Nanjing, where he is currently a Professor of Computer Science. His current research interests include machine learning, data mining, and computer vision.

Meng Wang received the B.E. and Ph.D. degrees from the Special Class for the Gifted Young and the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, China, in 2003 and 2008, respectively.

He is currently a Professor with the Hefei University of Technology, Hefei. His current research interests include multimedia content analysis, computer vision, and pattern recognition. He has authored over 200 book chapters, journal, and conference papers in these areas.

Wei Liu (M’14) received the M.Phil. and Ph.D. degrees in EECS from Columbia University, New York, NY, USA, in 2012.

He has been a Research Scientist with the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, since 2012. He is currently a Computer Vision Director with the Tencent AI Lab, Shenzhen, China. He has been involved in the research and development of computer vision, machine learning, data mining, and information retrieval. He has authored over 100 peer-reviewed journal and conference papers.