Learning Discriminative Feature Representations for Visual Categorization

Li Liu

Department of Electronic and Electrical Engineering

University of Sheffield

This thesis is submitted for the degree of

Doctor of Philosophy

February 2015

Abstract

Learning discriminative feature representations has attracted a great deal of attention due to its potential value and wide usage in a variety of areas, such as image/video recognition and retrieval, human activities analysis, intelligent surveillance and human-computer interaction.

In this thesis we first introduce a new boosted key-frame selection scheme for action recognition. Specifically, we propose to select a subset of key poses for the representation of each action via AdaBoost and a new classifier, namely WLNBNN, is then developed for final classification. The experimental results of the proposed method are 0.6% $\sim$ 13.2% better than previous work. After that, a domain-adaptive learning approach based on multi-objective genetic programming (MOGP) has been developed for image classification. In this method, a set of primitive 2-D operators are randomly combined to construct feature descriptors through the MOGP evolving and then evaluated by two objective fitness criteria, i.e., the classification error and the tree complexity. Later, the (near-)optimal feature descriptor can be obtained. The proposed approach can achieve 0.9% $\sim$ 25.9% better performance compared with state-of-the-art methods. Moreover, effective dimensionality reduction algorithms have also been widely used for obtaining better representations. In this thesis, we have proposed a novel linear unsupervised algorithm, termed Discriminative Partition Sparsity Analysis (DPSA), explicitly considering different probabilistic distributions that exist over the data points, simultaneously preserving the natural locality relationship among the data. All these above methods have been systematically evaluated on several public datasets, showing their accurate and robust performance (0.44% $\sim$ 6.69% better than the previous) for action and image categorization. Targeting efficient image classification , we also introduce a novel unsupervised framework termed evolutionary compact embedding (ECE) which can automatically learn the task-specific binary hash codes. It is regarded as an optimization algorithm which combines the genetic programming (GP) and a boosting trick. The experimental results manifest ECE significantly outperform others by 1.58% $\sim$ 2.19% for classification tasks. In addition, a supervised framework, bilinear local feature hashing (BLFH), has also been proposed to learn highly discriminative binary codes on the local descriptors for large-scale image similarity search. We address it as a nonconvex optimization problem to seek orthogonal projection matrices for hashing, which can successfully preserve the pairwise similarity between different local features and simultaneously take image-to-class (I2C) distances into consideration. BLFH produces outstanding results ($0.017\% \sim 0.149\%$ better) compared to the state-of-the-art hashing techniques.

I would like to dedicate this thesis to my parents and my fiancee

Acknowledgements

My deepest gratitude goes first and foremost to my supervisor Prof. Ling Shao for his kind and delicate supervision in both research and life. He gives me constant encouragement and leads me into the field of computer vision and machine learning. He has walked me through all the stages of my Ph.D research and shared a professional attitude with me. Without his consistent and illuminating instruction, I could not reach the present progress. Besides, my thanks also go to Dr. Peter Rockett who as my secondary supervisor always conveys novel ideas and his strict working style inspires me a lot.

I would like to thank my colleagues: Dr. Xiantong Zhen, Dr. Simon Jones, Dr. Ruomei Yan, Dr. Di Wu, Fan Zhu, Bo Dong, Peng Peng, Redzuan Bin Abdul Manap, Mengyang Yu, Yang Long, Ziyun Cai, Xu Dai, Danyang Wang and Feng Zheng. I feel grateful to Mengyang and Feng who show their strong background of mathematics and excellent collaboration on our research. Xiaotong, Di and Fan have also expressed their kind assistance along my whole Ph.D study. Thanks to Yang and Ziyun for bringing me joyful experience in my leisure time. My special thanks express to Postgraduate Administrator, Ms. Hilary J Levesley, as well.

I would also like to thank other researchers in our laboratory: Dr. Yilong Cao, Dr. Ji Ni, Dr. Wenting Duan, Dr. Jing Li, Zhizhong Yu, Zia Khan, Yanxiang Wang and Tian Feng for their kind help on my research.

I would like to thank Dr. Dabo Guo, Dr. Hui Zhang, Dr. Peng Li, Dr. Jun Tang, Dr. Shoubiao Tan and Dr. Xuezhi Wen. I feel happy to have refreshing conversations with them on both study and life, and also learn a lot from them.

I also express my thanks to my friends. Gaochao Zhang, a kind boy and good chef, delivers his consistent encouragement to me when I am in my hardest time. Thanks also go to Dr. Haoyu Zhang, Miss Jiaxin Xu, Miss Qian Guo, and Miss Mei Li.

Particularly, I would like to thank Prof. Xuelong Li who is always regarded as a brother and friend to mentor my research and career. Without him, I may not have had the change to do my PhD in Sheffield.

My grateful thanks also go to Dr. Matt Mellor and his team. I am very happy to have a unforgettable internship experience with them in Createc.Ltd. In there, I learned more piratical skills on realistic vision applications. Thanks for this rare opportunity. I highly appreciate the amazing techniques there and the worthy discussions with them.

I would like to thank Dr. Tao Xiang and Dr. Charith Abhayaratne as my examiners for carefully reviewing my thesis.

At last, I would like to thank my loving parents for their unconditional supports in the period of my Ph.D. I am also grateful to my wife Xiaoyu for her constant love and for taking care of my life in UK.

My Publications

I hereby declare that parts of the following papers have been included in this thesis.

Journal Articles:

- **L. Liu**, L. Shao, X. Zhen and X. Li, "Learning Discriminative Key Poses for Action Recognition", IEEE Transactions on Cybernetics, vol. 43, no. 6, pp. 1860-1870, Dec. 2013. **[Chapter 2]**
- **L. Liu**, L. Shao and P. Rockett, "Boosted Key-Frame Selection and Correlated Pyramidal Motion Feature Representation for Human Action Recognition", Pattern Recognition, vol. 46, no. 7, pp. 1810-1818, Jul. 2013.
- **L. Liu**, L. Shao and P. Rockett, "Human Action Recognition Based on Boosted Feature Selection and Naive Bayes Nearest Neighbor Classification", Signal Processing, vol. 93, no. 6, pp. 1521-1530, Jun. 2013.
- **L. Liu**, L. Shao and X. Li, "Evolutionary Compact Embedding for Large-scale Image Classification", Information Sciences (2014), doi: org/10.1016/j.ins.2014.06.030. **[Chapter 6]**
- **L. Liu**, L. Shao and F. Zheng and X. Li, "Realistic Action Recognition via Sparsely-Constructed Gaussian Processes", Pattern Recognition, vol. 47, no. 12, pp. 3819-3827, Dec. 2014.
- **L. Liu**, L. Shao and M. Yu, "Multiview Alignment Hashing for Image Search", IEEE Transactions on Image Processing (2015), doi: 10.1109/TIP.2015.2390975.
- L. Shao, **L. Liu** and X. Li, "Feature Learning for Image Classification via Multiobjective Genetic Programming", IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 7, pp. 1359-1371, Jul. 2014. **[Chapter 4]**
- R. Yan, L. Shao, **L. Liu** and Y. Liu, "Natural Image Denoising Using Evolved Local Adaptive Filters", Signal Processing, vol. 103, pp. 36-44, Oct. 2014.
- **L. Liu**, L. Shao, X. Li and K. Lu, "Learning Spatio-Temporal Representations for Action Recognition: A Genetic Programming Approach", IEEE Transactions on Cybernetics (2015), doi: 10.1109/TCYB.2015.2399172.

- **L. Liu**, L. Shao and M. Yu, "Bilinear Local Feature Hashing via Bigraph Regularization", submitted to International Journal of Computer Vision. **[Chapter 7]**

Conference Papers:

- **L. Liu**, L. Shao and X. Li, "Building Holistic Descriptors for Scene Recognition: A Multiobjective Genetic Programming Approach", ACM International Conference on Multimedia (MM), Barcelona, Spain, 2013. [Full Paper: Oral]
- **L. Liu** and L. Shao, "Learning Discriminative Representations from RGBD Video Data", International Joint Conference on Artificial Intelligence (IJCAI), Beijing, China, 2013.
- **L. Liu** and L. Shao, "Synthesis of Spatio-Temporal Descriptors for Dynamic Hand Gesture Recognition Using Genetic Programming", IEEE International Conference on Automatic Face and Gesture Recognition (FG), Shanghai, China, 2013. **[Chapter 3]**
- **L. Liu**, L. Shao and P. Rockett, "Genetic Programming Evolved Spatio-Temporal Descriptor for Human Action Recognition", British Machine Vision Conference (BMVC), Surrey, UK, 2012.
- **L. Liu** and L. Shao, "Discriminative Partition Sparsity Analysis", International Conference on Pattern Recognition (ICPR), Stockholm, Sweden, 2014. **[Chapter 5]**

Li Liu
February 2015

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Representation learning has become a field in itself in the machine learning community. A discriminative feature representations of the data can make it easier to extract useful information when building classifiers or other predictors. This thesis is about representation learning, i.e., learning representations of the data that make it easier to extract useful information when building classifiers or other predictors. In the case of probabilistic models, a good representation is often one that captures the posterior distribution of the underlying explanatory factors for the observed input. A good representation is also one that is useful as input to a supervised predictor. In this thesis, three different branches of discriminative representation learning schemes, i.e., feature selection, feature learning and feature embedding, are carefully discussed. The rapid increase in scientific activity on representation learning has been accompanied and nourished by a remarkable string of empirical successes both in academia and in industry. For instance, learning discriminative feature representations has successfully involved in speech recognition and signal processing. The recent revival of interest in representation learning has had a strong impact in the area of speech recognition, with breakthrough results [1–3] obtained by several academics as well as researchers at industrial labs bringing these algorithms to a larger scale and into products. Furthermore, MNIST digit image classification problem as a object recognition task has attracted lots of researchers using various feature learning schemes [4] on it. Besides, natural language processing (NLP) needs the phase of representation learning, as well. Among the various applications using learned representations, in this thesis, high-level visual categorization and retrieval tasks are focused and discussed in detail.

Visual categorization is regarded as a wide and significant tasks attracts increasing number of attentions. It Includes 2D applications such as, object classification, face recognition and scene classification, and also cover the relevant 3D problems such as action recognition and gesture recognition. The basic visual categorization algorithm is generally introduced in [5, 6], and involves two main stages: (1) low/middle level feature extraction and representation; (2) high level visual classification. The low level part is commonly accomplished by local or global feature descriptors, *e.g.*, histogram of oriented gradients (HOG) [7], scale invariant feature transform (SIFT) [8] and local binary pattern (LBP) [9], to represent the extracted most salient features such as: edges, corners, orientations, which significantly affect high level classification performance. To learn discriminative feature representations, some learning schemes can be used. One of the effective and intuitive way is to select the most discriminative features from the original feature pool. This scheme can successfully discard the noisy data and make the classifier easier to conduct the final decisions. In addition to select features, another big branch of research focuses on learning the totally new but discriminative feature representations. This kind of scheme is usually based on advanced machine learning techniques such as: deep learning and genetic programming and can automatically learn the feature representations from the raw visual sources. Of course, besides the low/middle-level feature extraction, some embedding schemes are also proposed to learn the discriminative representation such as principal component analysis (PCA) and linear discriminative analysis (LDA). Although, these previously published techniques lead to reasonable results in various applications, they are only suitable for a particular research domain or data type and would result in poor performance on other unknown usages. So how to propose an optimal solution that can be considered as a generalized way to extract the most meaningful features for any user-defined application is presented to us.

Moreover, with spreading of World Wide Web and development of computer technologies, a huge amount of digital data, including text, images and videos, is generated, stored, analyzed, and accessed every day. To overcome the shortcomings of text-based image retrieval, content-based image classification and retrieval has attracted substantial attention. The most basic but essential scheme for image retrieval is the nearest neighbor search: given a query image, to find an image that is most similar to it within a large database and assign the same label of the nearest neighbor to this query image. However, greedily searching a dataset with N samples is infeasible because linear complexity $O(N)$ is not scalable in practical applications. Due to this kind of computational complexity problem, researcher have already developed some approach to fast index data, e.g., K-D tree and R tree [10]. Nevertheless, most of these methods can only handle the data within dimensionality of 100 [11, 12]. In addition, curse of dimensionality problems[1] always exists in the most of the vision-based applications due to visual representations usually with hundreds or even thousands of dimensions. Thus, to make large-scale search or classification practical,

[1]The effectiveness and efficiency of these methods drop exponentially as the dimensionality increases, which is commonly referred to as the "curse of dimensionality".

some hash-based methods have been proposed to effectively reduce the dimension of data and increase the retrieval speed and accuracy. In the following sections, some of our main work with its motivations for categorization and retrieval tasks will be briefly introduced.

1.1 Boosted Key-pose Selection for Action Recognition

Huamn action recognition, nowadays, plays a significant role in various applications, *e.g.*, human-computer interaction [13], human activities analysis [14], [15], [16], [17], and real-time surveillance systems [18], [19]. The goal of human action recognition is to identify the actions being performed in a video sequence under different complications such as cluttering, occlusion and change of lighting conditions.

Generally, the approaches to action recognition involve two main stages: (1) feature extraction and representation; (2) action classification.

In recent years, some popular spatio-temporal features extracted from video sequences are commonly extended from their counterparts in the 2D image domain, and are demonstrated to achieve relatively good results for action recognition. Those methods include histogram of optical flow (HOF) [20], 3D scale invariant feature transform (3D-SIFT) [21], 3D histogram of oriented gradients (3D-HOG) [22], 3D speeded up robust features (3D-SURF) [23], etc. These spatio-temporal features can be used either globally or locally for human action representation.

Local methods [24], [14] [25] represent human actions as a set of spatio-temporal interest points detected from video sequences. Local methods based on the bag-of-features model have achieved impressive results in various action recognition tasks. They follow a typical procedure: unsupervised techniques are used to detect interest points around which spatio-temporal features are extracted; then clustering methods, *e.g.*, K-means clustering [26], are employed to construct a codebook on which all the features from a video sequence are mapped to form a histogram representation; the final representation is then fed to a classifier, *e.g.*, SVM, for action classification. The bag-of-features [16] model tends to be more robust in challenging scenarios, but this kind of sparse representation is often not precise and informative because of the quantization error during codebook construction and the loss of structural relationships among local features.

On the contrary, global methods [27] represent corresponding actions by treating the entire video sequence as a whole which contains the complete motion and appearance information. Such a representation has attracted much attention due to its abilities to extract more informative and accurate motion features from both spatial and temporal dimensions. However, global methods are quite sensitive to shift, scaling, occlusion and cluttering, which commonly exist in action sequences because the required background subtraction and segmentation [28] tend not to be very accurate.

Both local and global methods have achieved remarkable results, but action recognition by human suggests that they might be using more information than required [29]. Given the available actions, the human brain can easily recognize what a person is doing by just looking at a few poses without examining the whole sequence [30]. Inspired by this, recently, pose representations for human action recognition have drawn more attention and achieved promising results [30–33]. Pose-based representations can capture sufficient appearance information of actions and spatial layout of human bodies, and therefore it can to a large extent overcome the information loss induced in local representations. If key poses are selected well [30, 31, 33], pose-based representations are able to avoid the limitations such as background variations, occlusions and shifts in global representations.

Considering all the advantages and drawbacks mentioned above, thus, human actions is modeled based on the key pose representation. Each frame in an action sequence is treated as a pose. Inspired by the multi-resolution analysis in image processing, we describe poses using the extensive pyramidal features (EPF), which are composed of the Gabor pyramid, the Gaussian pyramid and the wavelet Laplacian pyramid. These features capture the orientation, intensity and contour information and thus provide an informative representation of poses.

Directly representing the video sequence by all the poses (frames), which contain redundant and indiscriminate information, would confuse the classifiers in action recognition [29]. We, therefore, propose to select a subset of key poses for the representation of each action by a supervised machine learning algorithm, *i.e.*, AdaBoost [34]. The selected key poses for each action type are not only more compact but also constitute the most discriminative body poses of an action, because the common poses that can exist in different action types have been eliminated.

In the proposed action representation, each frame contains the whole human body with the full spatial structural information, which shares the advantages with the global representation methods, while each video sequence is sparsely represented by a subset of key poses, which enjoys advantages of local representations. Therefore, the proposed method can be regarded as a semi-holistic representation of human actions and inherits the advantages of global features in the spatial dimensions and, meanwhile, has the superiority of local features in the temporal axis.

For local representations, the bag-of-features model and its variants plus SVM is the standard method for action recognition. Recently, a classifier called Naive Bayes Nearest Neighbor (NBNN) [35] is proposed for image classification tasks based on sets of local features. The NBNN classifier avoids the quantization error in the bag-of-features model by computing 'Image-to-Class' distance rather than the 'Image-to-Image' distance. McCann and Lowe [36] later developed the local Naive Bayes Nearest Neighbor (LNBNN) classifier with a remarkable increase of accuracy and decrease of running time compared with NBNN.

Inspired by LNBNN, we propose to further improve the NBNN model and introduce an enhanced version of NBNN, named weighted local Naive Bayes Nearest Neighbor (WLNBN-N), for the final action classification.

1.2 Genetic Programming-evolved Feature Learning

Much of previous work for feature representation has explored the use of low-level image features for high-level classification. Commonly, we adopt either dense sampling with local descriptors or pure holistic descriptors for feature representations. Popular choices for densely sampled feature descriptors include SIFT, HOG, P-HOG [37], LBP and shape context [38]. These features can work well for certain image types but may not be so descriptive for other types, since this kind of "hand-crafted“ features are not adaptive for more complex and challenging work. With the development of artificial intelligence, evolutionary methods simulate biological evolution to automatically generate solutions for user-defined tasks. GP, as one of the most outstanding evolutionary methods (see Algorithm 1), has been widely used in the computer vision domain.

Genetic Programming (GP), which is inspired by biological evolution and widely used in machine learning, artificial intelligence and other human-computer interaction applications, allows the computer automatically solving pre-defined tasks without requiring users to know or specify the form or structure of the solution in advance. In genetic programming, we initially generate a population of computer programs which are regarded as the potential candidates going through the following evolution strategies and optimized by the fitness evaluation according to programs' ability to perform the target task. After running all of evolution generations, computer can successfully select the best-so-far solution for user as the optimal result. The relevant Genetic programming algorithm flow is shown in Algorithm. 1.

Since the previous methods for human action recognition are highly depended on a fixed group of popular computer vision techniques and commonly achieve better results in some certain domain or application type, we aim to use Genetic Programming (GP) to find a relatively generalized motion feature descriptor for human action/gesture/objects recognition. Each of the individual in our GP population represents a candidate feature descriptor. The pre-processed action sequences as the input of GP are then calculated by computer programs

Algorithm 1 Genetic Programming

```
Start
  Initialization  Randomly create an initial population of computer programs from the
  available primitives (terminal set & function set).
    Repeat
      (1) Execute each program and evaluate its fitness;
      (2) Choose one program from the population with a particular probability based
      on the fitness to involve genetic operations;
      (3) Create a new generation programs by applying genetic operations;
      If  An acceptable solution is found or reach the maximum number of generations
      defined by user;
    Stop
  Return  The best-so-far solution selected by Genetic Programming.
End
```

and the output form is a vector matrix of feature representations. To start with our GP algorithm, here we need to pre-define three most significant concepts: terminal set, function set and fitness functions.

The terminal and function sets are regarded as the key components of genetic programming. The variables and constants of the genetic programming are always included in the terminal set. While, the functions usually can be as set of mathematical functions, i.e., addition, subtraction, multiplication and division, or other task-specific orders such as complex filtering operations.

The most important concept of genetic programming is the fitness function which determines how well a program is able to solve a problem. The basis of evolutionary methods is always to maximize the performance of individual solutions as gauged by some appropriate fitness function. Fitness function can be regarded as a 'Engine' driving the genetic programming to find the optimal solutions.

Furthermore, for pattern recognition, GP with a single fitness objective, which is always the accuracy of the program in recognising patterns, is usually used to measure solutions over a learning set. However, the single-objective GP is limited by the manner in which other objective criteria could be added to help guide the search, especially when conflicts exist. Moreover, during the GP process, the complexity of evolved solutions is one of the most difficult things to control in evolutionary systems, where the size and shape of the evolved solutions are under the control of evolution. In some applications, the trees in GP have a tendency to grow without limit - this phenomenon is called *bloat* [39], which always causes serious *overfitting* at the pattern classification phase. To avoid *bloat*, we adopt an improved GP version *i.e.*, multi-objective genetic programming (MOGP), which considers both the accuracy rate and the tree complexity as the fitness objectives based on Pareto dominance relations, to optimize the solutions and simultaneously reduce the *overfitting* effectively.

Inspired by MOGP, a new approach using MOGP for generating (near-)optimal feature descriptors has been developed in image classification applications. Given a group of primitive operators and a set of terminals (*i.e.*, labelled training examples and random constants), MOGP evolves (hopefully) better-performing individuals in the next generation. Since MOGP selection depends on two fitness objectives, we use the *Pareto front* to represent a set of non-dominated 'rank-1' solutions which provide different trade-offs among the two objectives based on standard evaluation criteria. Furthermore, we test these evolved solutions on the evaluation set to select the (near-)optimal one with the lowest test error as the final feature descriptor. Typically, the features extracted by the MOGP system contain domain-specific information which makes the later image classification easier. It should be noted that our proposed method does not guarantee any mathematical optimum, while, in practice, it usually finds a 'good' solution to an NP-hard problem in an acceptable amount of computing time. A preliminary version of this approach has been applied to evolve holistic scene features for scene recognition [40].

1.3 Learning Compact Embedding

Learning compact embedding has been a critical preprocessing step in many fields of information processing and analysis, such as data mining [41–45], information retrieval [46, 47], and pattern recognition [48–51]. In this section, we mainly introduce manifold learning based dimensionality reduction methods and hashing based binary code learning methods.

1.3.1 Manifold Learning for Dimensionality Reduction

Dimensionality reduction [52] has been a key problem attracting much attention in many fields of information processing, such as data mining [53], information retrieval, and pattern recognition [54, 55]. When data are represented as points in a high-dimensional space, one is often confronted with tasks like nearest neighbor search. Actually, greedily searching a dataset with N samples is infeasible because linear complexity $O(N)$ is not scalable in practical applications. To overcome this shortcoming of linear search, researchers have developed methods to index the data for fast query response, such as K-D tree, R tree, R* tree [10], which efficiently decrease the computational complexity from linear $O(N)$ to sublinear $O(N^{\frac{1}{2}})$. However, when the dimension of data points is more than 100 bits, the effectiveness and efficiency of those searching schemes will drop exponentially as the dimensionality increases, which is commonly referred to as the 'curse of dimensionality'. During the last decade, with the advances in computer technologies and the advent of the World Wide Web, there has been an explosion in the amount and complexity of digital data being generated, stored, analyzed, and accessed. Much of this information is multimedia in nature, including text, image, and video data. The multimedia data are typically of very high dimensionality, ranging from several thousands to several hundreds of thousands. Learning with such high dimensionality in many cases is almost infeasible. Therefore, to make large-scale search or classification practical, many methods have been proposed to effectively reduce the dimension of data and increase the classification speed and accuracy. Among them, linear embedding techniques show their promising efficiency and robustness for scalable data reduction tasks.

For manifold learning based dimensionality reduction algorithms, one of the most baseline methods might be principal component analysis (PCA). PCA is usually applied to compact the feature with a large amount of original variables into a smaller set of newly projected variables or dimensions, also guarantees a minimum loss of information. Another effective scheme for dimensionality reduction is linear discriminant analysis (LDA). LDA is a supervised method that has been proved successful on classification problems [56, 57]. Following the Fisher discriminant criterion, in which we maximize the between-class covariance meanwhile minimize the within-class covariance, the final projection vectors can be obtained. However, the classical LDA is a linear method and cannot tackle nonlinear problems. In order to overcome this limitation, kernel discriminant analysis (KDA) [58] is then developed. KDA is the nonlinear extension of LDA using the kernel trick that can be implicitly performed in a new feature space, which allows non-linear mappings to be learned. Beyond that, some other dimension reduction methods can also achieve promising results for different applications. In addition, another popular method, termed discriminative locality alignment (DLA) [59], is also been used as dimensionality reduction algorithms for classification.

we also review some structure preserving learning methods, such as ISOMAP [52], Laplacian Eigenmap (LE) [60] and Locally Linear Embedding (LLE) [61], were designed to preserve the manifold structure in the original space. A unified review and other manifold learning algorithms can be seen in [62]. Generally, linear methods possess high efficiency. Locality preserving projections (LPP) [42] are linear projective maps that are obtained by solving a variational problem that preserves the neighborhood structure of the data set. LPP aims to find the optimal linear approximations to the eigenfunctions of the Laplace Beltrami operator on the manifold. Neighborhood Preserving Embedding (NPE) [63] also tried to preserve the local representation of data. Capturing the intrinsic geometrical structure of data, Sparse Concept Coding (SCC) [64], a matrix factorization method, provides a sparse representation of the image space. For fast vision applications, pairwise structure preserving was adopted in [65] and each image was represented by a binary vector calculated via boosting coding.

In this thesis, we also introduce a novel linear unsupervised algorithm, termed Discriminative Partition Sparsity Analysis (DPSA), explicitly considering different probabilistic distributions that exist over the data points, meanwhile preserving the natural locality relationship among the data. Specifically, the Gaussian mixture model (GMM) is first applied to partition all samples into several clusters. In each cluster, a number of sparse sub-graphs are computed via the $\ell 1$-norm constraint to optimally represent the intrinsic data structure. Such sub-graphs are demonstrated to be robust to data noise, automatically sparse and adaptive to the neighborhood. All the sub-graphs from the clusters are then combined into a whole discriminative optimization framework for final reduction.

1.3.2 Hashing

Hashing on global representation

Recently, hashing has become a popular method applied to large-scale vision problems including object recognition, image retrieval, local descriptor compression, image matching, etc. In these problems, hashing is exploited to expedite similarity computation and search. Since the encoding of high-dimensional image feature vectors to short binary codes as proposed in, compact hashing has enabled significant efficiency gains in both storage and speed. In many cases, hashing with only several tens of bits per image allows search into a collection of millions of images in a constant time. The early exploration of hashing focuses on using random projections to construct randomized hash functions. One of the well-known representatives is Locality-Sensitive Hashing (LSH) which has been continuously expanded to accommodate more distance/similarity metrics including ℓ_p distance for $p \in (0,2]$, Mahalanobis distance, and kernel similarity. Furthermore, kernelized locality-sensitive hashing (KLSH) [44] has also been successfully proposed and utilized for large-scale image retrieval and classification. Because of theoretical guarantees that original metrics are asymptotically preserved in the Hamming (code) space with increasing code length, LSH-related methods usually require long codes to achieve good precision. Nonetheless, long codes result in low recall since the collision probability that two codes fall into the same hash bucket decreases exponentially as the code length increases

To design effective compact hashing, a number of methods such as projection learning for hashing have been introduced. In [66], the authors proposed to use stacked Restricted Boltzmann Machine (RBM) [67], and showed that it is indeed able to generate compact binary codes to accelerate document retrieval. Recently, another attempt called boosted similarity sensitive coding (BSSC) [65] has also been proposed to learn a weighted Hamming embedding for task specific similarity search. Furthermore, principled linear projections like PCA Hashing (PCAH) [68] has been developed for better quantization compared with random projection hashing. Additionally, another popular technique called Spectral Hashing (SpH) [69] was proposed, which preserves the data locality relationship to keep neighbors in the input space as neighbors in the Hamming space. After that, researchers utilize anchor graphs to obtain tractable low-rank adjacency matrices for efficient similarity search, termed Anchor Graphs Hashing (AGH) [70]. Beyond that, Self-Taught Hashing (STH) [71], Kernel-Based Supervised Hashing (KSH) [72], Compressed Hashing (CH) [73], *etc.*, have been effectively applied for large-scale data retrieval tasks, as well.

Although the existing learning methods achieve promising results in a variety of applications, they basically rely on complex and advanced mathematical knowledge to optimize the pre-defined object functions. However, for some optimization problems, direct solutions cannot always be found. Besides, in large-scale settings, matrix factorization techniques used in the above methods can also cause a heavy computational burden. So how to automatically generate better solutions to optimization problems becomes an interesting topic for real-world vision applications. In this thesis, we propose evolutionary compact embedding (ECE), which applies genetic programming (GP) in combination with AdaBoost to automatically evolve dimensionality reduction. ECE is demonstrated to enable accurate and robust large-scale image classification/retrieval.

In particular, we intentionally combine GP with a boosting trick to obtain a novel embedding method. For an M-bits embedding, GP is used to iteratively generate a best-performing weighted binary classifier for each bit by jointly minimizing its empirical risk with the Gentle AdaBoost strategy [74] on a training set. This embedding scheme reduces the Hamming distance between the data from the same class, while increasing the Hamming distance for data from different classes. The final optimized reduction representation is defined as the code calculated from the non-linear GP-evolved binary learner for each embedding bit.

Local feature hashing

However, being either unsupervised or supervised, all these existing hashing algorithms are primitively designed for global representations, e.g., GIST features [75] computed from whole images. For realistic image retrieval tasks, however, these global hashing techniques achieve suboptimal performance when images contain different complications such as cluttering, scaling, occlusion and viewpoint changes. To overcome these problems, local features, e.g., SIFT [8], are usually adopted to represent the images, which has been proved to be more robust in challenging scenarios. Inspired by advantages of local representations, in this thesis, we intend to develop a local feature based, supervised hashing method for improving the retrieval performance on realistic datasets. Being applied to detected keypoints, local hash codes are able to avoid the limitations such as background variations, occlusions, and shifts in global representations.

The goal of the work is to model a unified image retrieval technique based on supervised local feature hashing. To obtain more meaningful hashing code, two preserving schemes are adopted in our optimization, i.e., pairwise structure preserving and bigraph regularization. For pairwise structure preserving, a related work for fast vision applications has been carried out by Shakhnarovich *et al.* [65], in which each image is represented by a binary vector calculated via boosting coding. For the learning stage, positive examples are pairs of images x_i, x_j, so that x_j is one of the nearest neighbors of x_i, $x_j \in \mathrm{NN}(x_i)$. Negative examples are pairs of images that are not neighbors. A more related approach that works for local feature learning has also been introduced in [76]. In our framework, we preserve the pairwise structure information in hashing learning to embed high dimensional feature descriptors into a similarity-preserved Hamming space with a low dimension. However, only considering the pairwise relationship between each local features is not informative, since it lacks the high level measurement from source images. In fact, the images, which local features come from, and their corresponding class labels are noteworthy as well for pattern recognition tasks. Therefore, we are interested in the structure of the bipartite graph (a.k.a. bigraph) consisted of images and classes, i.e., the relationship between images and classes connected by local features. Image-to-class (I2C) distance provides a feasible way to measure it, which was first introduced in the naive Bayes nearest neighbor (NBNN) classifier [35]. It represents the sum of all distances from the local features of an image to their corresponding nearest neighbors in each class. This mechanism effectively avoids the quantization error in the bag-of-features model by computing 'image-to-class' distance rather than the 'image-to-image' distance. Furthermore, motivated by [77], to alleviate the computation of the eigen-decomposition of matrix in our method, a bilinear projection is employed to make the algorithm more efficient. To be specific, bilinear projection applies two projection matrices to local features, which have much smaller sizes than the original single projection matrix. Since the computational complexity of eigen-decomposition is cubic degree on the dimension of the matrix, the effect of applying smaller matrices is quite conspicuous.

We first propose a supervised Bilinear Local Feature Hashing (BLFH) framework for large-scale image similarity search, in which the pairwise structure preserving and the I2C distance preserving are naturally combined together. Since the raised problem is regarded as nonconvex and discrete, our objective function is then optimized via an alternate way with relaxation and converges to a near-optimal solution. Considering a more realistic application where not all samples' labels in dataset are available, a semi-supervised version of BLFH (referred as SBLFH) is additionally proposed. In the case of SBLFH, all the data can be used but only a few data are labeled. The relevant results show that our SBLFH can effectively learn the hash codes for image retrieval tasks as well. Moreover, our method is designed for local feature hashing, however, the original Hamming Ranking and Hamming Table cannot be directly applied to local feature based image indexing. Thus, we also introduce an image indexing/searching scheme named Local Hashing Voting (LHV), which has been demonstrated to be efficient and accurate for image similarity search in our experiments.

1.4 Summary of Remaining Chapters

The rest of the thesis is organized as follows. In Chapter 2, the work of learning discriminative key poses for action recognition is systematically reviewed. In Chapter 3, the detailed architecture of our genetic programming-based hand gesture recognition is presented. The work on feature learning for image Classification via multiobjective genetic programming is also illustrated in Chapter 4. In Chapter 5, the proposed work of discriminative partition sparsity analysis for dimensionality reduction is presented. The evolutionary compact embedding for large-scale image classification and retrieval is then presented in Chapter 6. In Chapter 7, a novel bilinear local feature hashing via bigraph regularization for improved image retrieval is demonstrated in detail. In the last chapter, a brief conclusion and future works is included.

Chapter 2

Learning Discriminative Key Poses for Action Recognition

2.1 Overview

In this chapter, we aim to model human actions based on the key pose representation. Each frame in an action sequence is treated as a pose. Inspired by the multi-resolution analysis in image processing, we describe poses using the extensive pyramidal features (EPF), which are composed of the Gabor pyramid, the Gaussian pyramid and the wavelet Laplacian pyramid. These features capture the orientation, intensity and contour information and thus provide an informative representation of poses.

Directly representing the video sequence by all the poses (frames), which contain redundant and indiscriminate information, would confuse the classifiers in action recognition [29]. We, therefore, propose to select a subset of key poses for the representation of each action by a supervised machine learning algorithm, *i.e.*, AdaBoost [34]. The selected key poses for each action type are not only more compact but also constitute the most discriminative body poses of an action, because the common poses that can exist in different action types have been eliminated.

In our proposed action representation, each frame contains the whole human body with the full spatial structural information, which shares the advantages with the global representation methods, while each video sequence is sparsely represented by a subset of key poses, which enjoys advantages of local representations.Therefore, the proposed method can be regarded as a semi-holistic representation of human actions and inherits the advantages of global features in the spatial dimensions and, meanwhile, has the superiority of local features in the temporal axis. Besides, another motivation of using pose representation is that it is possible to recognize actions by just looking at a few poses without examining the whole sequence.

After obtaining the feature representations, the bag-of-features model and its variants plus SVM is the standard method for action recognition. Recently, a classifier called Naive Bayes Nearest Neighbor (NBNN) [35] is proposed for image classification tasks based on sets of local features. The NBNN classifier avoids the quantization error in the bag-of-features model by computing 'Image-to-Class' distance rather than the 'Image-to-Image' distance. McCann and Lowe [36] later developed the local Naive Bayes Nearest Neighbor (LNBNN) classifier with a remarkable increase of accuracy and decrease of running time compared with NBNN.

Inspired by LNBNN, in this chapter, we propose to further improve the NBNN model and introduce an enhanced version of NBNN, named weighted local Naive Bayes Nearest Neighbor (WLNBNN), for the final action classification. We will show experimentally that WLNBNN is a more efficient and accurate classifier for action recognition.

2.1.1 Contributions

The contributions of this work lie in the following three aspects:

(1) We propose to describe action poses by extensive pyramidal features (EPF), which can effectively capture the orientation, intensity and contour properties in a pose, and are tolerant to shifts and scaling.

(2) The AdaBoost learning algorithm is employed for selecting discriminative poses, which can significantly improve the performance of action recognition.

(3) A new classifier named weighted local Naive Bayes Nearest Neighbor (WLNBNN) is proposed for final action classification which outperforms other classifiers, *e.g.*, NBNN.

2.2 Literature Review

Action recognition based on pose representation has been applied in a large number of previous works. Thurau and Hlavác [78] presented a method to recognize actions in videos or images based on primitive pose representations which are described by Histogram of Oriented Gradients (HOG) [79]. Niebels and Fei-Fei [80] have employed a pose-based method using a hierarchical model of shape and appearance for action recognition. Yang et al.[81] proposed a approach that treats the pose of the person in an image as latent variables and recognizes human actions from still images. Shao *et al.* [32] combined the pose silhouettes with correlograms [82] to achieve action recognition by adopting the K Nearest Neighbor

(KNN) classifier.

In a video sequence, however, not all of the poses are informative and discriminative for action recognition. Some poses carry neither complete nor accurate information and would even contain common patterns shared by various action types. Since these poses in a video sequence cannot well represent the action and would cause confusion during the classification phase, a great deal of work has been carried out to select the most representative and discriminative poses, *i.e.*, the key poses. To obtain visually distinct representations, Cooper and Foote [83] presented methods for key frame selection based on capturing the similarity to the represented segment and preserving the differences from other segments' key frames, so that different segments will have visually distinct representations. Zhao and Elgammal [84] developed an effective approach for action classification, in which they first described all the poses with the distribution of local motion features and their spatio-temporal arrangements, and then selected a small set of most discriminative poses by comparing their discriminative power for each independent action. Zhuang *et al.* [85] have applied an unsupervised clustering method for key pose selection. Baysal *el at.* [30] selected the most representative and discriminative poses from a set of candidates by ranking the potentiality of each candidate pose in distinguishing an action from others. Cao *et al.* [31] developed a PageRank-based centrality measure to select key poses according to the geometric structure recovered by a manifold learning technique.

However, the above methods all use unsupervised techniques to measure pose similarity or calculate the pose probability distribution for key pose selection. Since such unsupervised methods do not consider the relationship among poses from different classes and are not able to select the very discriminative poses from each actin type. Accordingly, in this chapter, we propose to use a supervised method, *i.e.*, the AdaBoost algorithm, to select a subset of key poses for action classification.

AdaBoost as a popular machine learning algorithm is widely used in computer vision. A pyramidal architecture has been developed by Fathi and Mori [86] to extract boosted mid-level motion features for action recognition. Shen and Bai [87] have combined Gabor wavelet features with AdaBoost selection algorithm for image classification. Due to its remarkable performance on various vision tasks, we adopt the AdaBoost algorithm to select discriminative key poses for action representation.

The use of the AdaBoost algorithm [88] for feature selection in computer vision is fairly recent although hitherto has only been applied to 2D data. Shan *et al.* [89] have used image tilings of local binary patterns (LBPs) for facial expression recognition where each feature ture was constructed from a fixed-size sub-region to yield a complete description of each face image. A small set of the most discriminative LBP features were extracted using the

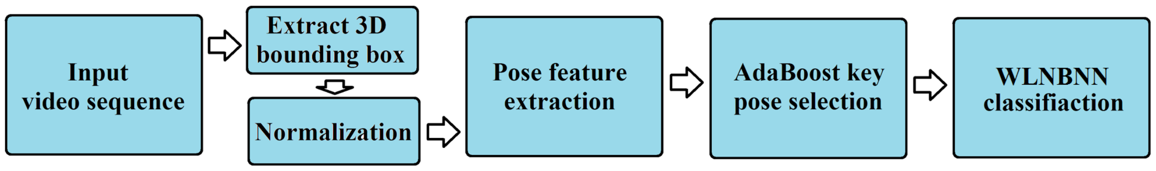


Fig. 2.1 The flowchart of the proposed method

AdaBoost learning algorithm for each independent facial expression and a support-vector machine (SVM) employed for classification.

Zhang et al. [90] have also used LBPs selected using AdaBoost for face recognition. Zhou and Wei [91] have used a similar method combining Gabor wavelet descriptors with AdaBoost to select features for face verification. The boosted-feature selection method has also been employed for real-time object detection [92] and fast pedestrian recognition [93] with improvements in recognition accuracy, compared to the original methods. Laptev [94] has applied the method to retrieving actions in movies.

2.3 Methodology

Our recognition system is comprised of three principal stages. **1)** Pose description: for each video sequence, the extensive pyramidal features are extracted from each frame to represent the pose appearing in it. **2)** Pose selection: the AdaBoost algorithm is adopted to select the most discriminative key poses for each video sequence to represent the corresponding action. **3)** Action recognition: based on the boosted poses, a newly proposed classifier, *i.e.*, weighted local Naive Bayes Nearest Neighbor (WLNBNN), is employed for action classification. The flowchart of the proposed method is illustrated in Fig. 2.1. We will detail the three stages in the following subsections.

2.3.1 Extensive Pyramidal Features

Given a frame containing a pose, we would like to describe it with informative features extracted from it. The descriptor is expected to capture the orientation, intensity and contour information which is the main cue of a pose. We therefore employ Gabor filters, the Gaussian pyramid and the wavelet transform to obtain extensive pyramidal features for pose representation.

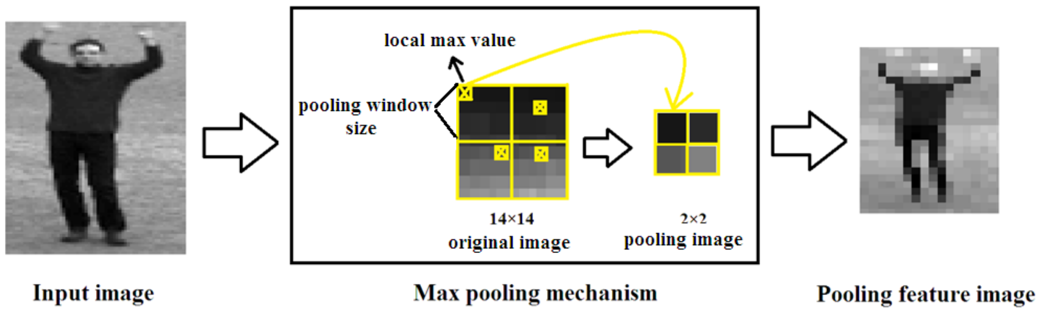


Fig. 2.2 Illustration of the max pooling mechanism

Gabor feature map

Gabor filtering is regarded as the most effective method to obtain the orientation information, which is widely used in feature extraction due to its property of orientation selection. To mimic the biological mechanism of the visual cortex, Riesenhuber and Poggio [95] proposed the HMAX model composed of four hierarchical feed-forward layers: S1, C1, S2 and C2, in which S1 is obtained by Gabor filtering. Inspired by their work, we convolve each pose frame with a bank of Gabor filters with multiple scales and orientations to extract the S1 feature map. More specifically, we adopt Gabor filters with six different scales: 7×7, 9×9, 11×11, 13×13, 15×15, 17×17, and four different orientations: 0, 45, 90, and 135 degrees. As a result, by convolving with $6\times4=24$ Gabor filters, we obtain 24 S1 feature maps. The Gabor filter function is defined as follows:

$$G(x,y)=\exp\left(-\frac{(X^2+\gamma^2Y^2)}{2\sigma^2}\right)cos\left(\frac{2\pi}{\lambda}x\right), \tag{2.1}$$

$$where,\ X=x\cos\theta-y\sin\theta, Y=x\sin\theta+y\cos\theta, \tag{2.2}$$

and (x, y) is the coordinate relative to the center of the filter.

To obtain our Gabor feature maps that are equivalent to C1 in the HMAX model, we perform max pooling operations among the different scales. In the other words, we pick the maximum value across all the S1 feature maps with filter scales in each orientation. The pooling among different scales is defined below:

$$I_{MAX}=\max_{(x,y)}[I_{7\times7}(x,y,\theta_s), I_{9\times9}(x,y,\theta_s),\ldots,I_{17\times17}(x,y,\theta_s)] \tag{2.3}$$

where, I_{MAX} is the output of max pooling and $I_{i\times i}(x,y,\theta_s)$ denotes the feature map with the scale $i\times i$ and the orientation θ_s.

Max pooling is also performed over local neighborhoods with windows varying from

8×8 to 12×12 and a shifting step of 2 pixels. It is regarded as a key mechanism for object recognition in human brain cortex, which can achieve more robust response when human recognize objects with clutter and complex background in the receptive field [95]. An example of max pooling over local neighborhoods is given in Fig. 2.2. The procedure of Gabor feature map extraction is illustrated in Fig. 2.3.

Gaussian center-surround feature map

Center-surround (CS) fields have the properties of edge enhancement, which can effectively work for the detection, location in the human visual system [96]. The features with different scales are enhanced and segregated into a series of sub-band images via the center-surround operation. Gaussian center-surround feature map is also inspired from neuroscience [96] similarity by mimicking perception of nerve cells which commonly respond to the dramatic change of colors (*i.e.*, the dark pixels surrounded by bright ones or the bright pixels surrounded by dark ones). Center-surround operation has been successfully used to capture the intensity information for scene classification. We first construct a seven-level Gaussian pyramid on each frame (pose) of the input action sequence. For one given pose which is viewed as the first level of pyramid, the Gaussian pyramid can be built by successfully convolving a Gaussian filter (with $\sigma = 2$) with several copies of the original pose image with reduced resolutions obtained by down sampling. In this way, we obtain the corresponding Gaussian pyramid. To be precise, the construction of the Gaussian pyramid as follows:

$$W(x,y) = \frac{1}{(\sqrt{2\pi\sigma})^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.4}$$

$$\begin{aligned} &Gaussian_{level=l}(i,j) = \\ &\sum_x \sum_y W(x,y) Gaussian_{level=l-1}(2i+x, 2j+y) \end{aligned} \tag{2.5}$$

where, l denotes the levels of a Gaussian pyramid and (i, j) represents the position of a pixel in the pose image.

Gaussian center-surround feature map is then computed by subtracting point-by-point between different center levels (we use center level=2, 3) and surrounded levels (we use surround level=center level + d, where d=3, 4). However, since scales are different between center levels and surround levels, we need to interpolate images of surround levels to the same size as the corresponding center levels. In this way, the center levels and surround levels can be subtracted point-by-point to generate the relevant sub-band images. As a

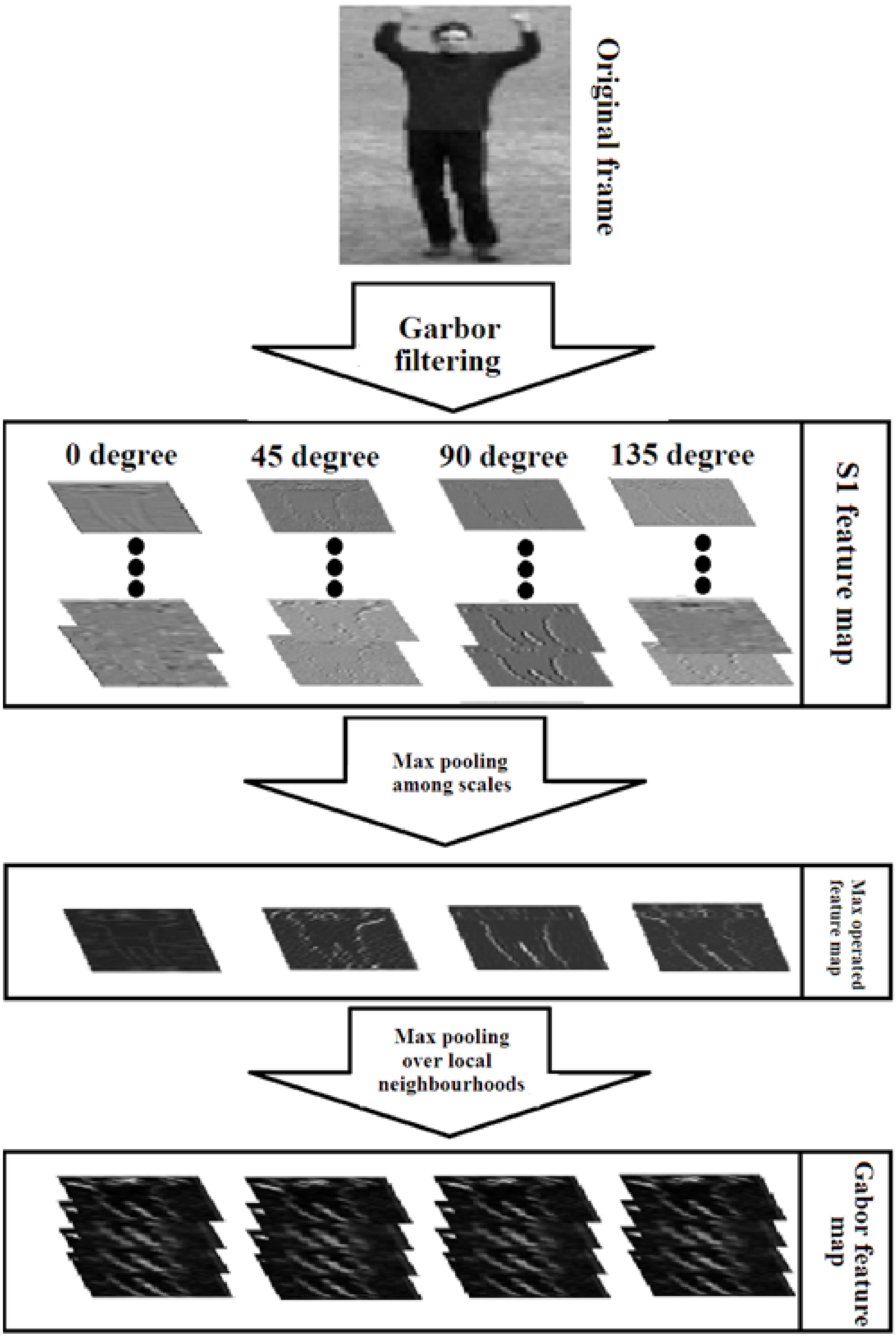


Fig. 2.3 The outline of the Gabor feature map extraction

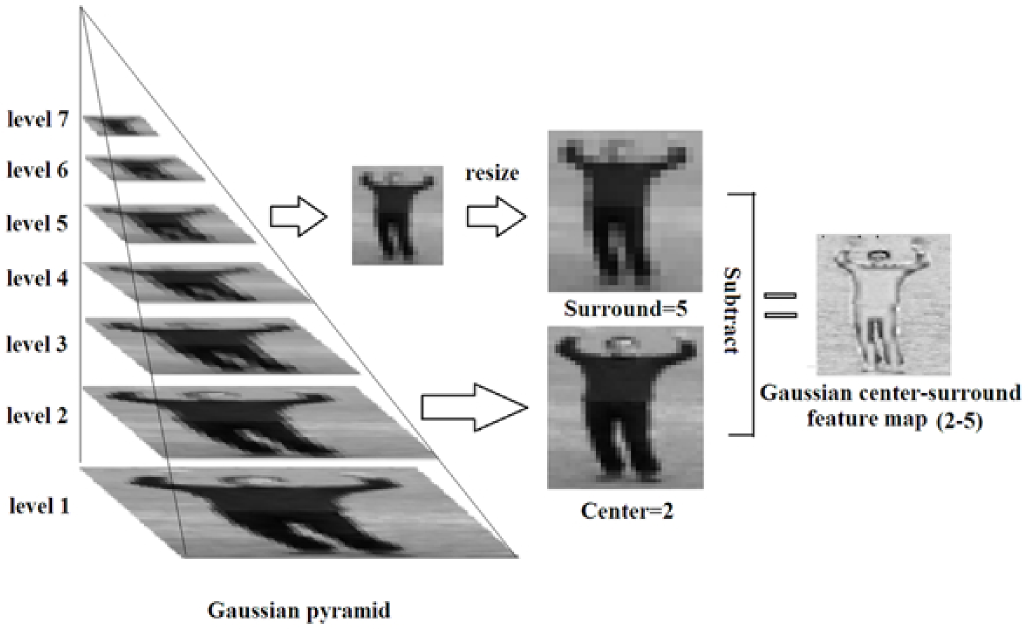


Fig. 2.4 Illustration of the Gaussisan center-surround feature map

result, four levels of feature map (*i.e.*, levels of 2-5, 2-6, 3-6, and 3-7) are calculated as our Gaussian center-surround feature map. An example of the center-surround operation is illustrated in Fig. 2.4.

Wavelet Laplacian pyramid feature map

Wavelet transform [97], [98] has been an efficient way of feature extraction. It decomposes the input image into low frequency and high frequency bands which carry the coarse information and detail information, respectively. In our method, we consider to use the Cohen-Daubechies-Feauveau (CDF) '9/7' wavelet [99] to construct a wavelet Laplacian pyramid, which is proved to be an effective technique for multi-resolution analysis, successfully obtaining the contour information of action poses.

Similar to the Gaussian center-surround feature map, we build our wavelet Laplacian pyramid by first using the five-level 2D CDF '9/7' wavelet decomposition which generates the coefficient matrices of the approximation (cA) and horizontal, vertical and diagonal details (cH, cV, cD, respectively). We utilize the five-level wavelet decomposition on each frame (pose) from a given action sequence and only adopt the approximation (cA) of each level to built a CDF '9/7' pyramid. To further compute the wavelet Laplacian pyramid feature map, we make the difference between each two CDF '9/7' pyramid adjacent levels, which have been already interpolated into the equal size (*i.e.*, $l_i = W_{i+1} - W_i$, where W_i and W_{i+1} are the adjacent levels from the multi-level CDF '9/7' pyramid; l_i denotes the obtained level of the wavelet Laplacian pyramid). For this case, a four-level wavelet Laplacian pyramid feature map has been calculated to extract the contour information of action poses.

Extensive pyramidal feature representation

We obtain our extensive pyramidal feature representation by flattening the Gabor feature maps, Gaussian center-surround feature maps and wavelet Laplacian pyramid feature maps into a 1D feature vector. The obtained extensive pyramidal features provide an informative representation of poses capturing multiple features including orientation, intensity and contour. In addition, Gabor filtering, the Gaussian pyramid and the wavelet pyramid incorporate a multi-resolution analysis, and therefore enjoy the properties of invariance to scaling. To make a compact representation, we further adopt principal component analysis (PCA) to reduce the high dimensional feature vector (1656D) into a low dimensional space by keeping the 99% principal components. The outline of our extensive pyramidal features extraction framework is visualized in Fig. 2.5.

2.3.2 Key-Pose Selection by AdaBoost

Each frame in a raw N-frame action sequence has been represented by the extensive pyramidal features. The AdaBoost learning algorithm is adopted to select the most discriminative poses from a large pose features pool, so as to increase the final classification accuracy and reduce the computational cost.

AdaBoost is one of the most popular machine learning algorithms, which aims to construct a strong classifier from several weak ones. Given a training set $\{(x_1,y_1)...,(x_N,y_N)\}$, where x_i is a pattern and $y_i \in \{+1,-1\}$ is the class label of the corresponding pattern. At first, all training patterns are assigned with the equal weights. In the learning phase, one weak classifier is trained and all patterns are updated. Patterns that are incorrectly classified have their weights increased, and on the contrary, the weights of those patterns that are correctly classified are decreased. After all iterations of training, patterns that are consistently difficult to classify acquire larger weights, while easily classified patterns acquire even s-maller weights. Here we adopt the 'Classification and Regression Trees' (CART) [100] as our basic weak classifiers. The outline of the AdaBoost selection algorithm is shown in Algorithm 2.

Since poses from different classes would share similar features, for instance, some poses from *Running* and *Jogging* in the KTH dataset are quite similar, these poses tend to confuse the classifier in the recognition stage. We would like to select the most discriminative poses for each class. The AdaBoost learning algorithm is then employed to select a subset of

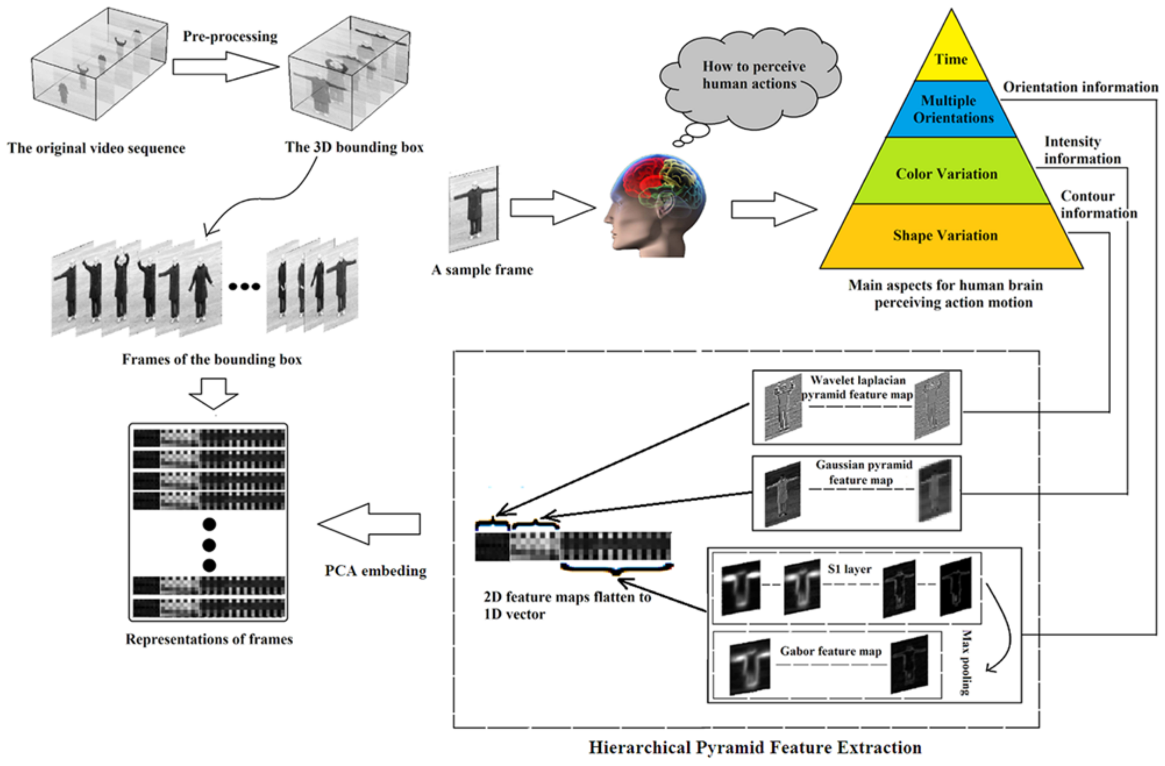

Pre-processing
The original video sequence
The 3D bounding box
Frames of the bounding box
Representations of frames
PCA embeding
A sample frame
How to perceive human actions
Time
Multiple Orientations
Color Variation
Shape Variation
Orientation information
Intensity information
Contour information
Main aspects for human brain perceiving action motion
Wavelet laplacian pyramid feature map
Gaussian pyramid feature map
S1 layer
Gabor feature map
Max pooling
2D feature maps flatten to 1D vector
Hierarchical Pyramid Feature Extraction

Fig. 2.5 The proposed extensive pyramidal features (EPF) extraction procedure: Firstly, bounding boxes are extraction for all sequences; Secondly, simulating the human brain cognition, the three different feature maps are extracted. Thirdly, the feature map is concatenated and then fed to PCA to generate the final EPF representation.

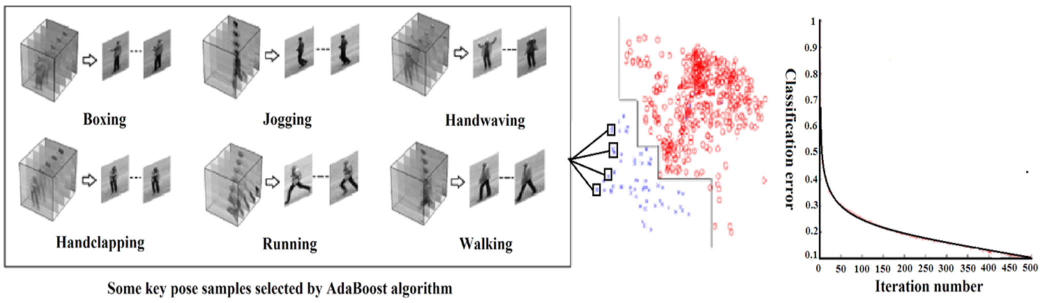


Fig. 2.6 The left sub-figure shows some selected key pose samples from the KTH dataset. The middle sub-figure illustrates the final AdaBoost boundary for pattern classification. The right sub-figure shows the error rate Vs. iterations during the AdaBoost learning.

Algorithm 2 AdaBoost Pose Selection

```
The training set:$(x_1,y_1),...,(x_N,y_N)$, where $x_i$ denotes the sample data, and $y_i \in \{1,-1\}$ stand for positive samples and negative samples respectively.
**First step**
Initialize weights:$D_1(i) = \frac{1}{N}$;
**Second step**
*For* $t = 1,...,T$:
    1. For each feature $i$, a weak classifier $h_t : X \to \{-1,1\}$ is trained and calculated. The error is evaluated with respect to $\varepsilon_t = P_{D_t}(h_t(x_i) \neq y_i)$;
    2. Choose the classifier with the lowest error in each iteration;
    3. Calculate the weight of this weak classifier: $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$;
    4. Update the weight of training data: $D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$;
*End*
**Final step**
Select those features with the smallest weight values.
```

poses which are assigned with lower weights and are the most easily classified patterns in the boosting stage. For each action category we select the top 25% most discriminative poses (with the lowest weights) from a whole sequence as the key poses. It is demonstrated that applying the AdaBoost learning algorithm can successfully reject those unrepresentative poses and dramatically increase recognition rates.

As the traditional AdaBoost learning algorithm is for binary classification, for the classification of multiple classes, a technique named 'One-Against-All decomposition'[1] has been proposed, in which the AdaBoost classifier is trained between one action type and all the other action types in the training set and this procedure is repeated for every action type in the dataset. This decomposition technique is an extension of the arbitrary binary method to a multi-class one. The decomposition method splits the original multi-class problem to a series of simpler binary problems which can be solved by the given binary method. The resulting binary classifiers are then properly combined together to form the multi-class classifier. The related algorithm is shown in Equation 2.6, where $q : \chi \to \varphi$ is the trained multi-classifier, $f_y \in \chi \to R$, $y \in \varphi$ are the real labels. Therefore, the most discriminative poses can be selected from the corresponding sequences for each action type. The key pose selection procedure is illustrated in Fig. 2.6.

$$q(x) = \arg\max_{y\in\varphi} f_y(x) \tag{2.6}$$

[1]http://cmp.felk.cvut.cz/cmp/courses/recognition/eprc/node7.html

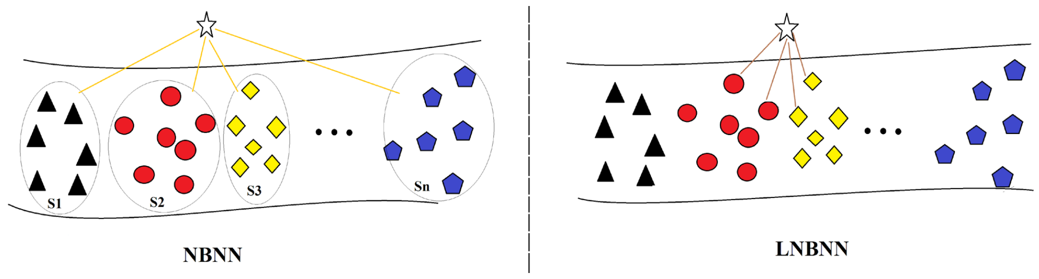


Fig. 2.7 The left sub-figure shows the NBNN algorithm which finds the nearest neighbors from each of the classes (different shapes mean different classes in this figure). The right sub-figure shows the LNBNN algorithm which only searches the local neighbors, and the k nearest neighbors may come from only some of the classes.

2.3.3 Weighted Local Naive Bayes Nearest Neighbor

The SVM kernel machines on the bag-of-features model have dominated the classification on local features, however, it suffers from the quantization error. Boiman *et al.* [35] proposed a simple NN-based classifier named Naive-Bayes Nearest-Neighbor (NBNN). In NBNN framework, ‘Image-to-Class’ distances are directly computed without any descriptor quantization. Although NBNN is significantly simple, efficient with no training phase, its performance can still achieve competitive results compared with the top leading learning-based image classifiers.

Algorithm 3 WLNBNN classification

```
Require: An exemplar set $\{p_i\}$ consisting of descriptors from all classes.
Require: A query Q consisting of descriptors from a certain class.
For all descriptors $d_i \in Q$ do;
  $\{p_1, p_2, p_3, \ldots, p_k\} \leftarrow NN(d_i, k)$, (NN means finding the nearest neighbor);
  For all categories C found in the k nearest neighbors do;
    $dist_C = (min_{p_j|Class(p_j)=C} \| d_i - p_j \|^2) \times weight$
    $(weight = [k/\#(Class(p_j) = C)])$;
    $totals[C] \leftarrow totals[C] + dist_C$;
  End
End
Return $argmin_C totals[C]$.
```

Recently, McCann and Lowe [36] introduced locality to NBNN and proposed the local Naive Bayes Nearest Neighbor (LNBNN) classifier for image classification which outperforms the original NBNN classifier. For a query from the test set, LNBNN searches k nearest neighbors regardless of their class labels, instead of finding the nearest neighbor in each class. As a result, the classification problem converts from ’Does this feature descriptor look like one from class A?, class B?, class C?...' to 'What does the feature descriptor look like?'. Fig. 2.7 shows the schematic diagram of LNBNN and more relative details can be found in [36].

Based on the LNBNN algorithm, we further improve the NBNN classifier by assigning weights to the Euclidean distances between a query descriptor and the nearest exemplar set. The weights are related to the number of descriptors of each action type appearing in KNN search. If m poses in the k nearest neighbors of the descriptor d_i fall in a certain class, the weight for this class is k/m. Our detailed WLNBNN is visualized in Algorithm 3. Since the size of the exemplar set has been dramatically reduced by the AdaBoost selection process compared to the primal number of raw data, and the WLNBNN classifier applies an k-nearest-neighbor search which is much faster than searching the nearest neighbor in each class, the running time of our final classification is greatly reduced.

2.4 Experiments and Results

We systematically test our proposed method on four different action datasets, KTH [101], Weizmann [102], IXMAS [103] and HMDB51 [104].

2.4.1 Datasets

The **KTH** dataset, is regarded as a benchmark action dataset containing 599 video clips with six human action classes, including walking, jogging, running, boxing, hand waving and handclapping, performed by 25 subjects in four different scenarios. We adopt the leave-one-person-out cross validation, *i.e.*, videos of 24 subjects are used as training data and videos of the remaining one subject are used for testing. Following the pre-processing step mentioned in [105], the coarse 3D bounding boxes are extracted from all the raw action sequences and further normalized into an equal size of $100 \times 100 \times 60$.

The **Weizmann** dataset contains 93 video sequences showing nine different people, each of which performing 10 different actions. We extract the bounding boxes by using foreground masks which are provided with the original dataset and normalize them into the size of $100 \times 100 \times 60$. The same leave-one-person-out evaluation scheme is adopted on this dataset.

The **IXMAS** dataset is a multi-view dataset which contains 11 action classes. Each action is repeatedly executed three times by ten actors and recorded by five cameras simultaneously. These actions are checking watch, crossing arms, scratching head, sitting down, getting up, turning around, walking, waving, punching, kicking and picking up. We pre-process all the action sequences into the size of $100 \times 100 \times 80$ and repeat the experiments we have mentioned above. The same leave-one-person-out evaluation scheme is adopted on this dataset.

The **HMDB51** dataset collects 6849 action sequences from various movies and online videos. In our case, we adopt 2241 sequences from 19 body action categories (*i.e.*, cartwheel, clap hands, climb, climb stairs, dive, fall on the floor, backhand flip, handstand, jump, pull up, push up, run, sit down, sit up, somersault, stand up, turn, walk and wave.) as our research data. In our experiments, bounding boxes have been extracted from all the sequences through masks released with the dataset and initialized into the size of $250 \times 300 \times 150$. Following the methods in [104], we perform our evaluation on the three split settings.

2.4.2 Comparison Results

The experimental results on the KTH dataset are illustrated in Table 7.2. Our method achieves an average recognition rate of 94.8%, which outperforms the state-of-the-art techniques. In addition, we have evaluated the newly proposed WLNBNN classifier. From Table 7.2, we can see that WLNBNN outperforms the baseline SVM, LNBNN and NBNN classifers. We also plotted the confusion matrix of accuracies for the KTH dataset in Fig. 2.9. We can see from the confusion matrix that our method can give excellent recognition rates on actions such as *Boxing*, *Handclapping*, *Handwaving* and *Walking*, however, the greatest confusion exists between *Jogging* and *Running* which are two actions that would intuitively seem hard to distinguish reliably. Fig. 2.8 shows some failure cases in the recognition. It can be observed that two actions, i.e., Jogging and Running, which share similar motion patterns, are hard to distinguish even by human eyes. In addition, the comparison between with and without AdaBoost indicates that the employed AdaBoost algorithm can successfully select the most discriminative poses and improve the recognition performance.

The results on the Weizmann dataset are shown in Table 6.2. As expected, our method achieves a perfect 100% recognition rate, since the Weizmann dataset is a relatively 'easy' dataset with greater inter-class variations. For comparison, we have also included results of previous work. The proposed WLNBNN classifier outperforms the SVM, NBNN and LNBNN classifiers and the AdaBoost pose selection contributes to the recognition perfor-

[2]The value of k is correlated to the highest classification accuracy by applying WLNBNN shown in Fig. 2.12.

[3]The value of k is correlated to the highest classification accuracy by applying LNBNN shown in Fig. 2.12.

[4]Here we adopt AdaBoost algorithm as a classifier directly by considering the weight value of each pose's feature together to make a combined judgment for action recognition.

[5]We utilize the linear-SVM for classification.

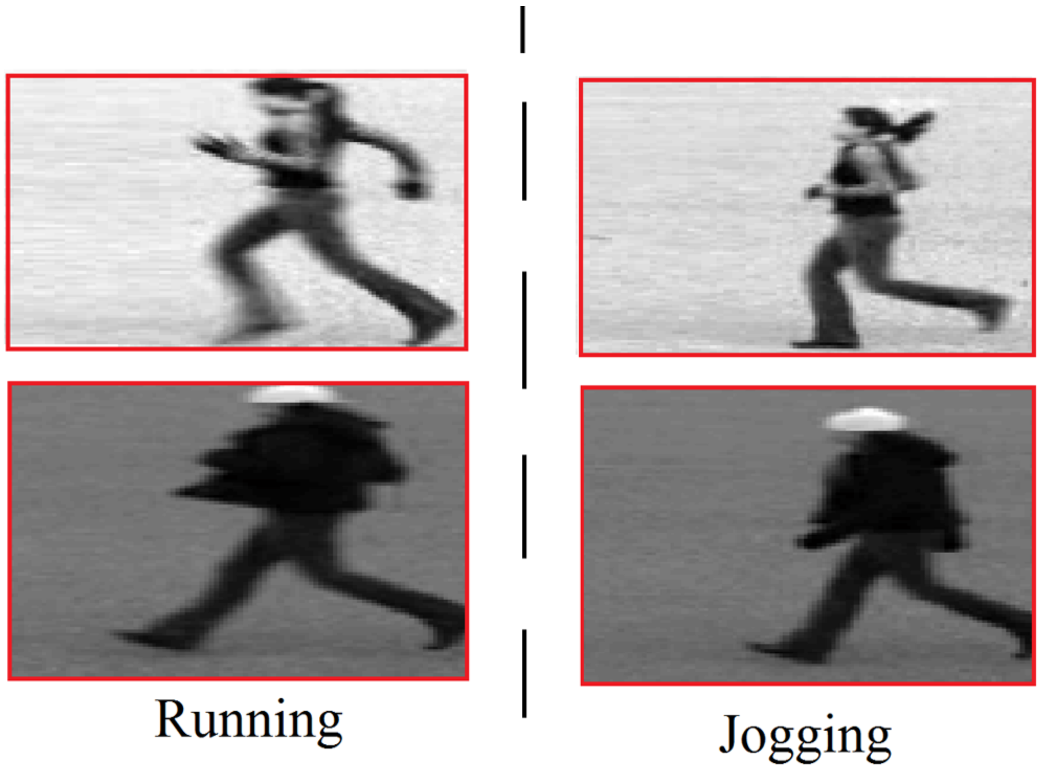


Fig. 2.8 Some failure recognition samples on the KTH dataset.

Table 2.1 Comparison of action recognition accuracies in percentage (%) on the KTH dataset with different methods

Actions Methods	Boxing	Handclapping	Handwaving	Jogging	Running	Walking	Average
EPF+AdaBoost+WLNBNN (k=25)[2]	95	97	**98**	**89**	87	98	**94.8**
EPF+AdaBoost+LNBNN (k=25)[3]	96	96	98	89	88	97	94.0
EPF+AdaBoost+NBNN	93	95	95	86	83	96	91.8
EPF+WLNBNN (k=25)	91	93	94	84	81	93	89.3
EPF+AdaBoost[4]	88	90	92	79	74	91	85.7
EPF+BOW+SVM[5]	89	91	93	84	80	94	88.5
Dollár *et al.* [14]	93	77	85	57	85	90	81.2
Niebles *et al.* [16]	**98**	86	93	53	**88**	82	83.3
Taylor *et al.* [106]	-	-	-	-	-	-	90.0
Ji *et al.*[27]	90	94	97	84	79	97	90.2
Jhuang *et al.* [107]	92	**98**	92	85	87	96	91.7
Schindler and van Gool [29]	-	-	-	-	-	-	92.7
Le *et al.* [108]	-	-	-	-	-	-	93.9
Liu and Shah [109]	**98**	95	96	**89**	87	**100**	94.2

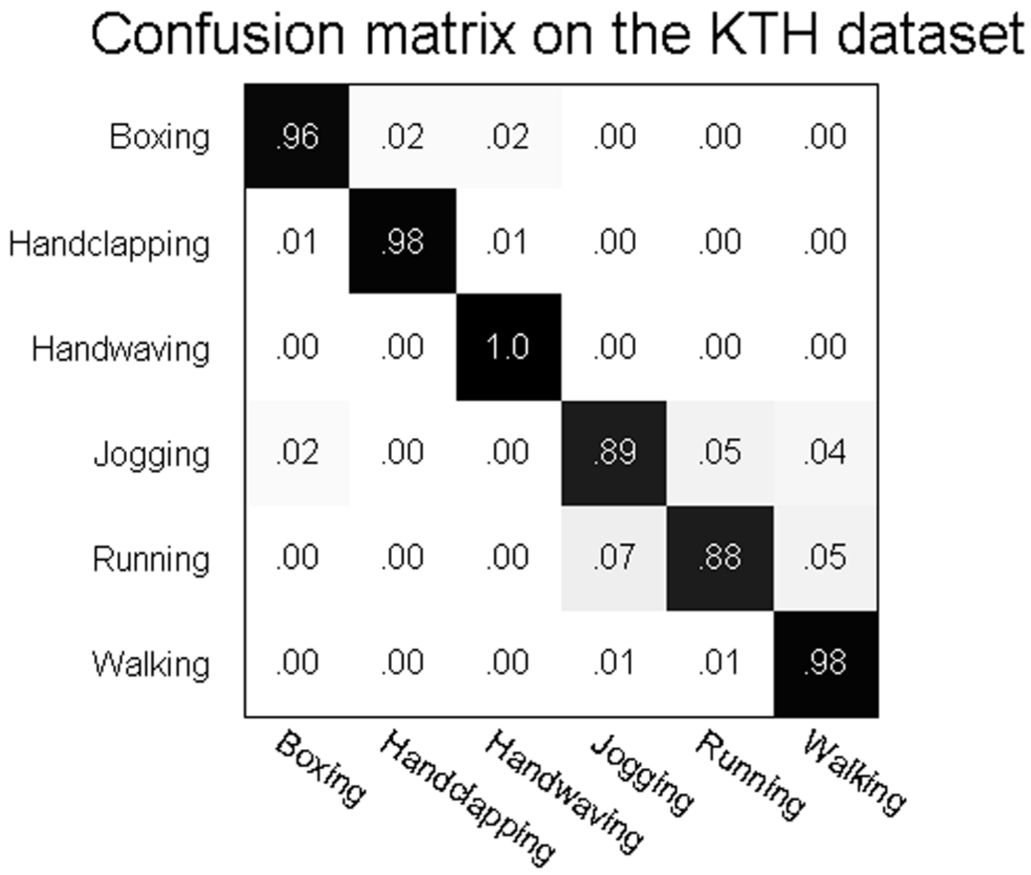


Fig. 2.9 The confusion matrix of classification results on the KTH dataset.

Table 2.2 Comparison of action recognition accuracies in percentage (%) on the Weizmann dataset with different methods

Methods	Accuracy
EPF+AdaBoost+WLNBNN (k=25)[2]	**100**
EPF+AdaBoost+LNBNN (k=30)[3]	100
EPF+AdaBoost+NBNN	99.2
EPF+WLNBNN (k=25)	98.5
EPF+AdaBoost[4]	95.2
EPF+BOW+SVM[5]	97.8
Niebles *et al.* [16]	72.8
Jhuang *et al.* [107]	98.8
Yang *et al.* [110]	99.4
Fathi and Mori [86]	**100**

mance as well.

On the IXMAS dataset, we have evaluated our method not only on each single view, but also on multiple views by combining the data from all five cameras. The overall accuracy we obtain by applying multi-view fusion achieves a recognition rate of 94.5% which significantly exceeds all the other results listed in Table 6.3. For each single view, our method achieves the best results among all the methods. The WLNBNN classifier outperforms the SVM, NBNN and LNBNN classifiers and the AdaBoost pose selection improves the accuracies both on single view and multiple views. Fig. 2.10 shows the confusion matrix of the multiple-view fusion results on the IXMAS dataset. Our method can achieve 100% recognition rates on several action categories.

The HMDB51 dataset is a quite challenging dataset, however, our method can still provide relatively good recognition rates on three split settings based on [104]. As far as we are aware, this is the first report of action recognition on this subset of 19 body action categories, therefore, we do not compare with other recognition results, and only present ours in Table 6.4. Consistent with the other three datasets, the proposed WLNBNN classifier performs better than the SVM, NBNN and LNBNN classifiers on this dataset. The AdaBoost pose selection improves the results as well. The confusion matrix is plotted in Fig. 2.11, which demonstrates that our method gives a reasonable and meaningful recognition for each of the action category.

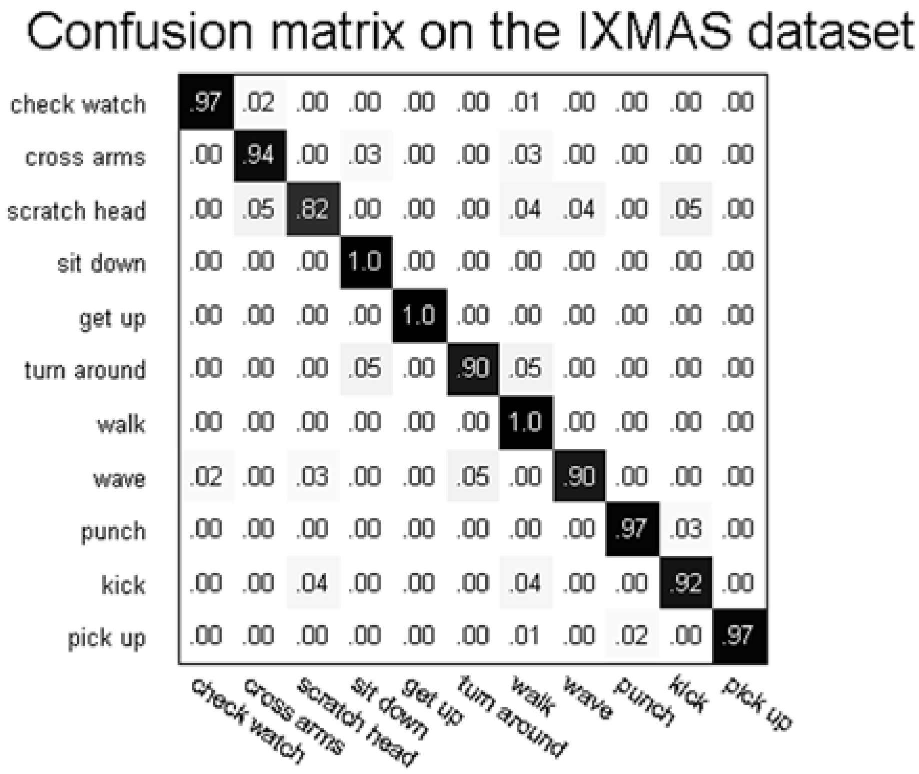


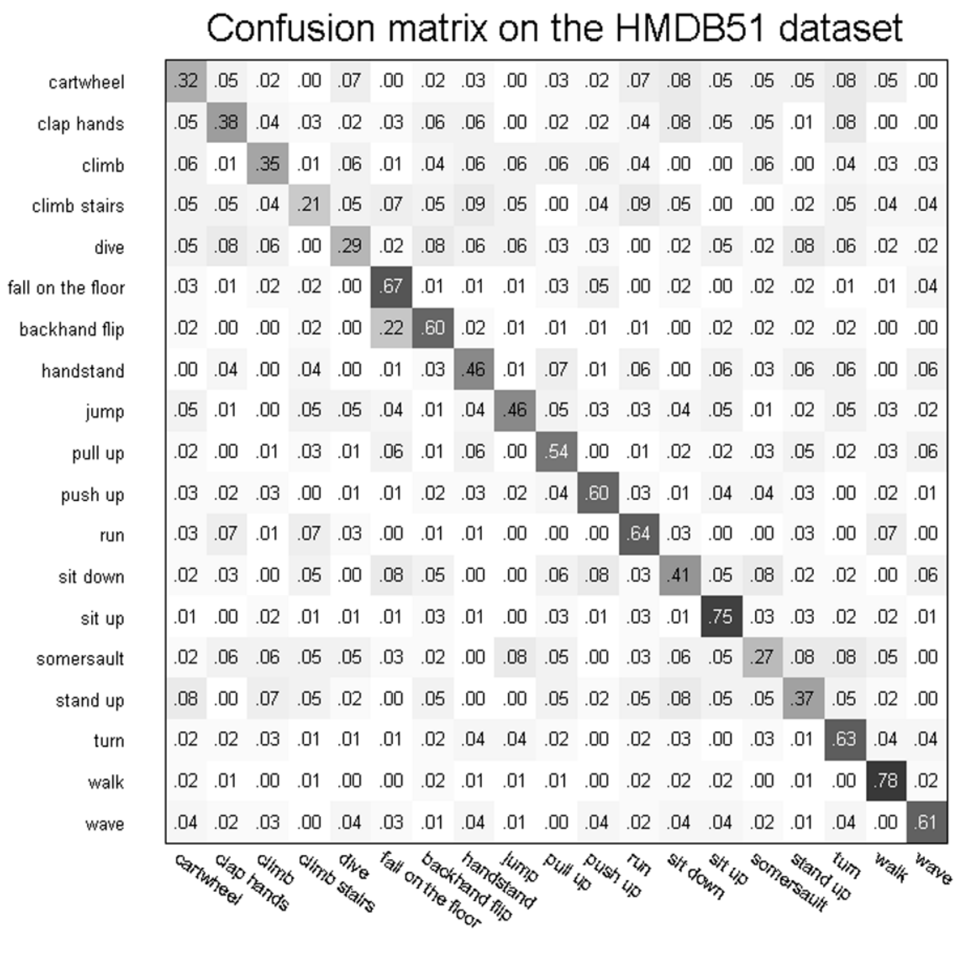
Fig. 2.10 The confusion matrix of the multiple camera fusion results on the IXMAS dataset.

Table 2.3 Comparison of action recognition accuracies in percentage (%) on the IXMAS dataset with different methods

Actions / Methods	Camera 1	Camera 2	Camera 3	Camera 4	Camera 5	Camera 1-5 fusion
EPF+AdaBoost+WLNBNN (k=30)[2]	**85.6**	**91.2**	**88.6**	**86.2**	**82.3**	**94.5**
EPF+AdaBoost+LNBNN (k=25)[3]	83.1	90.0	86.2	84.9	82.0	94.0
EPF+AdaBoost+NBNN	84.1	88.2	85.8	85.1	79.4	93.1
EPF+WLNBNN (k=30)	82.0	86.4	83.5	83.1	75.9	91.7
EPF+AdaBoost[4]	79.2	82.5	83.1	80.5	70.7	87.7
EPF+BOW+SVM[5]	81.4	84.8	86.1	82.7	77.3	89.6
Varma and Babu [111]	76.4	74.5	73.6	71.8	60.4	81.3
Liu and Shah [109]	76.7	73.3	72.1	73.1	-	82.8
Wu *et al.* [112]	81.9	80.1	77.1	77.6	73.4	88.2
Weinland *et al.* [113]	-	-	-	-	-	93.3

Table 2.4 Classification accuracies (%) on the HMDB51 dataset

Methods \ Splits	Split1	Split2	Split3	Average
EPF+AdaBoost+WLNBNN (k=40)[2]	**49.1**	**54.3**	**44.6**	**49.3**
EPF+AdaBoost+LNBNN (k=20)[3]	47.3	51.1	45.7	48.0
EPF+AdaBoost+NBNN	47.3	52.5	42.9	47.6
EPF+WLNBNN (k=40)	45.0	49.4	40.9	45.1
EPF+AdaBoost[4]	43.8	40.6	38.1	40.8
EPF+BOW+SVM[5]	45.7	48.4	39.2	44.4



	cartwheel	clap hands	climb	climb stairs	dive	fall on the floor	backhand flip	handstand	jump	pull up	push up	run	sit down	sit up	somersault	stand up	turn	walk	wave
cartwheel	.32	.05	.02	.00	.07	.00	.02	.03	.00	.03	.02	.07	.08	.05	.05	.05	.08	.05	.00
clap hands	.05	.38	.04	.03	.02	.03	.06	.06	.00	.02	.02	.04	.08	.05	.05	.01	.08	.00	.00
climb	.06	.01	.35	.01	.06	.01	.04	.06	.06	.06	.06	.04	.00	.00	.06	.00	.04	.03	.03
climb stairs	.05	.05	.04	.21	.05	.07	.05	.09	.05	.00	.04	.09	.05	.00	.00	.02	.05	.04	.04
dive	.05	.08	.06	.00	.29	.02	.08	.06	.06	.03	.03	.00	.02	.05	.02	.08	.06	.02	.02
fall on the floor	.03	.01	.02	.02	.00	.67	.01	.01	.01	.03	.05	.00	.02	.00	.02	.02	.01	.01	.04
backhand flip	.02	.00	.00	.02	.00	.22	.60	.02	.01	.01	.01	.01	.00	.02	.02	.02	.02	.00	.00
handstand	.00	.04	.00	.04	.00	.01	.03	.46	.01	.07	.01	.06	.00	.06	.03	.06	.06	.00	.06
jump	.05	.01	.00	.05	.05	.04	.01	.04	.46	.05	.03	.03	.04	.05	.01	.02	.05	.03	.02
pull up	.02	.00	.01	.03	.01	.06	.01	.06	.00	.54	.00	.01	.02	.02	.03	.05	.02	.03	.06
push up	.03	.02	.03	.00	.01	.01	.02	.03	.02	.04	.60	.03	.01	.04	.04	.03	.00	.02	.01
run	.03	.07	.01	.07	.03	.00	.01	.01	.00	.00	.00	.64	.03	.00	.00	.03	.00	.07	.00
sit down	.02	.03	.00	.05	.00	.08	.05	.00	.00	.06	.08	.03	.41	.05	.08	.02	.02	.00	.06
sit up	.01	.00	.02	.01	.01	.01	.03	.01	.00	.03	.01	.03	.01	.75	.03	.03	.02	.02	.01
somersault	.02	.06	.06	.05	.05	.03	.02	.00	.08	.05	.00	.03	.06	.05	.27	.08	.08	.05	.00
stand up	.08	.00	.07	.05	.02	.00	.05	.00	.00	.05	.02	.05	.08	.05	.05	.37	.05	.02	.00
turn	.02	.02	.03	.01	.01	.01	.02	.04	.04	.02	.00	.02	.03	.00	.03	.01	.63	.04	.04
walk	.02	.01	.00	.01	.00	.00	.02	.01	.01	.01	.00	.02	.02	.02	.00	.01	.00	.78	.02
wave	.04	.02	.03	.00	.04	.03	.01	.04	.01	.00	.04	.02	.04	.04	.02	.01	.04	.00	.61

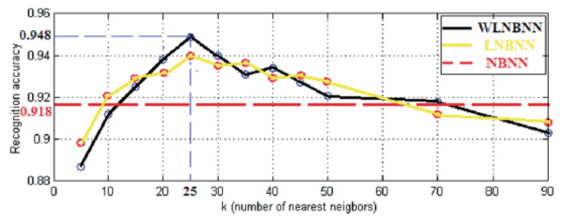
Fig. 2.11 The confusion matrix of classification results on the HMDB51 dataset.

Table 2.5 The comparison of computational complexity (seconds) between NBNN and WLNBNN.

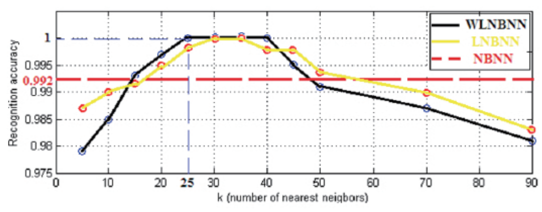
Dataset / Methods	KTH	Weizmann	IXMAS	HMDB51
NBNN	37s	22s	92s	1105s
WLNBNN	23s	14s	56s	227s

Table 2.6 Evaluation of individual features on the KTH, the Weizmann, the IXMAS and the HMDB51 datasets. NB:(1) for WLNBNN classification, the number of nearest neighbors k on each dataset is consistent in previous tables. (2) the accuracies of the IXMAS dataset in this table denote the multi-view fusion results.

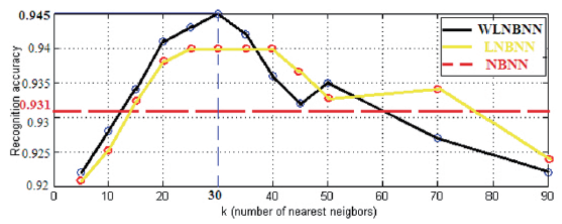
Dataset / Methods	KTH	Weizmann	IXMAS	HMDB51
Gabor feature map+AdaBoost+WLNBNN	87.5	95.2	89.4	42.1
Gaussian center-surround feature map+AdaBoost+WLNBNN	89.2	94.9	90.2	43.4
Wavelet Laplacian pyramid feature map+AdaBoost+WLNBNN	85.0	93.7	86.1	35.5
EPF+AdaBoost+WLNBNN	**94.8**	**100**	**95.5**	**49.3**



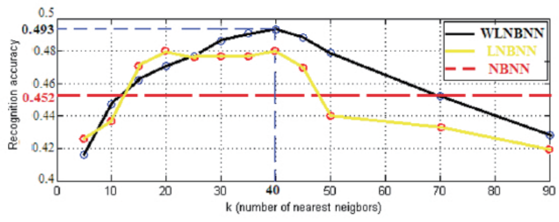
(a) The KTH dataset



(b) The Weizmann Dataset



(c) The IXMAS Dataset



(d) The HMDB51 Dataset

Fig. 2.12 Evaluation of the effects of k on the performance of the proposed method on the KTH, the Weizmann, the IXMAS and the HMDB51 datasets.

Table 2.7 The analysis of the computational complexity for the entire system on the KTH dataset

modules / Methods	EPF extraction	AdaBoost selection	WLNBNN classification	Accuracy
EPF+AdaBoost+WLNBNN (k=40)	1.02×10^3 seconds	2.24×10^4 seconds	23 seconds	94.8%
EPF+WLNBNN (k=40)	1.02×10^3 seconds	-	85 seconds	89.3%

2.4.3 Performance Analysis

To evaluate the parameters of the proposed method, we have also conducted analysis experiments on the four datasets. As both the WLNBNN and LNBNN classifiers use k nearest neighbors search, we investigated the effects of k on the performance of WLNBNN and LNBNN. The results on the four datasets are shown in Fig. 2.12. The proposed WLNBNN classifier achieves higher accuracies than the LNBNN and NBNN classifiers on all the four datasets.

In addition, as in the representation of poses, we combine Gabor features, the Gaussian pyramid and the wavelet pyramid, we would like to evaluate the individual contribution of each feature to the representation. The results are illustrated in Table 6.4. It is obviously manifested that the extensive pyramidal features, *i.e.*, the combination of the Gabor features, the Gaussian pyramid and the wavelet pyramid, outperform any individual feature. The results validate the effectiveness of the proposed extensive pyramidal features.

To demonstrate the efficiency of the proposed WLNBNN classifier, we compare the classification time between NBNN and WLNBNN on four different datasets in Table 6.7. The results show that our WLNBNN classifier runs faster and more accurate than the original NBNN.

Additionally, we have also evaluated the computational cost of each component and the whole system. Table 6.5 shows the time comparison of the methods with and without AdaBoost selection on the KTH dataset. From the results, we can observe that the AdaBoost key pose selection is time-consuming, however, the computational time in the classification phase is significantly reduced while producing a higher accuracy. The same trend would exist on the other three datasets.

2.5 Summary

In this chapter, we have presented a new method based on key pose selection for human action recognition. Poses from each video sequence are described by invariant and informative extensive pyramidal features constructed by computing relevant Gabor feature map, Gaussian pyramid feature map and wavelet Laplacian pyramid feature map. AdaBoost is then employed to learn the most discriminative key poses to represent actions. With the boosted poses for each action, a new classifier named weighted local naive bayes nearest neighbor (WLNBNN) is proposed for action classification. We have further systematically evaluated our proposed method on four different datasets, *i.e.*, the KTH dataset, the Weizmann dataset, the IXMAS dataset and the HMDB51 dataset, and obtained superior results for action recognition over previously published works. Typically, the AdaBoost algorithm takes much time to learn the discriminative key poses, however, it only needs to be carried out once on the training set and the computational cost will be greatly reduced after the key pose selection.

Although, these features selecting from original feature pool demonstrate their discriminant in current research domain, for some more realistic and complex scenarios, this kind of selective features have been proved still not effective and flexible for task-specific problems. Thus, in the next chapter, the new machine learned features via genetic programming (GP) are introduced for dynamic gesture recognition.

Chapter 3

Dynamic Hand Gesture Recognition Using Genetic Programming

3.1 Overview

In the last section, we have thoroughly discussed learning discriminative representations via feature selection for human action recognition. In terms of the experimental results, it indeed leads to a promising results compared with previous reports. However, most of the current works rely on these kinds of hand-crafted features, which heavily limit the effectiveness of the recognition methods. The main motivations of this work can be concluded as follows: "Hand-crafted" schemes to extract information from the video is not meaningful enough; and "Hand-crafted" even deep learned schemes are not adaptive and flexible to task-specific video recognition. Therefore, in this chapter, we successfully apply an evolutionary method *genetic programming* (GP) to synthesize machine learned spatio-temporal descriptors for automatic gesture recognition instead of using hand-crafted descriptors. In our architecture, a set of primitive low-level 3D operators are first randomly assembled as tree-based combinations, which are further evolved generation-by-generation through the GP system, and finally a well performed combination will be selected as the best descriptor for high-level gesture recognition. To the best of our knowledge, this is the first report of using GP to evolve spatio-temporal descriptors for gesture recognition.

We address this as a domain-independent optimization issue and evaluate our proposed method, respectively, on two public dynamic gesture datasets: Cambridge hand gesture dataset and Northwestern University hand gesture dataset to demonstrate its generalizability. The experimental results manifest that our GP-evolved descriptors can achieve better recognition accuracies than state-of-the-art hand-crafted techniques.

The contributions of this work lie in the following two aspects:

(1) We utilize genetic programming (GP) to generate machine-learned spatio-temporal descriptors for gesture recognition.

(2) The proposed methodology and the generated descriptors can be applied to other video analysis related applications directly.

3.2 Literature Review

A comprehensive survey of recent gesture recognition methods can be found in [114]. S-ince this work is on the design of descriptors, we briefly review feature representation-based methods for gesture recognition. Holte *et al.* [115] proposed a view-invariant gesture recognition algorithm by finding motion primitives in the 3D data using 3D optical flow and harmonic motion context representation. A probabilistic Edit Distance classifier is further applied to identify which gesture best describes a string of primitives. Cutler and Turk [116] developed a real-time, view-based gesture recognition system, in which optical flow is estimated and segmented into motion blobs. Gestures are recognized using a rule-based technique charactering the motion blobs. Campbell *et al.* [117] evaluated ten different features combined with HHMs for learning and recognition of gestures and the final results indicate that velocity features are superior to positional features, and partial rotational invariance is sufficient for accomplishing good performance. Bretzner *et al.* [118]represented hand poses using multi-scale color image features with qualitative inter-relations in terms of scale, position and orientation.

Genetic programming (GP), as a powerful machine learning method, has been gradually adopted in computer vision. Trujillo and Olague [119]used GP for the automatic generation of a 2D low-level feature extractor that can be applied to high-level computer vision tasks. Torres *et al.* [120] utilized GP to find a well-performed combination of the similarity functions for image retrieval. Poli [121] applied GP to automatically select an efficient optimal filter for segmenting the brain in medical images. On the same line, a GP-based detector was proposed by Howard *et al.* [122] to detect ships in synthetic aperture radar (SAR) images. Davis *et al.* [123] adopted GP to select the most discriminative features for multivariate data analysis without any prior information. In this chapter, we use GP to automatically synthesize spatio-temporal descriptors from a set of 3D filters and operators for dynamic hand gesture recognition.

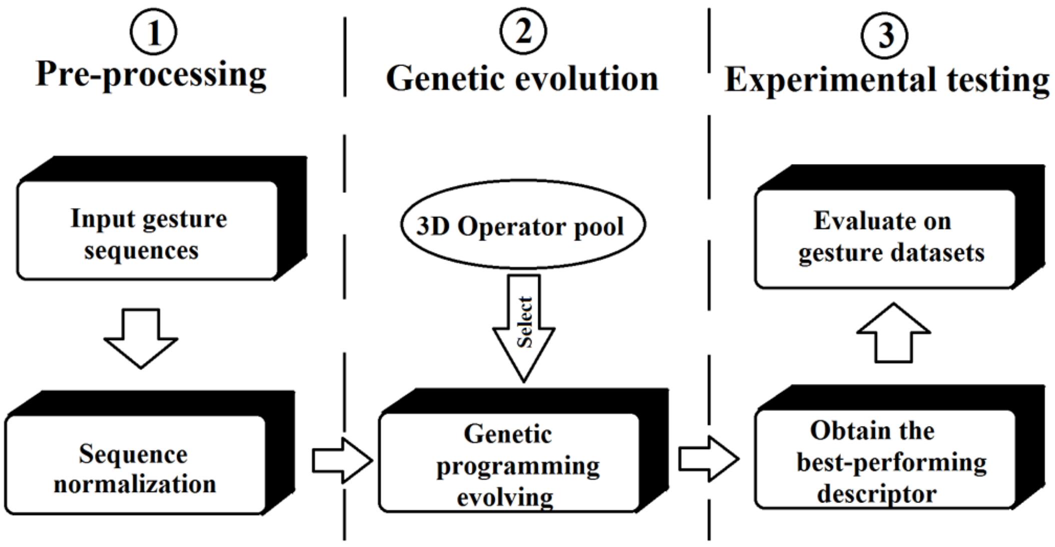


Fig. 3.1 The outline of our architecture.

3.3 Methodology

Automatic gesture recognition has received much attention due to its potential in various applications. In this chapter, we propose a domain-independent machine learning methodology to automatically generate low-level spatio-temporal descriptors for high-level gesture recognition using genetic programming (GP). A group of 3D operators are assembled to construct an effective problem-specific descriptor which is capable of selectively extracting features from hand gestures. The final evolved descriptor, combining the nice properties of those primitive 3D operators, can both extract meaningful features and form a compact gesture representation. We learn our proposed system over a training set, in which descriptors are evolved by maximizing the recognition accuracy through a fitness function, and further evaluate the GP-selected one over a testing set to demonstrate the performance of our method. The architecture of our proposed model is illustrated in Fig. 3.1.

Genetic programming (GP) is an evolutionary computation (EC) [124] technique that automatically solves problems without requiring the user to know or specify the form or structure of the solution in advance. The basic steps in GP is shown in Fig. 3.2. Generally, GP programs can be represented as a tree structure, evolved (by selection, crossover and mutation) through sexual reproduction with pairs of parents being chosen stochastically but biased in their fitness on the task at hand, and finally select the best performing individual as the terminal solution. In our method, each individual in GP represents a candidate spatio-temporal descriptor and is evolved continuously through generations. To establish the architecture of our model, three significant concepts: function set, terminal set and fitness

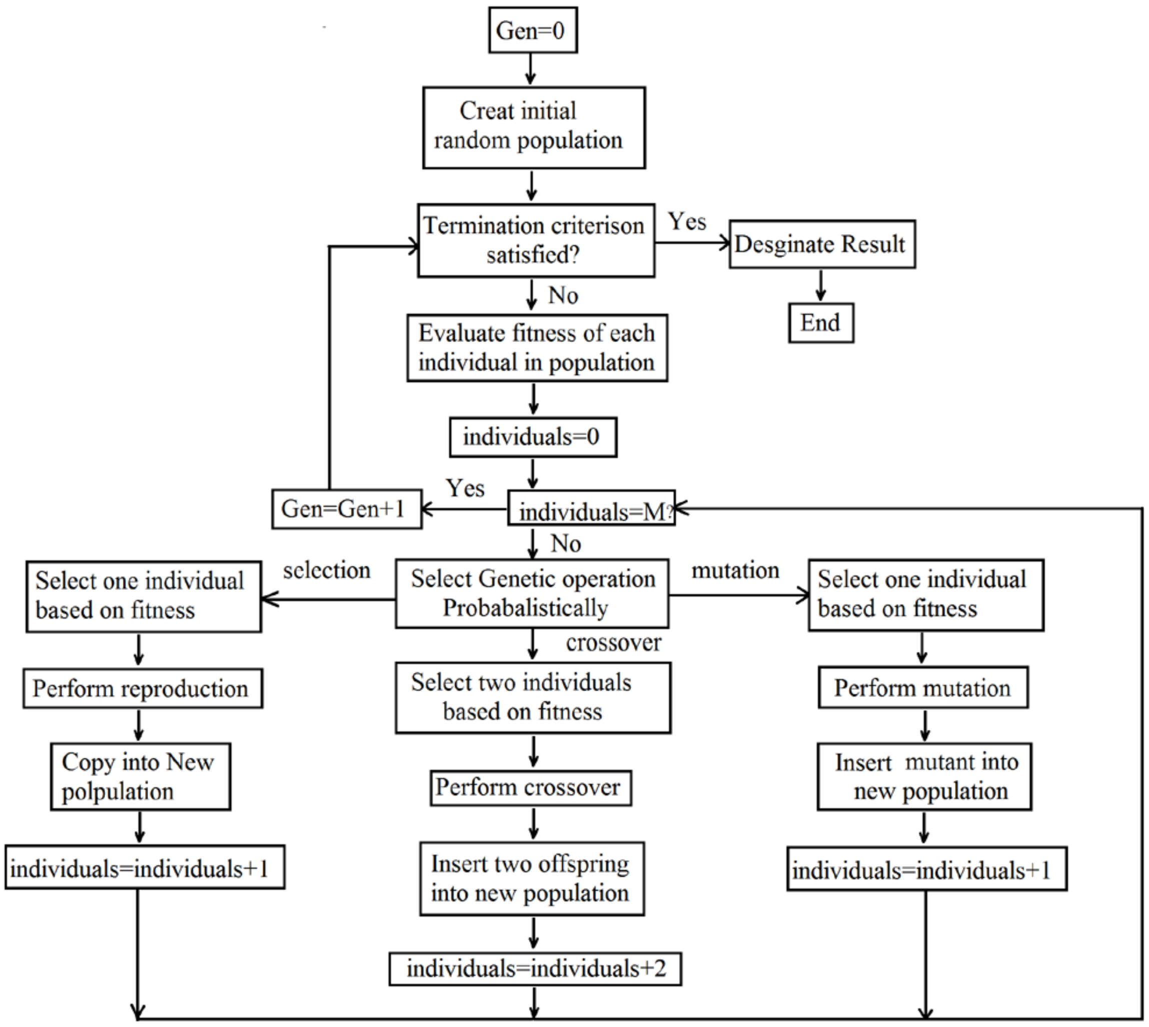

Gen=0
Creat initial random population
Termination criterison satisfied?
Yes
Desginate Result
End
No
Evaluate fitness of each individual in population
individuals=0
Yes
Gen=Gen+1
individuals=M?
No
selection
Select one individual based on fitness
Select Genetic operation Probabalistically
mutation
Select one individual based on fitness
crossover
Perform reproduction
Select two individuals based on fitness
Perform mutation
Copy into New polpulation
Perform crossover
Insert mutant into new population
individuals=individuals+1
Insert two offspring into new population
individuals=individuals+1
individuals=individuals+2

Fig. 3.2 The basic steps in a genetic programming.

function should be first defined.

3.3.1 Function Set and Terminal Set

A key component of GP is the function set which constitutes the internal nodes of the tree and is typically driven by the nature of the problem. To make the GP evolution process fast, more efficient operators that can extract meaningful information from gesture sequences are preferred. Our function set consists of 18 unary operators and 3 binary ones, including processing filters and basic arithmetic functions, as illustrated in Table 3.1.

In our GP structure, we divide our function set into two tiers: filtering tier (bottom tier) and max-pooling tier (top tier). The order of these layers in our structure is always fixed. This means, in any GP-evolved program, the operators in the filtering tier must be located

Table 3.1 Function set in genetic programming

Operator Name	Input	Function Description	Operator Type
Filtering tier			
Gau1	1 Sequence	3D Gaussian smooth filter with $\sigma = 1$	Filter
Gau2	1 Sequence	3D Gaussian smooth filter with $\sigma = 2$	Filter
LoG1	1 Sequence	3D Laplacian of Gaussian filter with $\sigma = 1$	Filter
LoG2	1 Sequence	3D Laplacian of Gaussian filter with $\sigma = 2$	Filter
GBO-0	1 Sequence	3D Multi-scale-max Gabor filter with orientation of 0 degree	Filter
GBO-45	1 Sequence	3D Multi-scale-max Gabor filter with orientation of 45 degrees	Filter
GBO-90	1 Sequence	3D Multi-scale-max Gabor filter with orientation of 90 degrees	Filter
GBO-135	1 Sequence	3D Multi-scale-max Gabor filter with orientation of 135 degrees	Filter
Wavelet	1 Sequence	3D CDF '9/7' wavelet filter	Filter
Aver	1 Sequence	3D Averaging filter with $5 \times 5 \times 5$ sampling window	Filter
Med	1 Sequence	3D Median filter with $5 \times 5 \times 5$ sampling window	Filter
ABS	1 Sequence	Take the absolute value pixel by pixel	Arithmetic
DoF	1 Sequence	Subtract between adjacent frames of the input sequence	Arithmetic
LOG2	1 Sequence	Take the logarithm of with 2 at the bottom for the input sequence	Arithmetic
ADD	2 Sequences	Add two input sequences pixel by pixel	Arithmetic
SUB	2 Sequences	Subtract two input sequences pixel by pixel	Arithmetic
ABSsub	2 Sequences	Absolute subtract two input sequences pixel by pixel	Arithmetic
Max-pooling tier			
Max-pooling5	1 Sequence	3D max-pooling with pooling window size $5 \times 5 \times 5$ on the input sequence	Filter
Max-pooling10	1 Sequence	3D max-pooling with pooling window size $10 \times 10 \times 10$ on the input sequence	Filter
Max-pooling15	1 Sequence	3D max-pooling with pooling window size $15 \times 15 \times 15$ on the input sequence	Filter
Max-pooling20	1 Sequence	3D max-pooling with pooling window size $20 \times 20 \times 20$ on the input sequence	Filter

below the operators in the max-pooling tier. In addition, not all the operators listed in the function set have to be used in a given tree and the same operator can be used more than once. The topology of the tree is essentially unrestricted.

Filtering Tier

In the filtering tier, aiming to extract meaningful features from dynamic hand gestures, we adopt 3D Gaussian filters, 3D Laplacian filers, 3D wavelet filters, 3D Gabor filters and some other sequence processing operators and basic arithmetic functions.

3D Gaussian filters are adopted due to their ability for denoising and 3D Laplacian filters are used for separating signals into different spectral sub-bands. 2D Gaussian and Laplacian operators have been successfully applied to capture intensity features for scene classification in [125]. Wavelet transforms can perform multi-resolution analysis and obtain the contour information of hand gestures by using the 3D CDF '9/7' wavelet filters. 3D Gabor filters are regarded as the most effective method to obtain the orientation information in a sequence. Following Riesenhuber and Poggio's work [126], we simulate the biological mechanism of the visual cortex to define our Gabor filter-based operators. Firstly, we convolve an input gesture sequence with Gabor filters at six different scales ($7 \times 7 \times 7$, $9 \times 9 \times 9$, $11 \times 11 \times 11$, $13 \times 13 \times 13$, $15 \times 15 \times 15$ and $17 \times 17 \times 17$) under a certain orientation (*i.e.*, 0, 45, 90, or

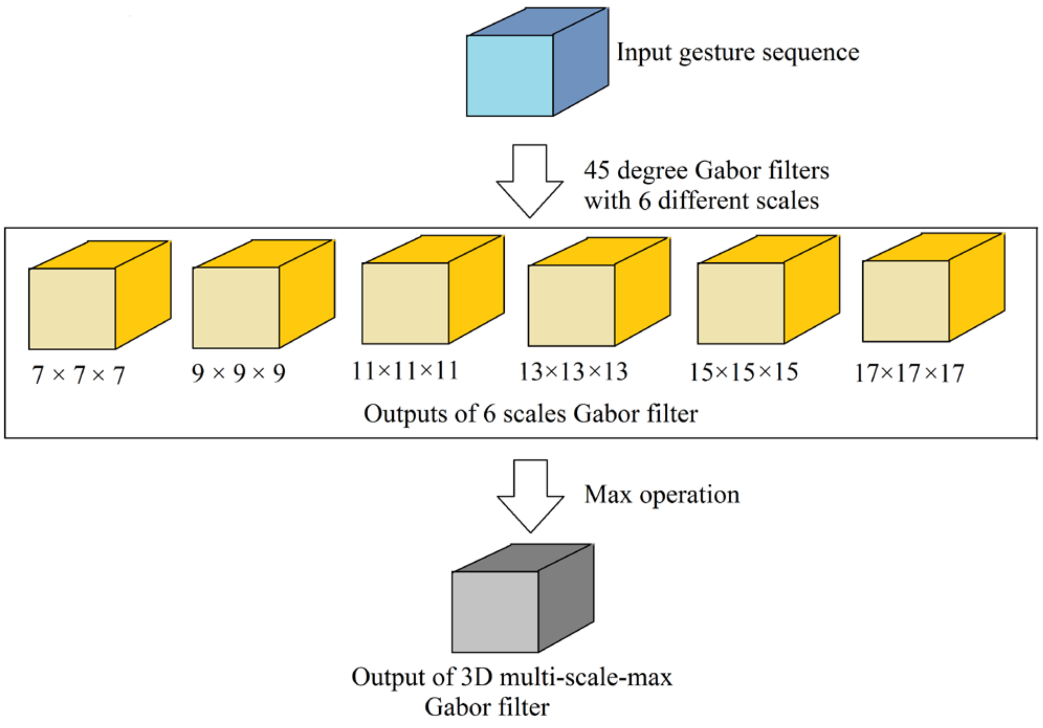


Fig. 3.3 The outline of multi-scale-max Gabor filter. This figure illustrates an example of multi-scale-max Gabor filter with fixed orientation of 45 degrees

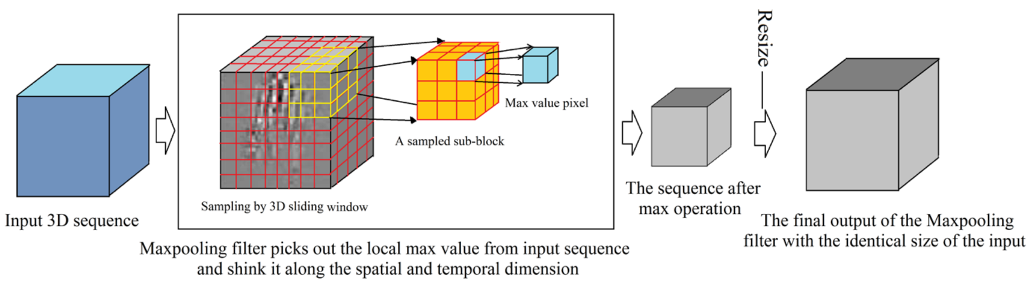

Input 3D sequence
Max value pixel
A sampled sub-block
Sampling by 3D sliding window
Maxpooling filter picks out the local max value from input sequence
and shink it along the spatial and temporal dimension
The sequence after
max operation
Resize
The final output of the Maxpooling
filter with the identical size of the input

Fig. 3.4 Illustration of the mechanism of max-pooling filter

135 degree); We further apply the max operation to pick the maximum value across all six convolved sequences for that particular orientation. Fig. 3.3 illustrates the procedure of our Multi-scale-max Gabor filters for a certain orientation. The max operation among different scales is defined as follows:

$$\begin{aligned} I_{MAX} = \max_{(x,y,z)} [&I_{7\times7\times7}(x,y,z,\theta_s), I_{9\times9\times9}(x,y,z,\theta_s), \\ &\ldots, I_{15\times15\times15}(x,y,z,\theta_s), I_{17\times17\times17}(x,y,z,\theta_s)] \end{aligned} \tag{3.1}$$

where I_{MAX} is the output of the Multi-scale-max Gabor filter. $I_{i\times i\times i}(x,y,z,\theta_s)$ denotes the convolved sequences with the scale $i\times i\times i$ and the orientation θ_s.

Beyond that, several other 3D operators that are common for feature extraction are added to the function set to increase the variety of the selection for composing individuals during the GP running. Basic arithmetic functions are chosen to realize operations such as addition and subtraction of the internal nodes of the tree to make the whole evolution procedure more natural.

Max-pooling Tier

In the max-pooling tier, we include four functions listed in Table 3.1, which are performed over local neighborhoods with windows varying from $5\times5\times5$ to $20\times20\times20$ with a shifting step of 5 pixels. This max-pooling operation (see Fig. 3.4) is a key mechanism for object recognition in the cortex and provides a more robust response, successfully tolerating shift and scaling, in the case of recognition in clutter or with multiple stimuli in the receptive field [126]. Given a sequence, max-pooling functions will pick out the local max values from the input and shrink it along spatial and temporal dimensions to compose a more compact representation of the input sequence. To ensure the closure property [39], we further resize outputs calculated from max-pooling functions to an identical size as inputs using linear interpolation. In this way, the sizes of inputs and outputs of our max-pooling functions are the same.

In addition, the terminal set is also a significant component of genetic programming. For gesture recognition, we consider the following aspects of our task: (1) The terminal set must capture the holistic information of each gesture sequence; (2) During the evolution process, the evaluation of the fitness function must be efficient. In our implementation, we use the raw gesture sequence as the terminal set. In each tree-based genetic structure, a gesture sequence is located as the bottom leaf of the entire tree and connects with the higher function nodes directly. A representative GP tree is illustrated in Fig. 3.5.

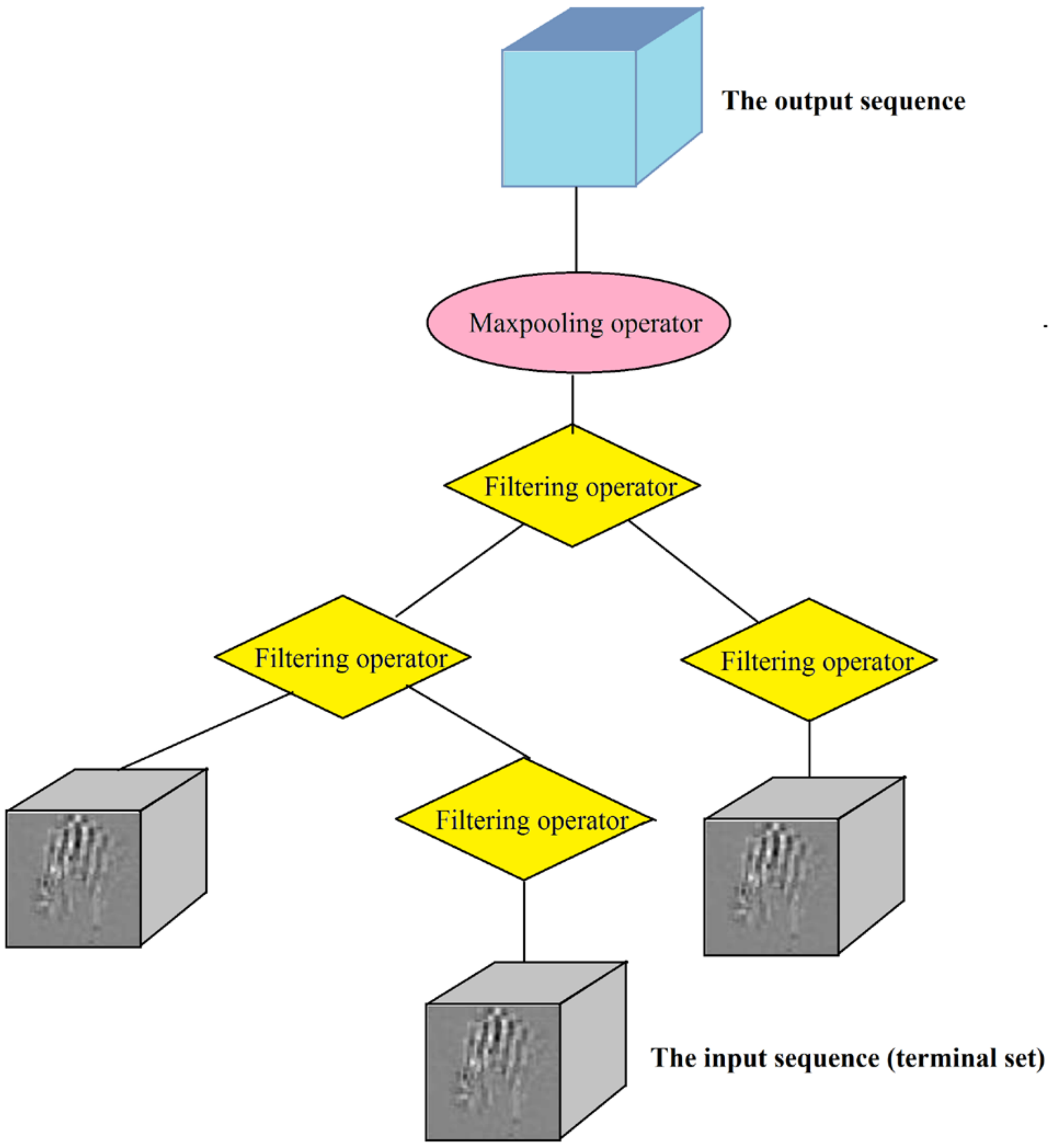


Fig. 3.5 Illustration of the tree structure of genetic programming

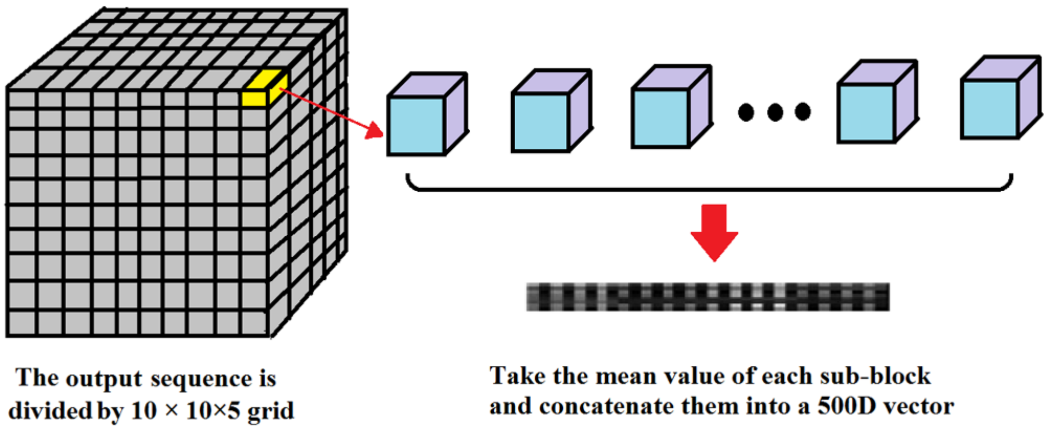

The output sequence is divided by 10 × 10×5 grid
Take the mean value of each sub-block and concatenate them into a 500D vector

Fig. 3.6 The procedure of spatio-temporal sequence representation

3.3.2 Fitness Function

The most important part of genetic programming is the fitness function which determines how well a program is able to solve the problem. To evaluate the candidate GP-evolved descriptors, we here adopt the classification accuracy calculated by a linear SVM classifier on the training set as the fitness function. In our GP architecture, for any of the input gesture sequences, we can obtain an output sequence with an identical size as the input due to the enclosure property.

We further divide the output by a $10 \times 10 \times 5$ grid. To make the final representation further invariant to shift and scaling, we take the mean values of each divided sub-block and concatenate them into a 500D vector as the input to the linear-SVM as shown in Fig. 3.6. To obtain a more reliable fitness evaluation, we adopt five-fold cross-validation for each new GP tree using SVM. We divide the GP training set randomly into five equally-sized parts and perform five repetitions of training the SVM on 4/5 of the set and testing on the remaining 1/5. The overall fitness E_r is the average of the five-fold cross-validation accuracies. The corresponding fitness function is defined as follows:

$$E_r = (1 - (\sum_{i=1}^{n} (SVM[acu_i])/n)) \times 100\% \tag{3.2}$$

where $SVM[acu_i]$ denotes the classification accuracy of fold i by the SVM, n indicates the total number of folds executed with cross-validation. Here n is equal to 5.

Table 3.2 Parameter settings for genetic programming running

Population Size	300
Generation Size	60
Crossover Rate	85%
Mutation Size	10%
Selection for Reproduction	'lexictour'
Survival Method	'totalelitism'
Stop Condition	$\leq 2\%$

3.4 Experiments and Results

In this section we describe the details of our GP implementation and the experimental results of our method.

3.4.1 GP Implementation

We evaluate our proposed method using 64-bit Matlab 2011a (with the genetic programming toolbox GPLAB [1]) on a server configured with a 6-core processor and 32G of RAM running the Linux OS. Some significant user-defined parameters for GP are listed in Table 6.5.

For GP evolution, a lexicographic parsimony pressure has been applied as the selection method in our running. Like the original selection method, a random number of individuals are chosen from the population and further the best of them is chosen. The only difference from the original selection is that, if multiple individuals are equally fit, the shortest one (the tree with the least number of nodes) is chosen as the best. Lexicographic parsimony pressure has shown effective control *bloat* [39] in different types of problems. In addition, we have adopted the 'totalelitism' scheme as the survival module in which all the individuals from both parents and children populations are ordered by fitness alone, regardless of being parents or children. This scheme has been demonstrated to lead to promising results in many applications. We also set the GP termination of 2%, which means if the value calculated by the fitness function is equal to or lower than 2%, our GP running will be stopped and return the best-so-far individual to users.

3.4.2 Datasets

We systematically evaluated our proposed method on two dynamic hand gesture datasets, namely, the Cambridge hand gesture dataset [127] and the Northwestern University hand

[1] http://gplab.sourceforge.net/download.html

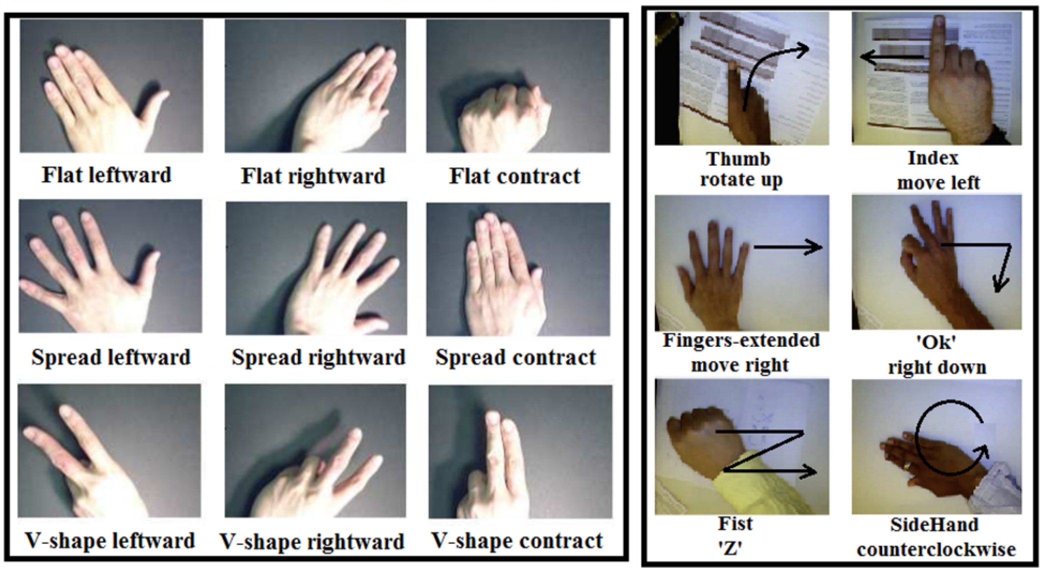
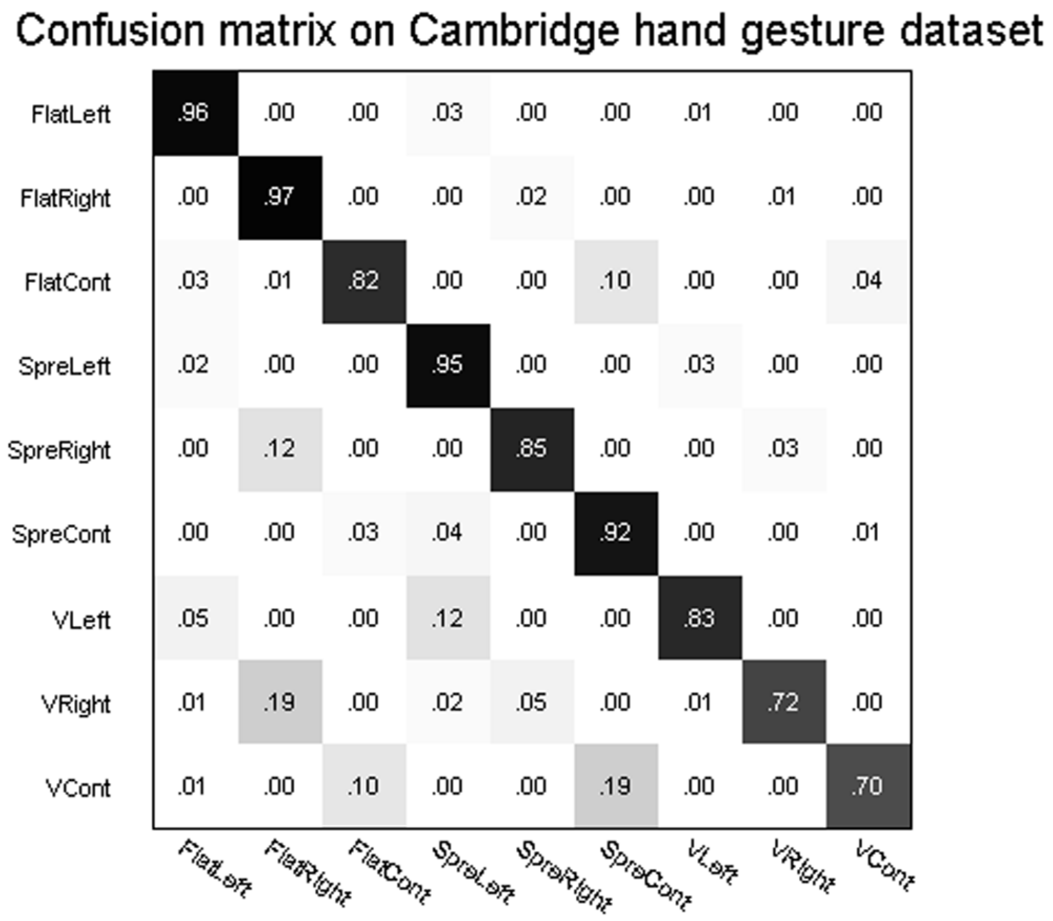


Fig. 3.7 Some example frames of two datasets. Images in the left black-box are from the Cambridge hand gesture dataset and images in the right black-box are from the Northwestern University hand gesture dataset.

gesture dataset [128]. Some example frames from these two datasets are visualized in Fig. 3.7.

Cambridge hand gesture dataset is a commonly used benchmark gesture dataset with 900 video clips of 9 hand gesture classes defined by 3 primitive hand shapes (*i.e.*, flat, spread, V-shape) and 3 primitive motions (*i.e.*, Leftward, Rightward, Contract). For each class, it includes 100 sequences captured with 5 different illuminations, 10 arbitrary motions and 2 subjects. Each sequence was recorded in front of a fixed camera having coarsely isolated gestures in spatial and temporal dimensions. All video sequences are further normalized into $200 \times 200 \times 50$ in our experiments by linear interpolation. Following the experimental setting in [127], the GP training is performed on the data acquired in the single plain illumination condition, while testing is done on the data acquired in the remaining four illuminations.

Northwestern University hand gesture dataset is a more diverse dataset which contains 10 categories of dynamic hand gestures in total: move right, move left, rotate up, rotate down, move downright, move right-down, clockwise circle, counterclockwise circle, "Z", and "cross". This dataset is performed by 15 subjects and each subject contributes 70 sequences of these ten categories with seven postures (*i.e.*, Fist, Fingers-extended, ’Ok’, Index, SideHand, SideIndex and Thumb). Our hand gesture recognition task is similar to [128] - we just focus on recognizing the 10 dynamic gestures. Therefore the samples within the same category in different hand postures are considered as one category, and each category


Confusion matrix on Cambridge hand gesture dataset

	FlatLeft	FlatRight	FlatCont	SpreLeft	SpreRight	SpreCont	VLeft	VRight	VCont
FlatLeft	.96	.00	.00	.03	.00	.00	.01	.00	.00
FlatRight	.00	.97	.00	.00	.02	.00	.00	.01	.00
FlatCont	.03	.01	.82	.00	.00	.10	.00	.00	.04
SpreLeft	.02	.00	.00	.95	.00	.00	.03	.00	.00
SpreRight	.00	.12	.00	.00	.85	.00	.00	.03	.00
SpreCont	.00	.00	.03	.04	.00	.92	.00	.00	.01
VLeft	.05	.00	.00	.12	.00	.00	.83	.00	.00
VRight	.01	.19	.00	.02	.05	.00	.01	.72	.00
VCont	.01	.00	.10	.00	.00	.19	.00	.00	.70

Fig. 3.8 The confusion matrix of classification results on the Cambridge hand gesture dataset.

accordingly has 105 samples. In our experiments, we first resize all the sequences into an identical size of $240 \times 240 \times \times 50$ and gesture sequences from the first 8 subjects are chosen to compose the GP training set and the rest of sequences are used as the testing set. We further evaluate the performance by adopting 'leave-one-out' cross validation on the remaining 7 subjects and consider the average accuracy as the final recognition result.

3.4.3 Results

For the Cambridge hand gesture dataset, Fig. 3.9 shows the LISP format of the GP-evolved best-performing descriptor which finally achieves an overall accuracy of 85% on the four testing sets with different illuminations using the linear SVM classifier. For comparison, we list the results published in the original work of the Cambridge hand gesture dataset. All these results were obtained using the same setting as ours. In addition, we also compare with

```
Output(Maxpooling5(SUB(ADD(ABS(Do
F(Gau1(Input))),GBO-45(ABSsub(Wavele
t(Gau2(DoF(Input))),GBO-90(Input)))),Lo
G2(Gau1(ABS(DoF(Wavelet(Input)))))))
```

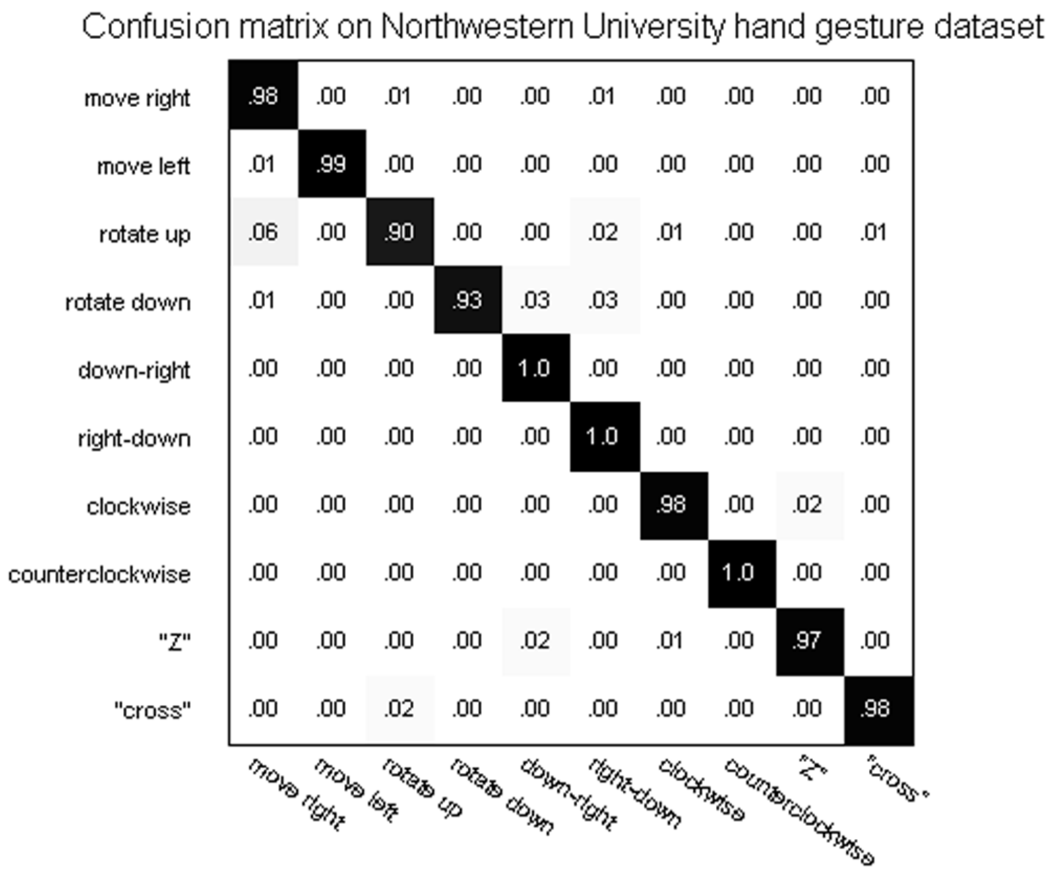
Fig. 3.9 The LISP format of the GP-generated descriptor on the Cambridge hand gesture dataset.

Table 3.3 Comparison of gesture recognition accuracies (%) on the Cambridge hand gesture dataset

Set Methods	Set1	Set2	Set3	Set4	Average accuracy
Our Method	**83**	**86**	**82**	**89**	**85**
Kim *et al.* [127]	81	81	78	86	82
Niebles *et al.* [16]	70	57	68	71	67
Wong and Cipolla [129]	-	-	-	-	44
HMHI [130]	79	79	83	81	81
HOG/HOF [20]	81	77	79	80	79
3D-HOG [131]	76	72	80	75	76
3D-SIFT [132]	79	74	69	77	75

prevalent hand-crafted 3D descriptors including HMHI, HOG/HOF, 3D-HOG and 3D-SIFT. Under the same experimental setting, we use the hierarchical motion history image (HMHI) as a holistic descriptor to extract the motion features for gesture recognition. For the other three 3D descriptors, we first divide gesture sequences with a $10 \times 10 \times 5$ grid, and describe each sub-block with one of the descriptors. Then, we concatenate the obtained features on all the sub-blocks into a vector as the final representation which is fed to the linear SVM for classification. All the relevant results are shown in Table 6.9. It is obvious that our proposed method significantly outperforms both the state-of-the-art techniques and popular hand-crafted features. A confusion matrix for the total testing sets is plotted in Fig. 3.8. We can see that our method can produce excellent recognition rates on gestures with *flat* and *spread* hand poses, however, the greatest confusion exists in *V-shape* hand pose which is hard to distinguish from the other gestures reliably.

The results on the Northwestern University hand gesture dataset are shown in Table 3.4 with the comparison to other methods . As expected, our GP-evolved descriptor achieves



Confusion matrix on Northwestern University hand gesture dataset

	move right	move left	rotate up	rotate down	down-right	right-down	clockwise	counterclockwise	"Z"	"cross"
move right	.98	.00	.01	.00	.00	.01	.00	.00	.00	.00
move left	.01	.99	.00	.00	.00	.00	.00	.00	.00	.00
rotate up	.06	.00	.90	.00	.00	.02	.01	.00	.00	.01
rotate down	.01	.00	.00	.93	.03	.03	.00	.00	.00	.00
down-right	.00	.00	.00	.00	1.0	.00	.00	.00	.00	.00
right-down	.00	.00	.00	.00	.00	1.0	.00	.00	.00	.00
clockwise	.00	.00	.00	.00	.00	.00	.98	.00	.02	.00
counterclockwise	.00	.00	.00	.00	.00	.00	.00	1.0	.00	.00
"Z"	.00	.00	.00	.00	.02	.00	.01	.00	.97	.00
"cross"	.00	.00	.02	.00	.00	.00	.00	.00	.00	.98

Fig. 3.10 The confusion matrix of classification results on the Northwestern University hand gesture dataset.

```
Maxpooling10(ABSsub(SUB(ADD
(LoG1(DoF(GBO-45(ABS(Gau1
(Input))))),Dof(Input)),Gau1(GBO-90
(Wavelet(Input)))),ABSsub(GBO-135
(LoG2(Gau1(DoF(Input)))),Med
(Input))))
```

Fig. 3.11 The LISP format of the GP-generated descriptor on the Northwestern University hand gesture dataset.

Table 3.4 Comparison of gesture recognition accuracies (%) on the Northwestern University hand gesture dataset.

Methods \ Accuracy	'leave-one-out' overall
Our Method	**96.1**
Shen *et al.* [128]	95.8
HOG/HOF [20]	86.4
HMHI [130]	85.2
3D-HOG [131]	82.5
3D-SIFT [132]	77.4

a classification accuracy rate of 96.1% on the testing set combining with linear SVM, since this gesture dataset is a relatively easy dataset with small intra-class variations and large inter-class variations. The result of [128] was obtained by re-implementing relevant experiments under the experimental setting as ours. Results of the popular 3D descriptors were produced in the same manner as the Cambridge dataset described above. From Table 3.4, we can observe that our method is comparable to Shen *et al.*'s and outperforms prevailing hand-crafted descriptors. Fig. 3.11 shows the LISP format of the generated GP-evolved descriptor. We also illustrate the confusion matrix of the recognition results on the Northwestern University hand gesture dataset in Fig. 3.10. We can see that our method can achieve 100% recognition rates on several gesture categories.

3.5 Summary

In this chapter, we have proposed a domain-independent method using genetic programming (GP) to generate machine-learned descriptors for dynamic gesture recognition. Our method addresses gesture recognition as an optimization problem, and allows a computer to automatically synthesize holistic descriptors from a pool of primitive sequence processing operators without any prior knowledge. We have evaluated our method on the Cambridge hand gesture dataset and the Northwestern University hand gesture dataset and achieved accuracies of 85% and 96.1%, respectively, with the obtained best-performing descriptors evolved by GP. In both datasets, the GP-generated descriptors significantly outperform prevailing hand-crafted features.

To make the GP learned descriptor more discriminative and robust for noisy and complex data, in the next chapter, we will develop a multi-objective genetic programming (MOGP), which can better resist the over-fitting for noisy data than original GP, for image classification.

Chapter 4

Feature Learning for Image Classification via Multi-objective Genetic Programming

4.1 Overview

Feature extraction is the first and most critical step in image classification. Most existing image classification methods use hand-crafted features, which are not adaptive for different image domains. In the last chapter, a genetic programming learning scheme has been used for automatically evolving the discriminative feature representations and achieved better results compared with hand-crafted features. To further control the overfitting caused by noisy data and outliers in standard genetic programming, a multi-objective genetic programming has been successfully proposed. In this chapter, we report what we believe to be the first approach using MOGP for generating (near-)optimal feature descriptors in image classification applications. Given a group of primitive operators and a set of terminals (*i.e.*, labelled training examples and random constants), MOGP evolves (hopefully) better-performing individuals in the next generation. Since MOGP selection depends on two fitness objectives, we use the *Pareto front* to represent a set of non-dominated 'rank-1' solutions which provide different trade-offs among the two objectives based on standard evaluation criteria. Furthermore, we test these evolved solutions on the evaluation set to select the (near-)optimal one with the lowest test error as the final feature descriptor. Typically, the features extracted by the MOGP system contain domain-specific information which makes the later image classification easier. It should be noted that our proposed method does not guarantee any mathematical optimum, while, in practice, it usually finds a 'good' solution to an NP-hard problem in an acceptable amount of computing time. A preliminary version of our approach has been applied to evolve holistic scene features for scene recognition [40].

To evaluate the generalizability of our approach, we apply the MOGP on four independent datasets: object dataset - Caltech-101 [133], scene dataset - MIT urban and Nature [75], face dataset - CMU PIE [134] and hand posture dataset - Jochen Triesch II [135]. For comparison, we also show that the proposed method is superior to some state-of-of-art hand-crafted and learning techniques.

The contributions of this work lie in the following aspects:

(1) We have successfully utilized multi-objective genetic programming (MOGP) to automatically design feature descriptors for image classification and achieved better performance compared with state-of-the-art methods.

(2) The proposed MOGP learned descriptors provide an effective way to simultaneously extract and fuse the R, G, B and gray scale information into one feature representation.

4.2 Literature Review

Much of previous work has explored the use of low-level image features for high-level scene (or object) classification. As this work is on holistic descriptors, we review related works on holistic feature representations. Commonly, we adopt either dense sampling with local descriptors or pure holistic descriptors for holistic representations. Popular choices for densely sampled feature descriptors include SIFT, HOG, P-HOG [37], LBP and shape context [38]. These features can work well for certain image types but may not be so descriptive for other types.

The other group of methods attempt to describe an image as a whole. In [75], a representation of the structure for real-world scenes termed 'Spatial Envelope' was proposed. This representation is related to the shape of the space and is meaningful to human observer-s. It can be described as a set of perceptual properties (naturalness, openness, roughness, ruggedness and expansion). The spatial envelope properties provide a holistic description of the scene where local object information is not taken into account. Song and Tao [125] performed scene classification adopting biologically inspired features (BIF), which simulate the human brain cortex by encoding orientation, intensity and color information of the scene and can tolerate shifting, translation, and scaling. Beyond that, the work in [136] computes the feature descriptor by adopting a bank of Gabor filters tuned to 8 orientations at 4 different scales. The square output of each filter is then averaged on a 4×4 grid. In addition, the methods in [137, 138] have also applied global features to different tasks of image classification. Either discriminative or generative techniques used in the above methods are always based on *hand-crafted* features.

Recently, deep learning methodologies have been utilized to obtain machine-learned features for image classification. They can always effectively control training *overfitting* and overcome the limitations of back propagation. Among them, deep belief network (DB-N) [139] shows its capacity to automatically learn multiple layers of non-linear features from images and videos. Another similar learning method is called convolutional neural network (CNN) [140]. In these architectures, the features are extracted at different levels (*e.g.*, edges, object parts, and objects). These methods aim to extract general-purpose features for any images rather than learning domain-adaptive feature descriptors particularly for certain tasks such as classification.

With the development of artificial intelligence, evolutionary methods simulate biological evolution to automatically generate solutions for user-defined tasks. GP, as one of the most outstanding evolutionary methods (see Algorithm 1), has been widely used in the computer vision domain. Poli [121] has applied GP for image analysis which automatically selects efficient optimal filters for segmentation of the human brain in medical images. Torres *et al.* [120] have used GP for finding an optimal combination of the similarity functions to design a method for image retrieval. Specifically, GP has also been adopted to automatically extract features. Koza [141] has evolved character detectors using GP, while Tackett [142] has evolved a symbolic expression for image classification based on image features. Bot [143] has also applied GP to evolve new features in the decision space by using the k-nearest neighbour (KNN) classifier. Sherrah *et al.* [144] have proposed a system which automatically extracts features for a range of classification problems. In their method, GP is used to allow useful features to emerge from the raw data, simulating the innovative work of the human designer. The proposed method has successfully improved the classification performance on real-world problems. Atkins *et al.* [145] have explored the domain-independent image classification application based on GP, which is capable of evolving a single program that can both extract useful features and use those features to classify an image. In addition, Guo *et al.* [146] have used GP to perform automatic feature extraction from the original feature database with the aim of improving the discriminatory performance of a classifier and reducing the input feature dimensionality at the same time.

The previous usages of GP are mostly based on a single objective. However, in real-world design problems, there are often multiple competing objectives for target tasks. For instance, in a GP evolved classification system, it is obvious that the design objective is to minimize the classification error rate. In addition to the specific measure being taken for evaluation, the tree complexity should also be considered to control the *overfitting* caused by *bloat*. In fact, the experimental results show the multi-objective genetic programming

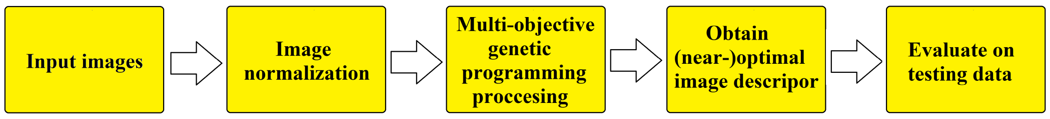

Input images
Image normalization
Multi-objective genetic programming proccesing
Obtain (near-)optimal image descripor
Evaluate on testing data

Fig. 4.1 The outline of our method.

(MOGP) can successfully restrict the over-parameterization problems during the program evolving. Zhang and Rockett [147] have proposed a genetic, optimal feature extraction method through the MOGP. Their method has been applied to edge detection problems for image processing and achieved a comparable result with the Canny edge detector. Similarly, Watchareeruetai *et al.* [148] have also adopted MOGP for automatic construction of feature extractor. Liddle and his colleagues [149] have proposed a MOGP approach for the task of providing a decision-maker with a diverse set of alternative object detection programs that balance between high detection rate and low false-alarm rate. In addition, an MOGP approach has been developed by Olague and Trujillo [150] for automated synthesis of operators that detect interest points. In their work, the MOGP provides the appropriate framework for the automatic design of image operators that achieve interesting trade-offs between relevant performances criteria that are meaningful for a variety of vision tasks. Inspired by the above successful applications, in this chapter, we have proposed a multi-layer MOGP-based learning system that evolves a single individual program mimicking the visual cortex and can simultaneously extract and fuse features from different color channels for image classification.

4.3 Methodology

In this chapter, we design a multi-layer feature extraction system by successfully applying multi-objective genetic programming (MOGP) for image classification. A group of primitive 2D operators is adopted in our architecture to automatically evolve a single individual program that is allowed to extract discriminative features from the 2D image data directly. The proposed MOGP-based method is regarded as a generalized methodology for image classification rather than just as a specific one for a certain task. We evaluate our method on different types of datasets and it achieves outstanding results, which outperform previously published techniques. The outline of our method is illustrated in Fig. 4.1.

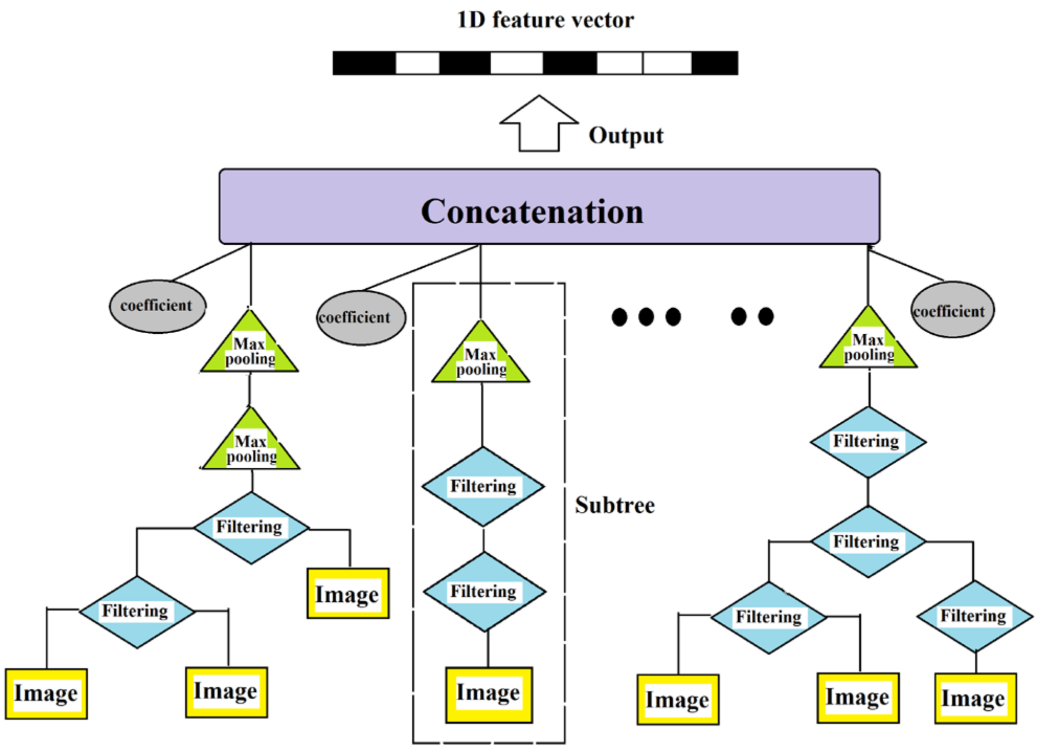


Fig. 4.2 The grammatical structure of an individual.

4.3.1 Program Structure Design

In our architecture, the individual MOGP program can be divided into four layers, *i.e.*, the data input (bottom layer), the filtering (middle 1 layer), max-pooling (middle 2 layer) and concatenation (top layer) .The order of these layers in our structure is always fixed, which is consistent with the physical structure of the human visual cortex [151]. Both data input and concatenation are one-depth layer, while, the depth of the filtering and max-pooling layers is dynamic and unrestricted according to the evolve procedure of MOGP. Fig. 4.2 illustrates the proposed structure of an example program.

The data input layer is located at the bottom of the whole structure and serves as the entry of the input 2D images which are all normalized into the same size by using bilinear interpolation. As a result, all the images are fed to the MOGP program with the identical size. The filtering layer is above the data input layer, and all the data are operated through the nodes among the filtering layer and the outputs of these nodes are still 2D images, which have the same size as the inputs. The third layer is max-pooling, where all the data are processed by the max-pooling technique (see in Fig. 4.4) which is considered as a key mechanism for robust response in the case of recognition in the clutter scene or with multiple stimuli in the receptive field [126]. On the top of our proposed structure is the concatenation layer,

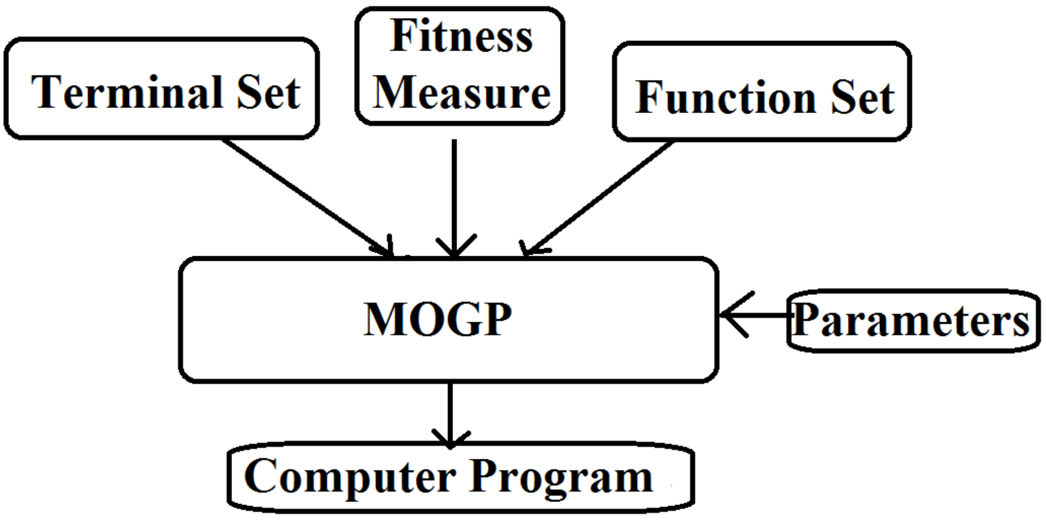


Fig. 4.3 The Terminal Set, Fitness Measure and Function Set (shown at the top of the figure) are the human-supplied inputs to the MOGP system. The computer program (shown at the bottom) is the output of the MOGP system.

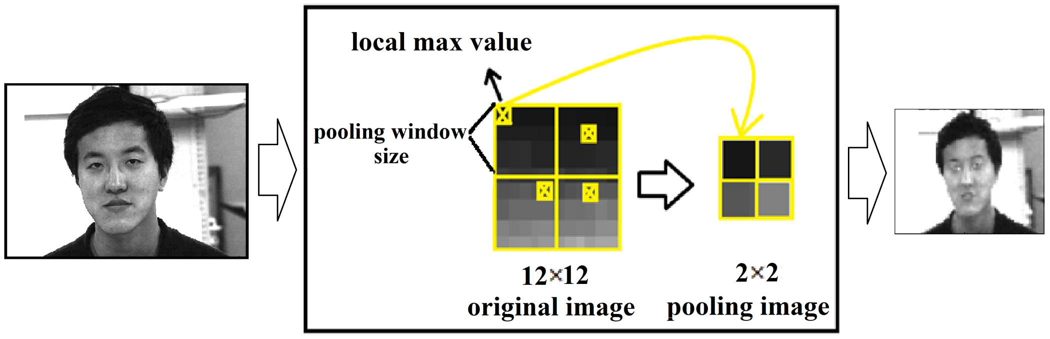


Fig. 4.4 The illustration of the max-pooling mechanism.

where all the data acquired from the adjacent layer are first ’flattened’ from 2D images to 1D vectors which are then fed to the concatenation operators. It fuses the features computed from the below sub-trees of MOGP and constructs a more informative vector as the final extracted feature descriptor of the input image.

The rest of this section describes the required terminal set, function set and fitness function, which are the most significant preparatory steps defined by users in our MOGP system as shown in Fig. 4.3

4.3.2 Terminal Set

With the inspiration from the human visual cognition system, we attempt to mimic the cortex of human brain to distinguish different scenes, objects based on the appearance features such

Table 4.1 Statement of terminal nodes

Terminal	Type	Description
I_r	Image	Red component of the RGB-image
I_g	Image	Green component of the RGB-image
I_b	Image	Blue component of the RGB-image
I_{gray}	Image	Gray scale image calculated from the RGB-image
C	double	Returns a random double between 0 and 5

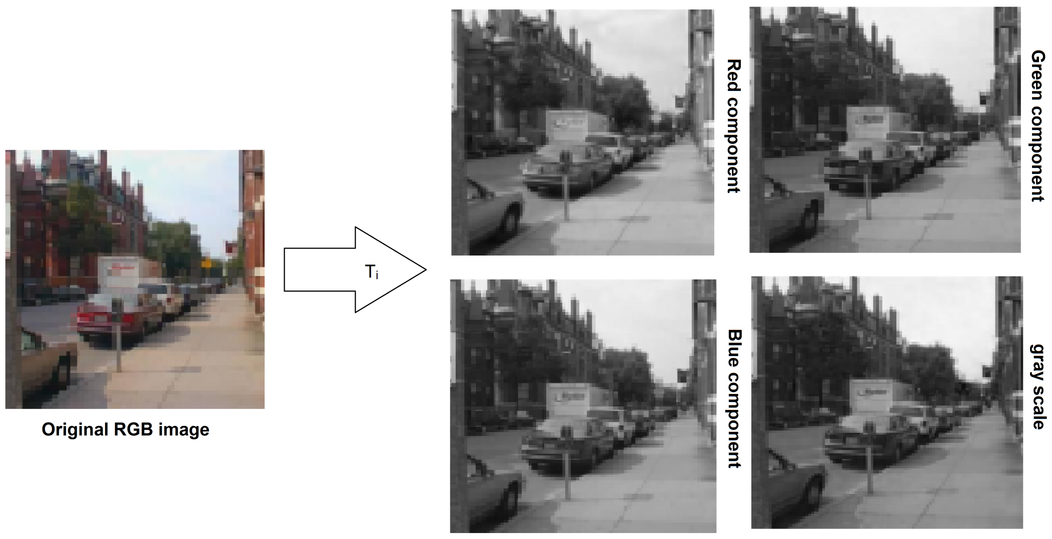


Fig. 4.5 An example of the components of an input image.

as color, intensity or contrast information. For most of the time, our brain is more sensitive to color-related information when recognizing the corresponding targets. As the relevant mechanism in human brain is much more complex than how we think about it, we simply simulate the basic principle of the human recognition procedure to solve practical tasks. In this way, the terminal set for our proposed methodology is expected to be more specific than a general GP definition and highly depends on the nature of specific tasks. Here, we consider proposing a system to extract features from the color components (*i.e.*, Red, Green, and Blue) and intensity information (gray scale) of RGB-images. Furthermore, the input of the bottom layer is image dependent, which means that each image I_i from the learning set has a corresponding T_i defined by: $T_i = \{I_r, I_g, I_b$ and $I_{gray}\}$ shown in Fig. 4.5

Additionally, to make the final feature fusion more flexible, we also define a group of double-precision number coefficients C in our terminal set for constructing a weighted linear concatenation of sub-MOGP-trees in the top layer. In the proposed architecture, the optimal weight coefficients for the sub-trees are randomly distributed and selected by MOGP. C is

Table 4.2 Filtering Layer functions

Operator Name	Input	Function Description	Operator Type
Gau1	1 Image	Gaussian smooth filter with $\sigma = 1$	Filter
Gau2	1 Image	Gaussian smooth filter with $\sigma = 2$	Filter
GauX	1 Image	The derivative along X axis of Gaussian filter	Filter
GauY	1 Image	The derivative along Y axis of Gaussian filter	Filter
LoG1	1 Image	Laplacian of Gaussian filter with $\sigma = 1$	Filter
LoG2	1 Image	Laplacian of Gaussian filter with $\sigma = 2$	Filter
Lap	1 Image	Laplacian filter	Filter
GBO-0	1 Image	Multi-scale-max Gabor filter with orientation of 0 degree	Filter
GBO-45	1 Image	Multi-scale-max Gabor filter with orientation of 45 degree	Filter
GBO-90	1 Image	Multi-scale-max Gabor filter with orientation of 90 degree	Filter
GBO-135	1 Image	Multi-scale-max Gabor filter with orientation of 135 degree	Filter
Aver	1 Image	Averaging filter with 5×5 sampling window	Filter
Med	1 Image	Median filter with 5×5 sampling window	Filter
EHIS	1 Image	Histogram equalization	Enhancement
ABS	1 Image	Take the absolute value pixel by pixel	Arithmetic
ADD	2 Images	Add two input images pixel by pixel	Arithmetic
SUB	2 Images	Subtract two input images pixel by pixel	Arithmetic
ABSsub	2 Images	Absolute subtraction of two input images pixel by pixel	Arithmetic
MULTI	2 Images	Multiply two input images pixel by pixel	Arithmetic
DIV	2 Images	Protected-division: divide two input images pixel by pixel and return 0 if divisor is 0	Arithmetic
SQR	1 Image	Square the input image pixel by pixel	Arithmetic
SQRT	1 Image	Extract a square root pixel by pixel	Arithmetic
LOG2	1 Image	Take the logarithm with 2 at the bottom for input image	Arithmetic
TIME-0.5	1 Image	Multiply 0.5 to the input image pixel by pixel	Arithmetic

defined by: $C = coefficient \in (0,5)$. Therefore, the complete terminal set is $T_i \cup C$. The data types used for the terminal of program nodes are listed in Table 4.1.

4.3.3 Function Set

A key concept for MOGP is the function set (*i.e.*, internal nodes of the tree) which is the alphabet of the programs and typically driven by the nature of the problem domain. Commonly, the choice of functions is based on the following principles:

(1)The set must contain functions which can extract meaningful information.

(2)To minimize the total running time of MOGP, all the operators in the function set need to be relatively simple and efficient.

Functions for Filtering Layer

The filtering layer is the most significant part of our MOGP procedure, extracting the effective features from the input data. We construct the filtering layer by using 19 unary functions and 5 binary functions, shown in Table 4.2. All the filtering functions listed here operate on the images which represent in the form of a pixel matrix.

In the proposed MOGP architecture, we adopt Gaussian filter series, Garbor filter series and some basic arithmetic functions. Since Gaussian and Gaussian derivative filters are

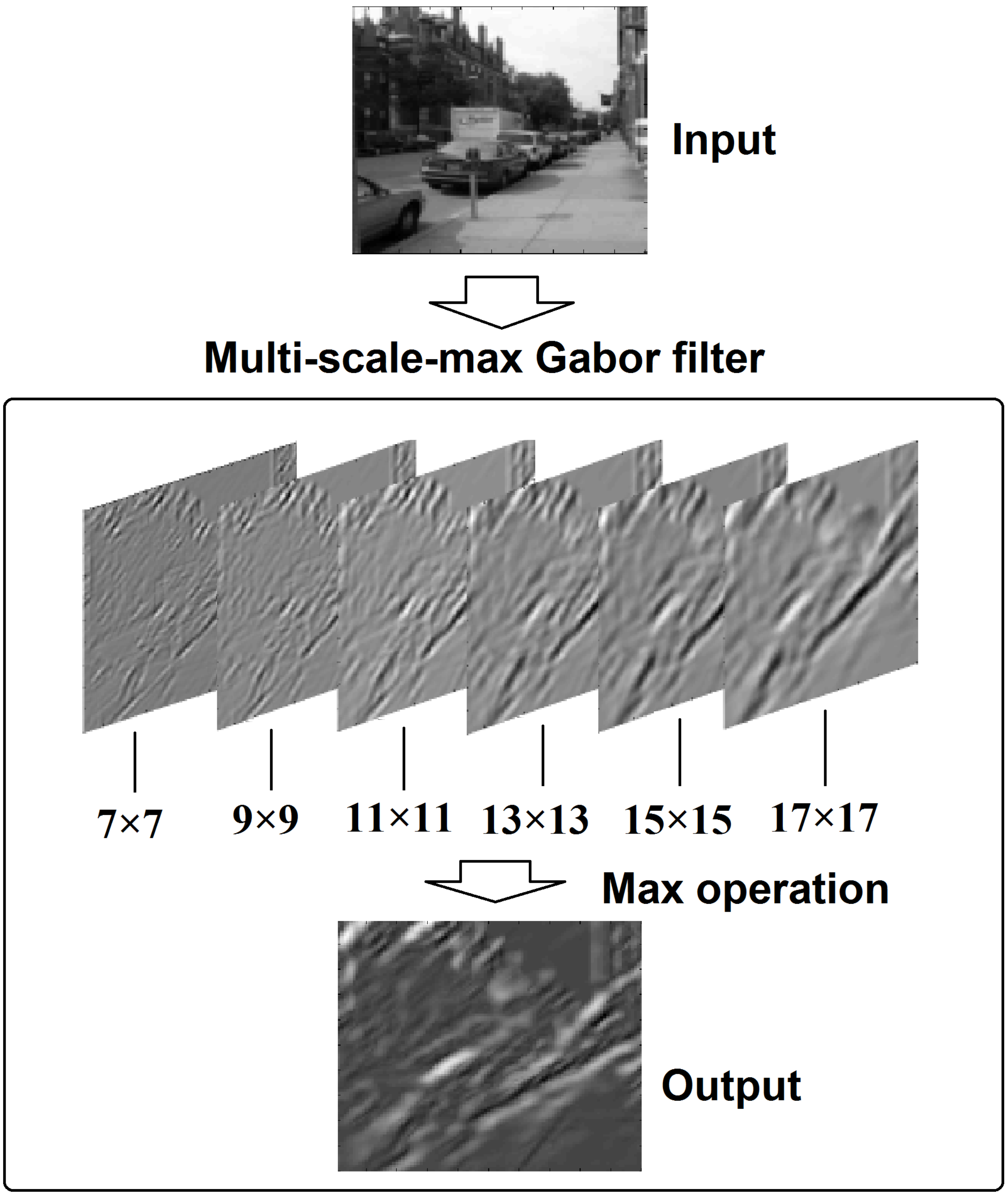


Fig. 4.6 The outline of multi-scale-max Gabor feature extraction.

Table 4.3 Max-pooling Layer functions

Operator Name	Input	Function Description	Operator Type
Maxpooling2	1 Image	2D max-pooling with pooling window size 2×2 on the input image	Filter
Maxpooling4	1 Image	2D max-pooling with pooling window size 4×4 on the input image	Filter
Maxpooling6	1 Image	2D max-pooling with pooling window size 6×2 on the input image	Filter
Maxpooling8	1 Image	2D max-pooling with pooling window size 8×8 on the input image	Filter
Maxpooling10	1 Image	2D max-pooling with pooling window size 10×10 on the input image	Filter

widely used to reduce the noise and smooth input images, we adopt them into our function set to achieve denoising and extract the meaningful contour information. 2D Laplacian filters are used for separating signals into different spectral sub-bands. Gabor filters are regarded as an effective method to obtain the orientation information. To mimic the biological mechanism of the human visual cortex, we follow Riesenhuber and Poggio's work [126] to define our GBO-θ_s (*i.e.*, u=0, 45, 90 and 135) filter by two steps: first, convolving the input image with six different scale Gabor filters (7×7, 9×9, 11×11, 13×13, 15×15 and 17×17) with a certain orientation θ_s; Second, using max operation to pick the maximum value across all six convolved images with the filter scales in each orientation. Fig. 4.6 illustrates our multi-scale-max Gabor filter. The pooling equation among different scales is defined below:

$$\begin{aligned} GBO\text{-}\theta_s = \max_{(x,y)}[&I_{7\times7}(x,y,\theta_s), I_{9\times9}(x,y,\theta_s), \\ &..., I_{15\times15}(x,y,\theta_s), I_{17\times17}(x,y,\theta_s)] \end{aligned} \tag{4.1}$$

where, GBO-θ_s is the output of a multi-scale-max Gabor filter and $I_{i\times i}(x,y,\theta_s)$ denotes the convolved images with the scale $i \times i$ and the orientation θ_s.

In addition, basic arithmetic functions are chosen based on the desire to allow the MOGP process computing features from images more naturally. The binary arithmetic functions also play a significant role to stretch the MOGP tree by splits and make the individual programs more variable and diversified as such qualities significantly influence final results.

To ensure operator closure [39], we have only used functions which map one or two 2D images to a single 2D image with the identical size (*i.e.*, the input and the output of each function node have the same size). In this way, a MOGP tree can be an unrestricted composition of function nodes but still always produce a semantically legal structure.

Functions for Max-pooling Layer

The max-pooling function set is listed in Table 4.3. All the functions in this set are performed over local neighbourhoods with windows varying from 2×2 to 10×10 with an

Table 4.4 Concatenation Layer functions

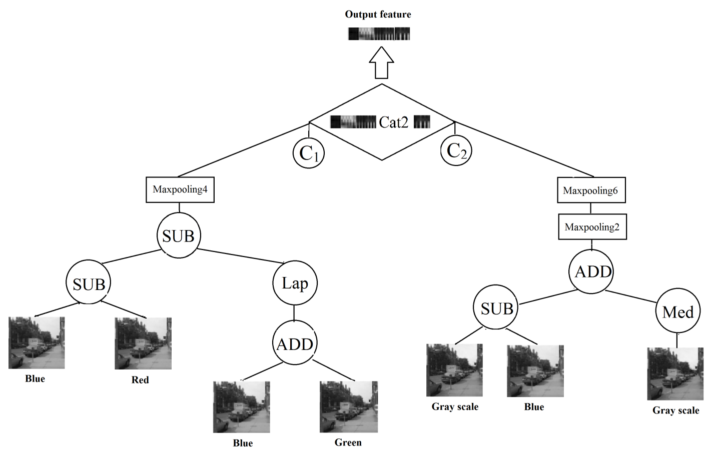
Operator Name	Input	Function Description	Operator Type
Cat1	1 Image and 1 coefficient	'Flatten' the 2D input image to a 1D vector with coefficient	Fusion
Cat2	2 Images and 2 coefficients	'Flatten' the two 2D input images to 1D vectors and concatenate these two vectors into a long vector with coefficients	Fusion
Cat3	3 Images and 3 coefficients	'Flatten' the three 2D input images to 1D vectors and concatenate these three vectors into a long vector with coefficients	Fusion
Cat4	4 Images and 4 coefficients	'Flatten' the four 2D input images to 1D vectors and concatenate these four vectors into a long vector with coefficients	Fusion

incremental step of 2 pixels both horizontally and vertically. This max-like feature selection operation is a key mechanism for object recognition in the human brain cortex and provides a robust response in the case of recognition with clutter or multiple stimuli in the receptive field. This kind of mechanism successfully achieves invariance to image-plane transforms such as translation and scaling. One noteworthy point here is that the output of a max-pooling filter is inevitably shrunk along the spatial dimensions relative to the input. To ensure the closure property mentioned above, we further resize each output calculated from max-pooling filters to an identical size with the inputs using linear interpolation. In this way, the size of inputs and outputs of our max-pooling filters are totally the same.

Functions for Concatenation Layer

After the filtering and the max-pooling layers, we have successfully extracted the corresponding features from the original input images through the tree-based MOGP programs. By first converting 2D images to 1D vectors, we further concatenate these 1D vectors calculated from different sub-trees into a weighted linear concatenation vector with coefficients randomly selected from the terminal set as shown in Fig. 4.7 , where C_1 and C_2 denote the corresponding coefficients automatically obtained by MOGP and '$\bigoplus$' means concatenation. To cope with different numbers of sub-trees, in our architecture, we include four different concatenation functions with different numbers of inputs. The relevant functions are illustrated in Table 4.4. Different from the other three functions in this set, for Cat1, there exists no concatenation procedure, but only 'flatten' the 2D input image to a 1D vector and complete multiplication by the corresponding coefficient for each element in the vector. We also restrict the concatenation layer with only one depth. Since this layer is the last layer in our program structure, the output we obtain from this layer is the final feature descriptor extracted from the input image.

Each function in the function set is regarded as a tree node in evolved programs, which connects the outputs of lower level functions or inputs. Note that, in our proposed MOGP architecture, not all the functions listed in the function set have to be used in a given structure



2 Sub-tree Concatenation Model:

$$y=C_1[\text{Maxpooling6}(\text{Maxpooling2}(\text{ADD}(\text{SUB}(I_i, I_b), \text{Med}(I_i))))] \oplus C_2[\text{Maxpooling4}(\text{SUB}(\text{SUB}(I_r, I_b), \text{Lap}(\text{ADD}(I_g, I_b))))]$$

Fig. 4.7 An illustration of Concatenation Layer.

and the same function can be used more than once. The topology of our MOGP structure is essentially unrestricted. Besides, functions in the function set are also highly related to our problem domain. For this work, we aim to construct novel discriminative feature descriptors for image classification. So, what we choose in the function set are effective filters for feature extraction, *i.e.*, Gaussian filters, Gabor filters, Laplacian filters, *etc*. We expect the whole learned architecture is consistent with the physical structure of the human visual cortex.

4.3.4 Fitness Function

The basis of evolutionary methods is to maximize the performance of individual solutions as gauged by some appropriate fitness functions. In our approach, we apply multi-objective genetic programming (MOGP) to evolve each individual program generation by generation and finally select the best-preformed individual as the optimal solution.

Two Defined Fitness Objectives

Classification error rate: To evaluate the candidate MOGP-evolved feature descriptor, we estimate their classification error rate E_r using a linear support-vector-machine (SVM) on the learning set. We take the output of the MOGP tree and first adopt principal component analysis (PCA) to reduce the high dimensional vector into a low dimensional one which comprises the input of the SVM. To obtain a more reliable fitness evaluation, for each candidate MOGP tree we estimate the classification error rate with the SVM by using n-fold cross-validation (we use $n = 10$ in all of our experiments below). We train the SVM on n-1/n-ths of the set and test on the remaining. The overall fitness of the candidate MOGP tree is taken as the average of the n SVM test-fold error rates. As an alternative way, other classifiers, such as K nearest neighbor (KNN), can be also used for fitness evaluation. The related fitness function is defined as follows:

$$E_r = (1 - (\sum_{i=1}^{n} (SVM[acu_i])/n)) \times 100\% \tag{4.2}$$

where, $SVM[acu_i]$ denotes the classification accuracy of fold i by the SVM, n indicates the total number of folds executed with cross-validation. Here n is equal to 10.

Tree complexity measurement: We usually measure solutions' complexity by counting the node number of the tree structure. Unless inhibited from doing so, trees in GP have a tendency to grow without limit, which is termed *bloat*. As an undesirable phenomenon, *Bloat* may cause poor generalizability, a manifestation of the well-known *overfitting* effect in machine learning. In addition, bloated trees are also time-consuming for evaluation, which slows the evolution and heavily influences the efficiency of program running, especially for on-line GP learning. According to [17], minimizing the tree node count in a multi-objective setting can effectively suppress tree *bloat*.

Tree Optimization

For the proposed multi-objective genetic programming (MOGP) method, each individual is evaluated against these two fitness objectives, each of which we wish to simultaneously minimize. In fact, what we can obtain is a set of solutions which are all listed as the 'rank-1' according to the *Pareto front* shown in Fig. 4.8. Members on the *Pareto front* dominate the other candidates but are equivalent to each other. We further test the set of 'rank-1' solutions on an independent evaluation set to select the (near-)optimal one with the best test error (average classification error rate) by adopting the same 10-fold cross-validation procedure mentioned above. In this way, we obtain the best-preformed solution through MOGP as the final feature descriptor for image classification.

4.4 Experiments and results

In this section, we describe the details of our GP implementation and the relevant experimental results we obtain by our approach.

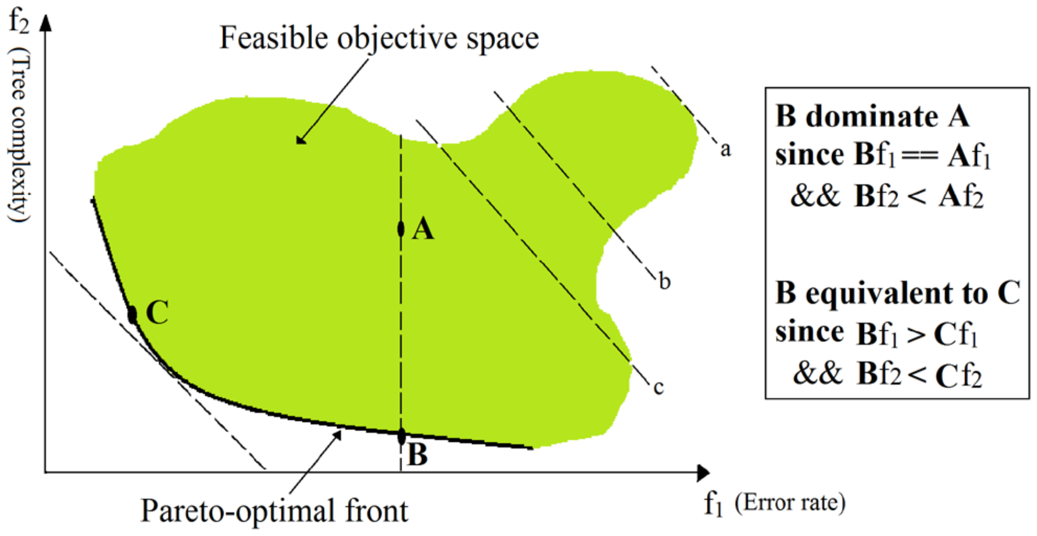

f_2 (Tree complexity)
Feasible objective space
a
b
c
A
B
C
Pareto-optimal front
f_1 (Error rate)
B dominate A since $Bf_1 == Af_1$ && $Bf_2 < Af_2$
B equivalent to C since $Bf_1 > Cf_1$ && $Bf_2 < Cf_2$

Fig. 4.8 A simple illustration of the Error-Complexity Pareto front. Given a set of objectives, a solution is said to Pareto dominate another if the first is not inferior to the second in all objectives, and, additionally, there is at least one objective where it is better. This notion can lead to a partial order, where there is no longer a strict linear ordering of solutions. Here, Point B significantly dominates Point A (A and B have the same error rate, while B has less tree complexity than A), but is equal to point C. The final 'rank-1' solutions are a set of feature descriptors which dominate the other candidates but are equivalent to each other according to our multi-objectives. In this figure, we consider that all the solutions (*e.g.*, Point B and Point C) existing on the Pareto front compose the 'rank-1' solution set.

4.4.1 GP Implementation

We evaluate the proposed method adopting Matlab 2011a (with the genetic programming toolbox GPLAB [1]) on a server with six processor cores and 32 GB of RAM running the Linux OS. To successfully operate the MOGP algorithm, some significant parameters are defined below:

Population and Generation Size: According to some previous relevant experiments, the larger population we define in GP running, the better solution we can potentially obtain. In this case, considering the high computational cost, we set a population size of 200 individuals with the initial population generated with the ramped half-and-half method [39]. The number of generation is defined as 70.

Genetic Operators: We use both cross over and mutation [39] as our genetic operators and fix their probabilities during the whole GP run at 90% and 10%, respectively.

Selection for Representation: The selection method we apply in GP is tournament which chooses each parent by randomly drawing a number of individuals from the population and selecting only the best of them.

Survival Method: We adopt the 'totalelitism' scheme for MOGP running. In this scheme, all the individuals from both parents and children populations are ordered by fitness alone, regardless of being parents or children. Consequently, the best individuals can be kept and inherited generation by generation. This scheme has been demonstrated leading to promising results in many applications.

Stop Condition: We set our MPGP termination as 0.5% of the error rate. If the classification error rate computed through the fitness function is equal or lower than 0.5%, our MOGP running will be stopped and return the best-so-far individual.

4.4.2 Datasets

To demonstrate the effectiveness, our proposed method is systematically evaluated on four datasets of different genres, *i.e.*, object dataset: Caltech-101, scene dataset: MIT urban and Nature, face dataset: CMU PIE and hand posture dataset: Jochen Triesch II. Note that in these four datasets, all the images are RGB color images. Some image examples from these four datasets are shown in Fig. 4.9. In the remaining part of this section, we will show the data structure of our methodology and the details of all of the datasets we use.

[1] http://gplab.sourceforge.net/download.html, A genetic programming Toolbox for MATLAB

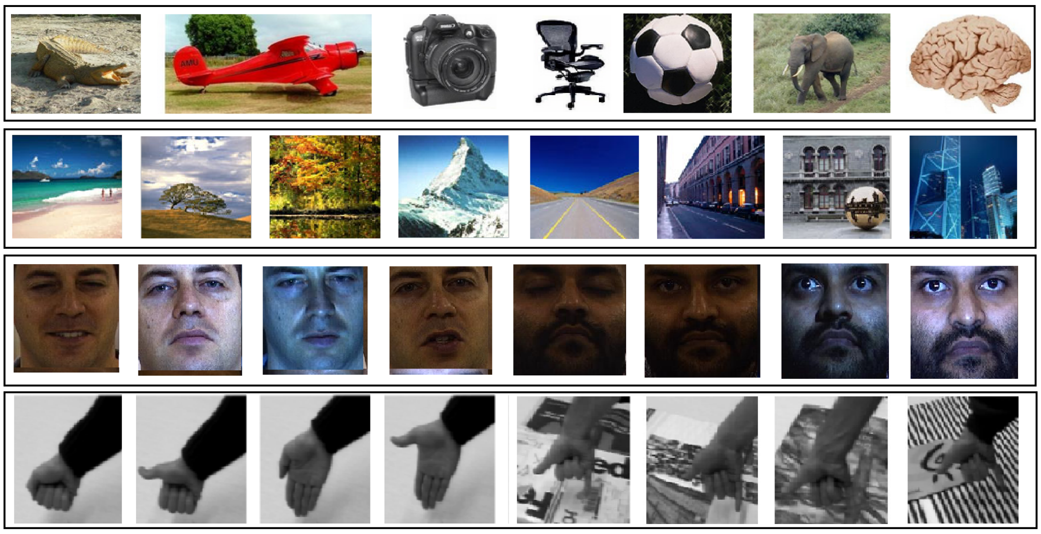


Fig. 4.9 Some image examples of four datasets. Images in the top black-box are from the Caltech-101 dataset, images in the second black-box are from the MIT urban and Nature scene dataset, images in the third black-box are from the CMU PIE face dataset and images at the bottom black-box are from the Jochen Triesch II hand posture dataset.

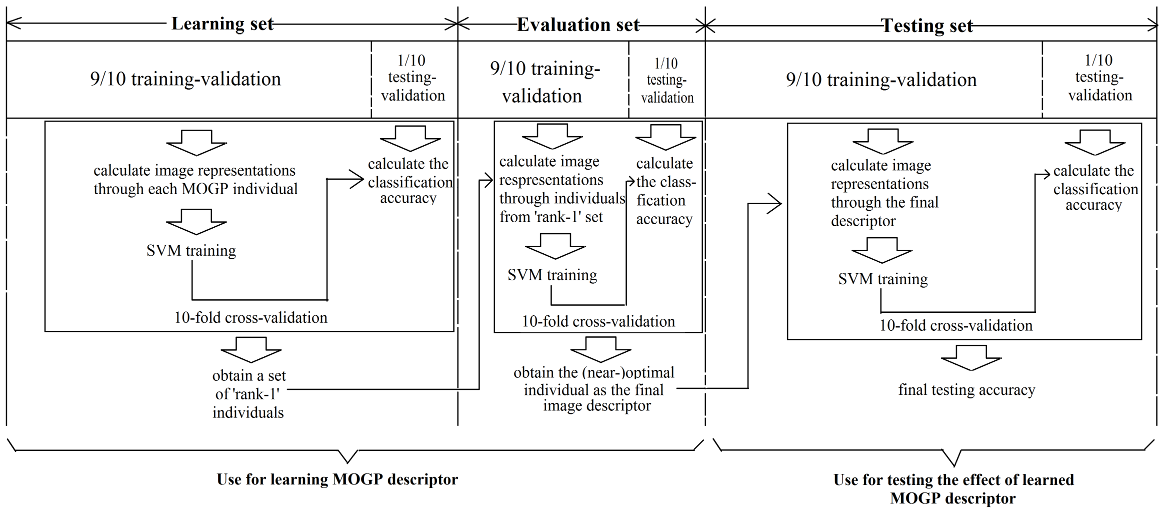

Learning set
Evaluation set
Testing set
9/10 training-validation
1/10 testing-validation
9/10 training-validation
1/10 testing-validation
9/10 training-validation
1/10 testing-validation
calculate image representations through each MOGP individual
calculate the classification accuracy
SVM training
10-fold cross-validation
obtain a set of 'rank-1' individuals
calculate image respresentations through individuals from 'rank-1' set
calculate the classification accuracy
SVM training
10-fold cross-validation
obtain the (near-)optimal individual as the final image descriptor
calculate image representations through the final descriptor
calculate the classification accuracy
SVM training
10-fold cross-validation
final testing accuracy
Use for learning MOGP descriptor
Use for testing the effect of learned MOGP descriptor

Fig. 4.10 The division of a dataset.

Experimental Data Division

As mentioned before, for each dataset, we first divide it into three parts, *i.e.*, the learning set, the evaluation set and the testing set. The details of the data division are shown in Fig. 4.10.

In our experiments, to obtain a more reliable fitness evaluation, each new MOGP individual is estimated by the average classification error rate with the SVM using ten-fold cross-validation. We divide the *learning set* randomly into ten equal parts and perform ten repetitions of training the SVM on 9/10-ths of the set and test on the remaining tenth. After MOGP evolution, we obtain a set of the 'rank-1' solutions as the potential candidates.

To select the best-performed solution, we further carry out the same procedure of ten-fold cross-validation to test the classification error on the *evaluation set*. In this way, we select one solution with the best testing error (lowest average error rate) as the (near-)optimal feature descriptor for final testing.

For the *testing set*, all images are first represented by the selected feature descriptor. We further divide the set into ten subsets with the identical size and in every loop we train the SVM using 9/10 of the testing set and test on the remaining. This procedure is repeated ten times and the overall classification result on a certain dataset is calculated as the average of the ten SVM test-fold accuracies.

Dataset Description

The first dataset is the **Caltech-101**, a standard dataset for image classification. It has 101 classes (animals, furniture, vehicles, flowers, face, *etc.*) with high intra-class appearance and shape variability. Due to the limitation of the computational resource, we only select the first 20 images from each category as the learning data, the following 15 images from each category as the evaluation data and the rest of images from each category as the testing data. We further normalize all of them into an equal size of 100×100 pixels using linear interpolation. In this way, 3030 images construct the learning set, 1515 images compose the evaluation set and the remaining 5800 images make up the testing set.

The second dataset is **MIT urban and nature scene**. This dataset is composed of 2688 color images with eight categories including 'coast&beach', 'highway', 'open country', 'tall building', 'forest', 'street', 'mountain' and 'city center'. In the pre-processing stage, each image in the dataset is first normalized into the size of 125×125 pixels by linear interpolation and then the first 100 images from each category are selected as the learning set, the following 50 images from each category form the evaluation set and the remaining images construct the testing set.

The third dataset is the **CMU PIE** face dataset which contains 41,368 images from 68

```
Cat4(Maxpooling6(Gau2(Lap(SQRT(LoG1(ABSsub(A
DD(GBO-90(Ib),ABS(GauX(LoG2(Time-0.5(Igray))))),
GauY(Ir))))))),0.75,Maxpooling4( Maxpooling2(ABSsu
b(EHIS(GEO-90(Igray)),ADD(GBO-0(Ig),GBO-45(Gau
1(Ig)))))),0.26,Maxpooling6(Gau2(LoG2(GBO-135(AB
S(GauY(Igray)))))),1.00,Maxpooling4(GauX(SUB(Lap(
Aver(GauY(Igray))),Ir))),0.84)
```

Fig. 4.11 The LISP format of the (near-)optical feature descriptor generated through MOGP on the Caltech-101 dataset.

subjects (people). All the face images are captured by 13 synchronized cameras and 21 flashed under 13 different poses, 43 different illumination conditions and 4 different facial expressions. We select all the images with five near frontal poses (*i.e.*, No. 05, No. 07, No. 09, No. 27, No. 29) under different illumination conditions and facial expressions. In this way, about 170 images from each subject are obtained to compose our dataset. We further pre-process all the images by employing the Viola-Jones face detector [152] to subtract the background and then normalize them into the same size of 64×64 pixels through linear interpolation. In our experiments, the first 80 images are chosen from each subject to compose the learning set, the following 40 images from each subject construct the evaluation set and the rest of the images are used as the testing set.

The last dataset we utilize is **Jochen Triesch Static Hand Posture Database II** which consists of more than 1000 RGB images of 128×128 pixels with 12 hand gestures performed by 19 persons with different skin colors under three backgrounds, *i.e.*, light, dark and complex. Considering the computational resource, we utilize linear interpolation to normalize all the images into the identical size of 50×50 pixels. Referring to [135], we collect the first 10 images from each gesture category as our learning set, the following 5 images from each category form the evaluation set and the remaining data are used as the testing set.

4.4.3 Results

For the Caltech-101 dataset, we select the best MOGP-evolved feature descriptor with 78.2 % from a set of potential solutions using the evaluation set and finally achieve a classification accuracy of 80.3% on the testing set using the linear-SVM. Since MOGP can effectively

Table 4.5 Comparison of image classification accuracies (%) based on color images.

Accuracy Methods	Method description	RGB fusion method	Caltech-101	MIT urban and nature	CMU PIE	Jochen Triesch Static Hand Posture II
Proposed	MOGP-evolved holistic descriptors	Tree structure fusion	**80.3**	**88.5**	**81.2**	**91.4**
Hand-crafted features						
Flattened image	Flatten the 2D raw image to a 1D vector	Concatenation	27.6	22.3	44.3	32.1
Shape context	Apply *Canny* edge detector to obtain contour information and diagram log-polar histogram bins (5 bins for *log r*, 12 bins for θ) are used in computing the shape context.	Concatenation	38.1	48.2	47.0	67.4
Color histogram	Build joint histograms of 3 colors in the RGB space and for each color component 4 bins (*i.e.*, $0-63$, $64-127$, $128-191$, and $192-255$) are used for a total of 64 dimensions.	Concatenation	20.1	27.6	34.7	28.8
HOG	Densely extract on a regular grid of 10×10 pixels with an overlap of 5 pixels along both the length and width and then concatenate the obtained features into a vector	Concatenation	60.7	65.4	55.3	70.2
P-HOG	Densely extract on a regular grid of 10×10 pixels with an overlap of 5 pixels along both the length and width and then concatenate the obtained features into a vector	Concatenation	67.2	71.8	60.0	74.2
SIFT	Densely extract on a regular grid of 10×10 pixels with an overlap of 5 pixels along both the length and width and then concatenate the obtained features into a vector	Concatenation	63.3	66.0	58.4	75.6
LBP	Densely extract on a regular grid of 10×10 pixels with an overlap of 5 pixels along both the length and width and then concatenate the obtained features into a vector	Concatenation	58.3	60.3	53.2	71.8
Gabor bank	Gabor filtering with 4 orientations at 6 different scales and output of each filter is then averaged on a 4×4 grid to form a vector.	Concatenation	67.1	77.2	67.2	77.9
BIF	Biologically-inspired feature combining orientation, intensity and color information to represent images, see in [125]	Contained in method	75.8	82.1	77.6	87.4
HRSE	Use a very low dimensional feature to represent the dominant spatial structure of a scene, see in [75]	Concatenation	72.4	78.0	73.1	85.5
Texton histogram	A 512-entries texton dictionary [153] is built by clustering responses to a bank of filters with 8 orientations, 2 scales and 2 elongations and then mapping all images into 512D histograms.	Concatenation	73.4	78.7	75.2	86.1
Centrist	Census transform histogram encoding the structural properties within an image and suppressing detailed textural information, see in [154]	Concatenation	75.1	84.0	74.3	86.7
Machine learned features						
DBN	Train a hierarchical architecture (with neuron numbers in hidden layers 500-500-2000) using the deep brief network with backpropagation fine-tuning for feature extraction	Learned	78.9	82.3	78.2	90.5
CNN	Train a 5-layer feature extraction mechanism using the convolutional neural network	Learned	75.8	80.8	77.8	89.4

```
Cat2(Maxpooling4(Gau1(Aver(GBO-0(GauX
(GBO-90(GauX(LoG1(Lap(SUB(DIV(ADD
(GauY(LoG1(Aver(Gau2(Ir)))),SUB(Ig,Igray)),Aver
(ADD(Ig,SUB(Ib,GauY(Igray))))),ABSsub(ABS
(GBO-90(Gau1(Ig))),Ig)))))))))),1.83,GBO-90
(GauX(LoG1(ABS(SUB(GauY(LoG1(ABS
(Igray))),ABSsub(SQRT(Med(Ib)),Igray)))))),0.70)
```

Fig. 4.12 The LISP format of the (near-)optimal feature descriptor generated through MOGP on the MIT urban and natural scene dataset.

restrain the *overfitting* and the number of samples in the testing set is much larger compared with the evaluation set, the (near-)optimal descriptor achieves a relatively unexpected higher result in the final testing set. In some sense, MOGP has managed to perform the task of feature extraction all by itself, without any help from any domain expert or background knowledge. On the other hand, this has somehow demonstrated the scalability of our approach for large-scale object recognition. The corresponding MOGP program is visualized in Fig. 4.11, in which Gaussian, Laplacian and Gabor operators have been automatically selected by MOGP at the filtering layer to extract the orientation and intensity features, and several scales of pooling operators are able to obtain the most robust and distinctive responses to different data resolutions on the top layer. The whole learned architecture is indeed consistent with the physical structure of the human visual cortex.

The results on the MIT urban and natural scene dataset are shown in Table 4.5. As expected, the best MOGP-evolved feature descriptor achieves a classification accuracy rate with 88.3% on the evaluation set and 88.5% on the testing set, since this scene dataset is a relatively easy dataset with small intra-class variations and large inter-class variations. Fig. 4.12 shows the LISP format of the corresponding MOGP program. In addition, we plot the confusion matrix of accuracies for the MIT urban and natural scene dataset in Fig. 4.13. We can see from this confusion matrix that, our evolved descriptor can extract meaningful information and lead to excellent classification accuracies on categories such as: Mountain, Forest and Coast&beach, however, yields the greatest confusion between Street and City Center which are, in some cases, intuitively hard to distinguish reliably.

On the CMU PIE face dataset, we successfully obtain a (near-)optimal feature descriptor through MOGP evolution as shown in Fig. 4.14 Although the face images, which are recorded with varying poses, illumination conditions and facial expressions, in this dataset

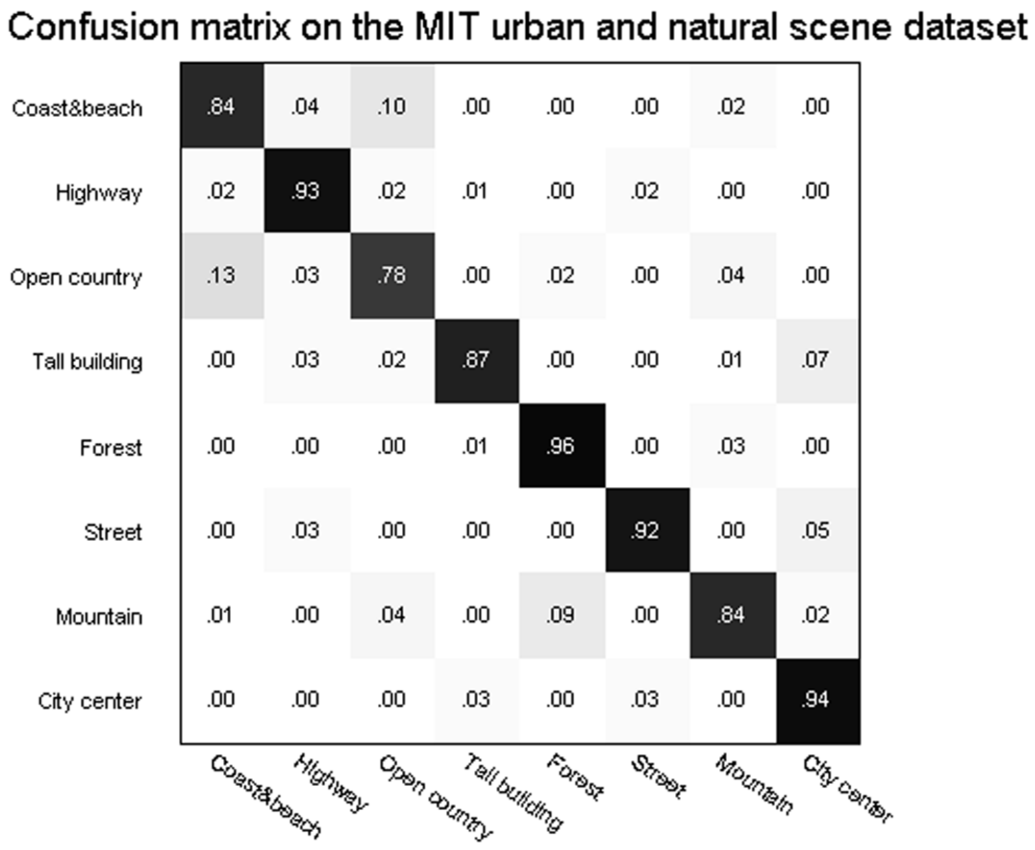


Fig. 4.13 The confusion matrix of classification results on the MIT urban and natural scene dataset.

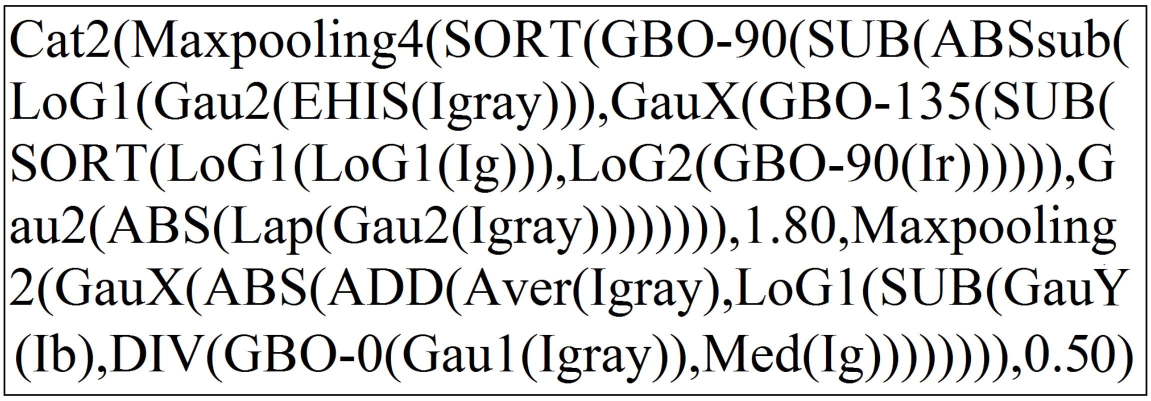

```
Cat2(Maxpooling4(SORT(GBO-90(SUB(ABSsub(
LoG1(Gau2(EHIS(Igray))),GauX(GBO-135(SUB(
SORT(LoG1(LoG1(Ig))),LoG2(GBO-90(Ir)))))),G
au2(ABS(Lap(Gau2(Igray))))))),1.80,Maxpooling
2(GauX(ABS(ADD(Aver(Igray),LoG1(SUB(GauY
(Ib),DIV(GBO-0(Gau1(Igray)),Med(Ig)))))))),0.50)
```

Fig. 4.14 The LISP format of the (near-)optimal feature descriptor generated through MOGP on the CMU PIE face dataset.

```
Cat1(Maxpooling8(GBO-45(GauX(LoG1(ABS
(SUB(GauY(LoG1(ABS(ADD(GauY(absSUB(Ir,
Igray)),ABS(ABSsub(GauY(SQRT(Lap(Ib))),SUB
(Ir,GBO-90(GauX(LoG1(ABS(ABSsub(LoG2(Ir),
ADD(Ig,Gau1(Igray))))))))))))),DIV(Igray, ADD(
SQRT(Ig),Igray)))))))),1.65)
```

Fig. 4.15 The LISP format of the (near-)optimal feature descriptor generated through MOGP on the Jochen Triesch Static II Hand Posture Dataset.

are quite challenging and complex for classification, our MOGP evolving procedure can tolerate the influence of these external factors and search for an optimized solution for face classification. Consequently, the best MOGP-evolved feature descriptor can still achieve relatively high results with 78.0% on the evaluation set and 81.2% on the testing set.

The Jochen Triesch Static Hand Posture Database II is a benchmark dataset for hand gesture recognition. In our experiments, our proposed method still works well to assemble a (near-)optimized feature descriptor by using MOGP. The LISP format of the evolved descriptor is shown in Fig. 4.15. Combining with the linear-SVM classifier, the MOGP descriptor outputs excellent performance on these gesture images with both simple and complex backgrounds. As a result, we use the evaluation set to select the best feature descriptor with 93.8% from the Perato front and achieve the final classification accuracy rate of 91.4%. on the testing set

For comparison, we have also evaluated some prevalent hand-crafted descriptors including flattened image, shape context, color histogram, HOG, pyramid HOG (P-HOG), SIFT, local binary pattern (LBP), Gabor bank [136], BIF, HRSE, texton histogram and Centrist [154]. As HOG, P-HOG, SIFT and LBP are usually used as local descriptors, dense sampling is applied on an image first and the final representation vector is the concatenation of the descriptor calculated on a dense grid. Most of these descriptors in comparison are designed for gray scale images. To fully exploit the color information of images, we compute these descriptors on the color versions of the images. For color images in the datasets, we calculate features on R, G, B and grey scale components, respectively, and then concatenate them into a long vector as the final representation. As the BIF method already encodes color information, we directly extract the BIF features from color images instead of calculating on color components. Under the same setting, we first adopt PCA for dimension reduction and then apply the linear SVM with 'ten-fold' cross-validation on the testing set to compute the recognition accuracies. All the results on color images are shown in Table 4.5.

In addition, we have also utilized two popular deep learning methods, *i.e.*, DBN and CNN, to learn hierarchical architectures for feature extraction on the combined learning and evaluation sets. In our experiments, we use *DeepLearnToolbox*[2] with default parameter settings according to previous publications by Hinton [139], to implement relevant tasks. We then apply the learned architectures (and associated parameters) to extract features on the testing sets and a same 'ten-fold' cross-validation is used with the linear SVM to calculate the recognition accuracies. To make the comparison fair, all the images used as the inputs of the architectures are the combinations of the R, G, B and gray scale components of the original color images. In Table 4.5, it is obvious that our MOGP-learned descriptors significantly outperform the state-of-the-art hand-crafted and machine learned features on four image datasets. The superior performance is mainly accredited to the simultaneous description and fusion of color components automatically assembled by the proposed methodology. The implicit supervised nature of the descriptor learning mechanism also contributes to the discriminative power of the MOGP-built descriptors.

4.4.4 Performance Analysis

To evaluate the performance of the proposed method, we also conduct analysis experiments on the four datasets to investigate the effects of our proposed multi-objective GP by comparing with the single-objective GP (with both tournament[3] and lexictour[4] [155] searching methods) which only considers the classification error rate as the main fitness. The testing results on the four datasets are shown in Table 6.4 . With the totally same experimental setting, the feature descriptors evolved by the proposed MOGP method lead to higher accuracies than the single-objective GP on all of the four datasets. Furthermore, for both single-objective GP based methods, it is obvious that using the lexictour searching scheme achieves better results than adopting the tournament searching scheme. In a sense, the lexictour scheme can also effectively restrain the tree *bloat* and reduce the training *overfitting*.

In addition, to illustrate time complexity of the feature learning process, we show the evolving time costs of MOGP and the single-objective GP in Table 6.5. It can be observed that our MOGP spends less time on evolving out a (near-)optimal feature descriptor than

[2]https://github.com/rasmusbergpalm/DeepLearnToolbox

[3]This method chooses each parent by randomly drawing a number of individuals from the population and selecting only the best of them.

[4]This method implements lexicographic parsimony pressure similar as tournament selection. The only difference is that the smallest individual (*i.e.*,fewest tree nodes) will be selected if more than one individual has the same best fitness in the selection competition.

Table 4.6 Evaluation of different evolutionary methods on the Caltech-101, the MIT urban and nature, the CMU PIE and the Jochen Triesch Static Hand Posture II datasets. NB: for single-objective genetic programming, we do not need to use the evaluation set and only adopt the learning set and the testing set for final results. The rest of the experimental setting is totally the same as our proposed method.

Datasets Methods	Caltech-101	MIT urban and nature	CMU PIE	Jochen Triesch Static Hand Posture II
Single-objective genetic programming+tournament searching	77.1%	84.5%	76.7%	89.3%
Single-objective genetic programming+lexictour searching	78.6%	84.8%	78.2%	90.0%
Multi-objective genetic programming	**80.3%**	**88.5%**	**81.2%**	**91.4%**

Table 4.7 The time costs of feature learning on the four datasets we used by applying three different genetic programming methods (Note that Matlab 2011a is used for coding)

Datasets Methods	Caltech-101	MIT urban and nature	CMU PIE	Jochen Triesch Static hHand Posture II	Total time
Single-objective GP+tournament searching	8.2h	3.6h	4.7h	3.1h	19.6h
Single-objective GP+lexictour searching	8.4h	3.7h	5.0h	3.3h	20.4h
Multi-objective genetic programming	**7.6h**	**3.3h**	**4.5h**	**3.0h**	**18.4h**

the single-objective GP. As many other learning algorithms, the training of the descriptors is time-consuming, but it can be performed offline. Once the optimal descriptor is obtained from the MOGP training phase, the classification phase will be very efficient, as the optimized descriptor can be just used as a handcrafted descriptor like SIFT. Of course, with the rapid development of silicon technologies, future computers will be much faster and even the training will become less a problem.

4.5 Summary

The goal of this work is to develop a domain-adaptive learning approach based on multi-objective genetic programming (MOGP) to generate (near-)optimal feature descriptors for image classification. MOGP is used to automatically evolve robust and discriminative feature descriptors with a set of domain-specific images and random constants as terminals, a number of primitive operators as functions, and both the classification error rate and tree complexity as the fitness criterion. The method can simultaneously extract and fuse features from different color and gray scale components of a color image. We have systematically evaluated our method on four different image datasets, *i.e.*,the Caltech-101, the MIT urban and nature scene, the CMU PIE and the Jochen Triesch Static Hand Posture datasets, and obtained superior results for image classification over previously published works. Although our MOGP method takes a quite long time for evolving, it only needs to be carried out once on the learning set and evaluation set, while the obtained (near-)optimal descriptor will be relatively generalized for image feature extraction and representation.

Besides learning the new descriptor via GP for visual classification, discriminative embedding method is also one of hot topics for feature representation learning. Thus, in the next chapter a new embedding method, termed Discriminative Partition Sparse Analysis (DPSA), will be introduced and the related experimental results show the DPSA can outperform other state-of-the-art algorithms for dimensionality reduction tasks.

Chapter 5

Discriminative Partition Sparsity Analysis

5.1 Overview

In the passed chapters, learning new feature representations via genetic programming has been proved to be effective for visual categorization. In this chapter, a another significant branch, i.e., dimensionality reduction, for discriminative representation learning is discussed.

Here, we develop a novel unsupervised linear dimensionality reduction algorithm, called Discriminative Partition Sparsity Analysis (DPSA). From the data probabilistic distribution point of view, samples in the high-dimensional space do not always follow the same distribution, but are naturally clustered into several groups. The data in each groupshare the same probabilistic distribution. To keep this property, we first apply the Gaussian Mixture Model (GMM) [156] to partition the training samples into different clusters. We then build a sub-graph weight matrix to describe the relationship between the data points in each cluster. Specifically, each data point is reconstructed as the linear combination of the remaining data samples in a cluster by minimizing the ℓ^1-norm of both the reconstruction coefficients and data noise, and the combination coefficients are designated as the values in the weight matrix. Different from constructing weight matrices via a manual neighborhood constraint as in LPP and NPE, the ℓ^1 weight matrix is more robust to data noise and can automatically realize sparsity. Besides, the neighbors selected through the ℓ^1 are also data-adaptive, which can discover the natural locality information of the data manifold and be a nice property for applications with uneven data distributions [157]. We further align these partitioned sub-graphs and obtain the final projection via a general linear reduction framework.

The contributions of this work lie in the following aspects:

(1) The proposed DPSA explicitly considers different probabilistic distributions that exist over the data points, meanwhile successfully preserves the natural locality relationship among the data.

(2) The proposed DPSA is proved to be robust to data noise, automatically sparse and adaptive to the neighborhood.

5.2 Literature Review

Dimensionality reduction [52] has become a key problem attracting much attention in many research areas, such as information retrieval, data mining [53], and pattern recognition [54, 55]. One baseline linear reduction algorithm is Principal Component Analysis (PCA) [158]. PCA is commonly applied to condense the information when the data manifold is embedded linearly or almost linearly in the ambient space. If the class information is available, Fisher Discriminative Analysis (FDA) [159] can be used to find an optimal subspace for discrimination where the final projection vectors can be obtained via maximizing the between-class covariance meanwhile simultaneously minimizing the within-class covariance. FDA has been proved to be successful on classification problems [56, 57].

Furthermore, another popular linear technique, termed Locality Preserving Projections (LPP) [42], has been proposed for dimensionality reduction that preserves local relationships within the data set and uncovers its essential manifold structure. After that, a Semi-supervised Discriminative Analysis (SDA) [160] embedding scheme has been developed, as well. SDA inherits the advantages from both FDA and LPP to find an appropriate subspace which perceives the intrinsic data structure from the high dimensional space and also maximizes the inter-class variation. Besides, a subspace learning algorithm called Neighborhood Preserving Embedding (NPE) has been used for linear reduction. Different from PCA, which aims at keeping the global Euclidean structure, NPE also aims at keeping the natural neighborhood structure on the data manifold.

Compared with non-linear methods, the above linear techniques are computationally much cheaper. Moreover, they yield projections that are not only defined on training data points, but also efficient for ‘out-of-sample’ extensions on novel test data. Naturally, for those ‘Big Data’ applications, we cannot manually annotate the ground truth for all training samples. Thus, efficient reduction methods without using label information but can still well describe the data manifold in the projected space are badly in need.

5.3 Linear Reduction framework

In this Section, we provide a general framework for the existing subspace learning algorithms from the graph embedding point of view.

Given a graph $G = \{X, W\}$, each of vertices indicate a data sample, let W be a symmetric $N \times N$ matrix with $W_{i,j}$ having the weight of the edge joining vertices i and j. The purpose of graph embedding is to represent each data points as a low dimensional vector that can effectively preserves similarities between the data pairs in the high dimensional space. The similarity is always represented by the edge weight.

Let us now consider a set of samples X=$[x_1, x_2 \dots, x_i, \dots, x_N] \in \mathbb{R}^{m \times N}$. The problem of linear dimensionality reduction is to find a projection matrix $U \in \mathbb{R}^{m \times d}$ that maps $X \in \mathbb{R}^{m \times N}$ to Y=$[y_1, y_2, \dots, y_i, \dots, y_N] \in \mathbb{R}^{d \times N}$, *i.e.*, Y=$U^T X$, where $d < m$. The basic idea of linear reduction is to preserve the intrinsic data structure in the projected low-dimensional space. The optimal Y is given by minimizing:

$$argmin \sum_{i,j=1}^{N} ||y_i - y_j||_2^2 W_{i,j} \tag{5.1}$$

under an appropriate constraint. This objective function leads to a heavy penalty if neighboring vertices i and j are embedded far apart. Thus, minimizing this objective function is to ensure that if vertices i and j are 'close', then y_i and y_j are close as well. $W_{i,j}$ is the graph weight over the whole data set and $y_i, y_j \in Y$.

Obviously, using different graph embedding techniques will lead to different dimensionality reduction performance. In the next section, we will present our Discriminative Partition Sparsity Analysis (DPSA) via ℓ^1 sparse graph construction.

5.4 Discriminative Partition Sparsity Analysis

Discriminative Partition Sparsity Analysis (DPSA) is proposed to preserve the locality information on different data distributions for dimensionality reduction. It operates in three stages. In the first stage, each sample in the dataset will be assigned to an individual cluster via the Gaussian mixture model (GMM). Thus, the whole dataset can be automatically partitioned into several groups. In the second stage, for samples in each cluster, an objective function is designed to construct a sparse sub-graph via the ℓ^1-norm constraint, which can successfully preserve the local discriminative information. Since samples in one cluster can be seen as a part of the whole dataset, this stage is termed 'part optimization'. We then align all the part optimizations together to form a global coordinate. In the last step, termed

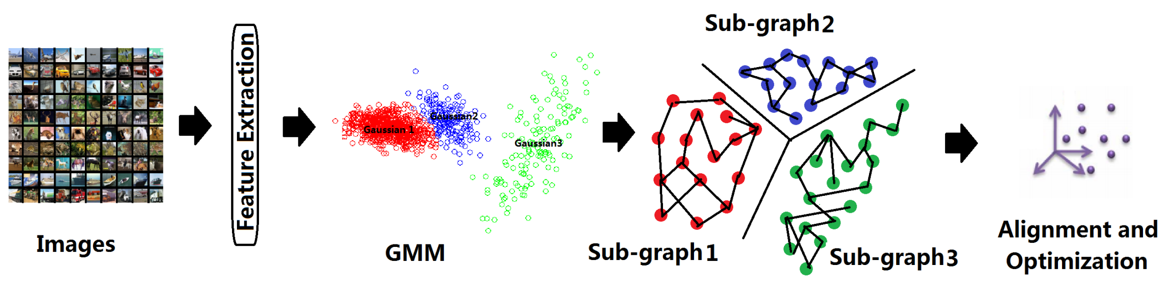


Fig. 5.1 The outline of DPSA. Here three clusters are used for illustration.

'global optimization', the projection matrix is obtained through a global alignment by solving a generalized eigenvalue eigenvector decomposition problem. Our proposed DPSA is outlined in Fig. 5.1.

It is worthwhile to highlight several aspects of the proposed approach here. DSPA shares some similar properties with other graph embedding algorithms, all of which aim to discover the local structure of the data manifold. However, our objective function is totally different from others. Furthermore, DSPA is regarded as a linear embedding. This makes it fast and suitable for practical applications. Finally, DSPA is also an unsupervised scheme, which is better for large-scale tasks where the label information is often unavailable.

5.4.1 Part Optimization

In the data space, each data point can be naturally assigned to a potential cluster, in which all samples share the same probabilistic distribution. Meanwhile, samples from different clusters always have different probability density functions. However, in either statistics or physics, real-world data distribution basically follows the same form, *i.e.*, Gaussian distribution. Therefore, each potential cluster could be Gaussian distributed but with different probabilistic parameters. Thus, how to estimate the Gaussian parameters for different data distributions becomes a core problem.

The Gaussian Mixture Model (GMM) as one of the clustering methods is regarded as the linear combination of different Gaussian components. The task of clustering is to group observations into different components through estimating each cluster's own parameters *i.e.*, ϕ, μ, Σ, under their likelihood function:

$$\begin{aligned} l(\phi,\mu,\Sigma) &= \sum_{i=1}^{m} \log p(x^{(i)};\phi,\mu,\Sigma) \\ &= \sum_{i=1}^{m} \log \sum_{z^{(i)}=1}^{K} p(x^{(i)}|z^{(i)};\mu,\Sigma)p(z^{(i)};\phi) \end{aligned} \tag{5.2}$$

Later, the Expectation Maximization (EM) algorithm [161] is always involved in such an estimation problem. Details of GMM can be found in [162].

After finishing GMM clustering, data are partitioned into K clusters $\{C_1, C_2, \ldots, C_k, \ldots, C_K\}$, and samples belonging to a certain cluster follow the same Gaussian distribution. To discover the intrinsic data structure, samples from different Gaussians are better to be considered separately. Thus, we first construct our partitioned sub-graphs for each cluster, respectively, instead of using all the data points.

In this chapter, for each cluster, we propose to obtain the sub-graph via the ℓ^1-norm constraint for data sharing the same distribution. Specifically, the neighboring samples of a data point and the corresponding similarities (graph weights) can be simultaneously calculated by solving an ℓ^1-norm optimization problem, which has been successfully utilized for spectral clustering [157], subspace learning [163], semi-supervised learning [164], *etc.*

Given a certain data point x with noise, a natural way to reconstruct this sample with a robust estimation of sparse representation α is formulated as:

$$x = D\alpha + \zeta = \begin{bmatrix} D & I \end{bmatrix} \begin{bmatrix} \alpha \\ \zeta \end{bmatrix} \tag{5.3}$$

where D is an over-complete dictionary, α indicates the sparse reconstruction coefficients, and ζ is the noise term. We further set B=$\begin{bmatrix} D & I \end{bmatrix}$ and α'=$\begin{bmatrix} \alpha \\ \zeta \end{bmatrix}$. Then, the ℓ^1-norm minimization problem can be solved for both the reconstruction error and data noise as follows:

$$\min_{\alpha'} ||\alpha'||_1, \; s.t. \; x = B\alpha' \tag{5.4}$$

In this chapter, since the sparse coefficients of the ℓ^1 construction can be used to indicate the similarities among different samples, we use the ℓ^1 sparse representation to construct our graph through part optimization. Each ℓ^1-graph, termed as partitioned sub-graph here, summarizes all the sample behavior of the corresponding cluster in sparse representation. The construction process is formally stated as the following three main stages:

1) **Input:** Data matrix includes a set of samples X_{C_k}=$[x_{C_{k1}}, x_{C_{k2}}, \ldots, x_{C_{ki}}, \ldots, x_{C_{kn}}] \in \mathbb{R}^{m\times n}$, where X_{C_k} denotes all data samples from the cluster C_k after GMM, n indicates the number of samples in C_k and $k \in K$.

2) **Robust sparse representation:** For each data point in a cluster, its robust sparse coding is achieved by solving the ℓ^1-norm optimization problem:

$$min||\alpha_i^{C_k}||_1, \; s.t. \; x_{C_{ki}} = B_i^{C_k}\alpha_i^{C_k} \tag{5.5}$$

where matrix $B_i^{C_k}$= $[x_{C_{k1}}, \ldots, x_{C_{ki-1}}, x_{C_{ki+1}}, \ldots, x_{C_{kn}}, I]$.

3) **Graph weight setting:** Denote $G_{C_k} = \{X_{C_k}, W^{C_k}\}$ as the sub-graph with the sample set X_{C_k} from the cluster C_k, and W^{C_k} as the corresponding graph weight matrix, and we set $W^{C_k}_{i,j}$=$\alpha^{C_k}_{i,j}$, if $i > j$; and $W^{C_k}_{i,j}$=$\alpha^{C_k}_{i,j-1}$, if $i > j$.

In more detail, given a data point $x_{C_{ki}} \in X_{C_k}$, the sparse graph is embedded through the ℓ^1-norm constraint among all the training samples from the same cluster. Specifically, the data sample $x_{C_{ki}}$ is represented by using all remaining data samples in C_k (*i.e.*, $X_{C_k} \backslash x_{C_{ki}}$). In real applications, we usually cannot estimate the existing noise in training data. Thus, constructing the ℓ^1 graph is constrained by a certain hyperparameter ε, which indicates the maximum value of the reconstruction error in the sparse representation. Then Eq. 6.4 can be rewritten as follows:

$$min||\alpha^{C_k}_i||_1, \ \ s.t. \ \ ||x_{C_{ki}} - b^{C_k}_i \alpha^{C_k}_i||_2 < \varepsilon \tag{5.6}$$

where $b^{C_k}_i$=$[x_{C_{k1}}, \ldots, x_{C_{ki-1}}, x_{C_{ki+1}}, \ldots, x_{C_{kn}}]$ and ε is always an extremely small value.

This ℓ^1 learning technique[1] makes the sparse graph more discriminative and robust to represent the relationship between two data samples from the same cluster, since it preserves the same-distribution correlation affinity, meanwhile discarding the different-distribution correlation after embedding. Furthermore, this kind of partitioned sub-graph can better reflect the data relations with the locality information for optimization. Particularly, such ℓ^1 embedding can effectively avoid influence from noisy data distributions and lead to a more precise final classification.

5.4.2 Global Optimization

We repetitively compute ℓ^1 sub-graphs for each cluster via the part optimization procedure. In this subsection, these partitioned graphs will be first unified as a whole: $G_{whole} = \{G_{C_1}, G_{C_2} \ldots, G_{C_k}, , \ldots, G_{C_K}\}$ and the corresponding weight matrix $W_{whole} \in \mathbb{R}^{N \times N}$,

$$W_{whole} = \begin{bmatrix} W^{C_1} & & & & \\ & \ddots & & & \\ & & W^{C_k} & & \\ & & & \ddots & \\ & & & & W^{C_K} \end{bmatrix} \tag{5.7}$$

G_{C_k} denotes the graph optimized from the cluster C_k. Since the unified graph G_{whole} is directed, W_{whole} will always be asymmetric. To satisfy the linear reduction framework

[1] ℓ^1-norm optimization toolbox is available at: http://sparselab.stanford.edu

mentioned in Eq. 5.1, we symmetrize W_{whole} by setting the matrix as $W=\frac{1}{2}(W_{whole}+W_{whole}^T)$. Then, after some simple algebraic formulations, we can transform Eq. 5.1 to:

$$argmin \sum_{i,j=1}^{N} ||y_i - y_j||_2^2 W_{i,j} = 2YLY^T \tag{5.8}$$

where, $L=D-W$ is the graph Laplacian, and D is a diagonal matrix whose entries are column (or row, since W is symmetric) sums of W, $D_{i,i} = \sum_j W_{j,i}$. It is easy to observe that L is a symmetric and semi-positive definite matrix. Finally, the minimization problem of Eq. 5.8 is reduced to a quadratically-constrained quadratic program:

$$\mathbf{min} \; \frac{YLY^T}{YY^T}, \;\; s.t. \;\; YY^T = I \tag{5.9}$$

where, the constraint $YY^T = I$ requires the projected data in the low-dimensional space to be uncorrelated.

Eq. 5.9 is a formulation of Rayleigh quotient. Thus, the optimal Y can be obtained by solving the minimum eigenvalue eigenvector problem:

$$YL = \lambda Y \tag{5.10}$$

However, the graph embedding approach described above only provides the mappings for the graph vertices in the training set. For classification purposes, a mapping for all samples, including new test samples, is required. If we can find a projection $U \in \mathbb{R}^{m\times d}$, we have $Y = U^T X$. Eq. 5.9 can be rewritten as:

$$\mathbf{min} \; \frac{U^T XLX^T U}{U^T XX^T U}, \;\; s.t. \;\; U^T XX^T U = I \tag{5.11}$$

The optimal projection U can be obtained by solving the minimum generalized eigenvalue eigenvector decomposition problem:

$$U^T XLX^T = \lambda U^T XX^T \tag{5.12}$$

Let the column vectors $U=\{U_0,\ldots,U_{d-1}\}$ be the solutions of Eq. 5.12, ordered according to their eigenvalues, $\lambda_0 \leq \ldots \leq \lambda_{d-1}$, from the smallest one to the $(d-1)$th smallest one. Therefore, $y_i \in Y$ is a d-dimensional vector after our DPSA reduction, and U is an $m \times d$ linear projection matrix.

5.4.3 Complexity Analysis

The computation of DPSA involves three steps: **1)** Data grouping into K clusters obtained by GMM; **2)** Partitioned sparse graph construction for each cluster via ℓ^1 optimization; **3)**

Table 5.1 The top recognition performance (%) on the USPS dataset, CMU PIE dataset and CIFAR-10 dataset, respectively.

Dataset \ Methods	Original feature	PCA	Fisherface	LDA	LPP	NPE	SDA	DPSA
USPS	93.04 (256)	93.16 (51)	89.74 (9)	95.28 (98,9)	94.79 (47)	94.67 (56)	95.24 (5,10)	**96.43 (30,39)**
CMU PIE	96.30 (1024)	96.42 (98)	96.65 (67)	97.88 (96,67)	97.04 (82)	97.47 (79)	97.82 (5,68)	**98.26 (28,54)**
CIFAR-10	82.26 (384)	83.21 (95)	82.22 (9)	84.98 (98,9)	83.71 (73)	83.88 (68)	84.74 (5,10)	**86.19 (120,60)**

Note that the numbers in parentheses are the corresponding feature dimensions with the best results after dimensionality reduction. For LDA, the first number is the percentage of energy retained in the PCA step, and the second number is the length of the final vector via Fisherface. For SDA, the first number is the K, which means the percentage (we fix it to a very small number, *i.e.*, 5%) of the labeled data used in the training phase, and the second number is the final feature length. For DPSA, the first number is the number of clusters by GMM and the second number is the reduced feature length.

Combining all partitioned sparse graphs and obtaining the final projection by solving the eigenvalue eigenvector problem.

Actually, the main computational cost lies in the first two phases. In the GMM phase (driven by the EM algorithm), it requires $O(KNT)$, where T is the number of iterations until convergence through EM. For the second phase, assuming each cluster after GMM has n samples, the complexity of graph construction for all the clusters is $O(Kn^2)$. Besides, the general complexity for the eigenvalue eigenvector problem is $O(N^3)$ in the last step. Thus, the total computational cost of DPSA is approximately $O(KNT) + O(Kn^2) + O(N^3)$.

5.5 Experiments and Results

In this section, we systematically evaluate the proposed DPSA on different datasets, in comparison to other popular dimension reduction algorithms.

5.5.1 Datasets

Three datasets are used to evaluate our DPSA algorithm, including handwritten digit images, face images and object images. The details of the three datasets are as follows:

The **USPS** handwritten digit database is described in [165], which contains 9298 16×16 handwritten digit images belonging to 10 classes. In our experiments, we split them into 7291 training images and 2007 test images.

The **CMU PIE** face dataset contains $41,368$ images from 68 subjects (people). Following [12], we select 11554 front face images, which are manually aligned and cropped into 32×32 pixels. Further, $7,500$ images are used as the training set and the remaining $4,054$ images are used for testing.

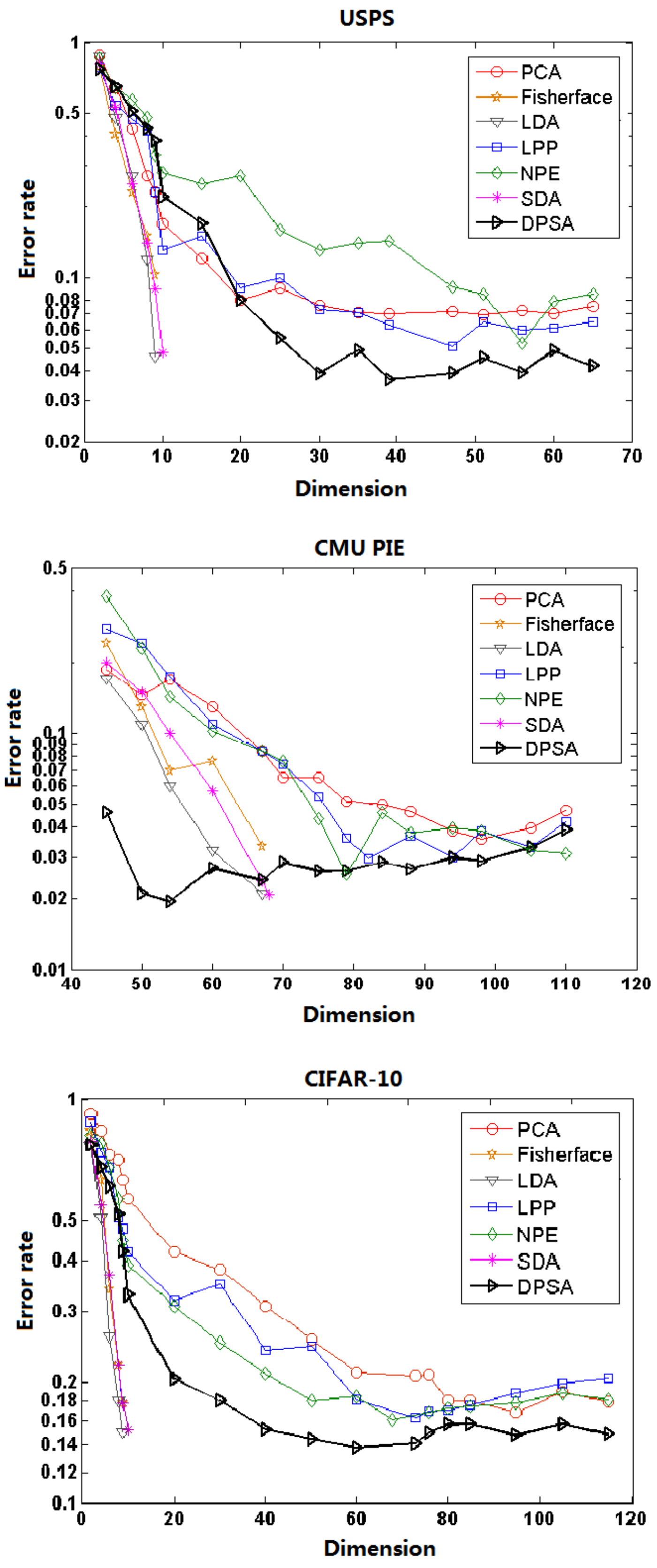


Fig. 5.2 The recognition error rate vs. number of dimensions on three datasets.

Table 5.2 Computational costs for DPSA with the highest classification accuracies.

Computational time / Datasets	Learning time	Coding time	Classification time
USPS (96.33%)	207 seconds (7291 training samples)	0.0032 seconds/sample	0.35 seconds/sample
CMU PIE (98.06%)	245 seconds (7500 training samples)	0.0048 seconds/sample	0.48 seconds/sample
CIFAR-10 (86.19%)	1521 seconds (50000 training samples)	0.0035 seconds/sample	0.83 seconds/sample

The **CIFAR-10** dataset is a labeled subset of the 80-million tiny images collection [166]. It consists of a total of 60000 32×32 color images in 10 classes. The entire dataset is partitioned into two parts: a training set with 50000 samples and a test set with 10000 samples and then we use a 384-d GIST descriptor to represent each image.

5.5.2 Results

We show the results of our DPSA algorithm on the three datasets compared with other state-of-the-art dimensionality reduction methods including PCA, Fisherface, LDA, LPP, NPE and SDA. Here, the Fisherface indicates the Fisher discriminative analysis, while LDA denotes the method of PCA+Fisherface.

Since our DPSA is a linear unsupervised reduction method, the methods we compare with are all linear unsupervised (or semi-supervised) except for Fisherface and LDA. For SDA, only a quite small number of labeled samples, with the rest of data unlabeled, are used to train the final projection. We compute the best recognition results for each method via the same linear SVM classifier. Table 7.2 lists the top recognition accuracies of the seven methods and their corresponding numbers of dimensions on the USPS, CMU and CIFAR-10 datasets, respectively. In addition, Fig. 5.2 also plots the corresponding curves of the recognition error rates of the seven comparable methods vs. the numbers of the projected dimensions.

In terms of the classification accuracy, our unsupervised DPSA approach consistently outperforms all the unsupervised methods, *i.e.*, PCA, LPP, NPE, and supervised (semi-supervised) methods, *i.e.*, Fisherface, LDA and SDA, on all three datasets. From Table 7.2, for both USPS and CMU PIE datasets, DPSA achieves 1.15% and 0.38% higher than the LDA, which gives the second best performance, and 1.64% and 0.79% higher than the best unsupervised methods on these two datasets, respectively. For the larger and more complex CIFAR-10 dataset, DPSA also reaches 1.21% higher than the best supervised method and 2.31% higher than the best unsupervised one. Regarding reduction effects, DPSA achieves lower dimensional representations on all three datasets, compared with other unsupervised techniques in Table 7.2.

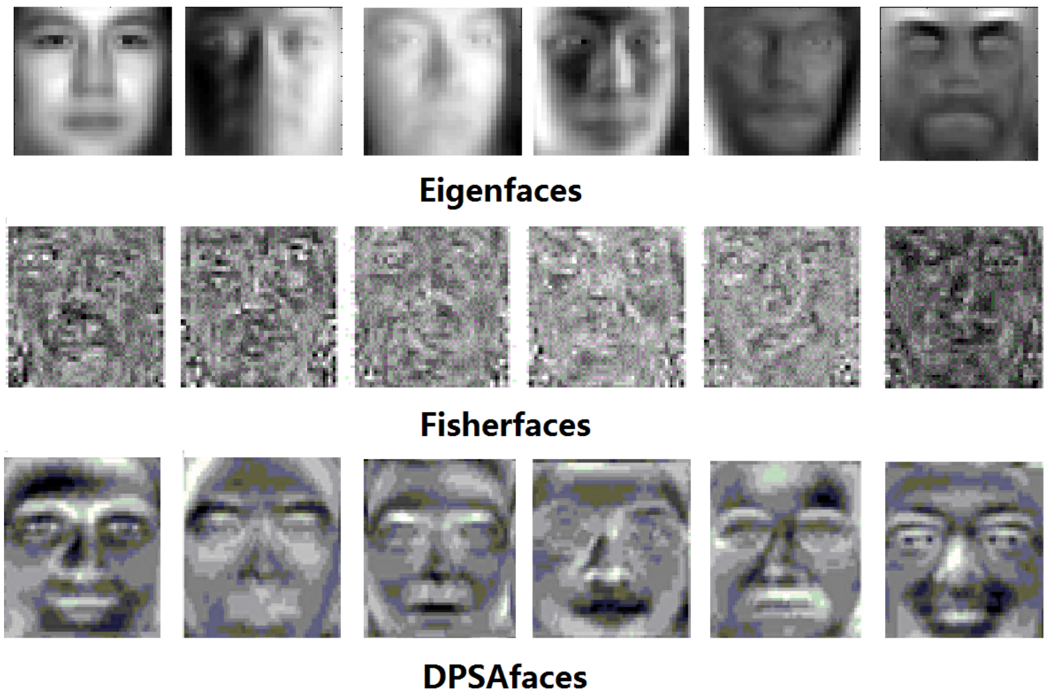


Fig. 5.3 The first 6 basis vectors of Eigenfaces, Fisherfaces, and DPSAfaces calculated from the face images in the CMU PIE dataset.

This is because that PCA produces a set of linearly uncorrelated principal component as the low-dimension projections, which only maximizes the variance of data features, but misses their intrinsic data structures in the original feature space. For LPP, the graph Laplacian indeed helps to keep the data locality structure in high dimension, and tries to preserve the same structure in the low-dimensional space as well. The locality information in LPP is always manually constructed via a neighborhood constraint, such that, if the two data points' pairwise distance exceeds a certain threshold, the value of graph Laplacian will be set as zero. However, this kind of construction is sensitive to data noise and one noisy feature may dramatically change the data's relationship. Furthermore, when data's distribution is not even, the weight matrix based on the pairwise-distance may also involve the far-distance inhomogeneous data together, if the threshold is large. The same drawbacks also exist in Neighborhood Perceiving Embedding (NPE) and Semi-supervised Discriminative Embedding (SDA).

For those supervised methods (*i.e.*, Fisherface and LDA), in our experiments, they involve the label information in their training phase and build an objective function to maximize the inter-class variation and minimize the intra-class variation. The advantage of Fisherface and LDA is that they can project the data into a very low-dimensional space with $C-1$, where C is the number of classes in the training data. However, the variation of the values of data from the same class is impaired and ignored in the reduced space. For instance, two far-away data points from the same class may be very close after projection, which would easily lead to over-fitting on testing data.

In contrast to all mentioned above, our DPSA treats the data samples on different distributions separately and focuses more on the natural relationship among samples, instead of manually setting a threshold to break the intrinsic data properties. Therefore, our DPSA is first clustered via GMM to partition all data into several groups, each of which shares the same Gaussian distribution. Then an ℓ^1 sub-graph is constructed to preserve the data locality structure in each cluster and finally all the sub-graphs are merged to compute the projection. DPSA shows its discriminative advantages for subspace learning as follows: (1) great robustness to data noise, (2) automatic sparsity instead of manual setting, and (3) adaptive neighborhood for each individual data point. Fig. 5.3 also visualizes the DPSA projection vectors as feature images on the CMU PIE face dataset, together with Eigenfaces and Fisherfaces.

Furthermore, a brief comparison of computational complexity is shown in Table 6.2. The results show that DPSA always needs a few hundred seconds for learning the projection. The learning speed highly depends on the size of the training set. Once the projection is obtained, it is very fast for the DPSA to code a new sample and classify it (always with the total time less than 0.9 seconds/sample in the Matlab environment).

5.6 Summary

In this chapter, we have presented a new unsupervised linear subspace learning approach, named Discriminative Partition Sparsity Analysis (DPSA). DPSA explicitly considers different distributions that exist in data points and also keeps the natural locality relationship among the data on each sub-distribution. Specifically, we introduced the Gaussian mixture model(GMM) clustering and sparsity optimization for dimensionality reduction tasks, in which each data point can be embedded via the ℓ^1-constraint to construct the graph and then the final projection is computed by solving the eigenvalue eigenvector problem .

We have systematically evaluated our method on the USPS, CMU PIE and CIFAR-10 datasets and produced the image classification accuracies of 96.43%, 98.26% and 86.19%, respectively. In all three datasets, our DPSA achieves better results compared with other popular supervised and unsupervised methods.

To make embedding methods more efficient for large-scale tasks, recently, binary embedding algorithm have attracted an increasing number of attention. In the next chapter, a new binary embedding approach via genetic programming has been proposed to solve large-scale visual classification and retrieval problems.

Chapter 6

Evolutionary Compact Embedding for Large-scale Image Classification/Retrieval

6.1 Overview

Effective dimensionality reduction is a classical research area for many large-scale analysis tasks in computer vision. In the last chapter, a new dimensionality reduction algorithm has been developed for data classification. Although the proposed one can achieve better results, it is not scalable for big data tasks. Several recent methods has attempted to learn binary hashing for fast and accurate applications in large-scale problems. In this chapter, we propose a novel framework to automatically learn the task-specific compact coding, called evolutionary compact embedding (ECE), which can be regarded as an optimization algorithm combining genetic programming (GP) and a boosting trick. As an evolutionary computation methodology, GP can solve problems inspired by natural evolution without any prior knowledge of the solutions. In our evolutionary architecture, each bit of ECE is iteratively computed using a binary classification function, which is generated through GP evolving by jointly minimizing its empirical risk with the AdaBoost strategy on a training set. We address this as greedy optimization leading to small Hamming distances for similar samples and large distances for dissimilar samples. The final optimized reduction representation is defined as the code calculated from the non-linear GP-evolved binary learner for each embedding bit. To the best of our knowledge, this is the first time that GP with the boosting trick has been successfully applied to feature embedding for large-scale image classification/retrieval. We then evaluate ECE on four datasets: USPS digital hand-writing, CMU PIE face, CIFAR-10 tiny image and SUN397 scene for classification and two dataset-s: SIFT 1M and GIST 1M for retrieval, showing the accurate and robust performance of our method for large-scale tasks.

It is worthwhile to highlight several properties of the proposed method:

(1) To the best of our knowledge, this is the first time that GP with the boosting trick has been successfully applied to feature embedding for large-scale image categorization and retrieval.

(2) To improve the efficiency of this work, a faster embedding scheme, i.e., Random batch parallel learning (RBPLECE), is proposed to speed up the learning phase without degrading much of the accuracy.

(3) The proposed methodology and the generated descriptors can be also applied to other data forms such as video or RGB-D data.

6.2 Literature Review

One of the most baseline dimensionality reduction algorithms might be principal component analysis (PCA), which is used to explain the variance-covariance structure of a set of variables through linear combinations of those variables. PCA is most commonly applied to condense the information contained in a large number of original variables into a smaller set of new composite variables or dimensions, at the same time ensuring a minimum loss of information. Another effective scheme for dimensionality reduction is linear discriminant analysis (LDA). LDA is a supervised method that has been proved successful on classification problems [56, 57]. Following the Fisher discriminant criterion, the projection vectors are commonly obtained by maximizing the between-class covariance and simultaneously minimizing the within-class covariance. However, the classical LDA is a linear method and cannot tackle nonlinear problems. In order to overcome this limitation, kernel discriminant analysis (KDA) [58] is then developed. KDA is the nonlinear extension of LDA using the k-ernel trick that can be implicitly performed in a new feature space, which allows non-linear mappings to be learned. Beyond that, some other dimension reduction methods can also achieve promising results for different applications. Locality preserving projections (LP-P) [42] are linear projective maps that are obtained by solving a variational problem that preserves the neighborhood structure of the data set. LPP aims to find the optimal linear approximations to the eigenfunctions of the Laplace Beltrami operator on the manifold. In addition, another popular method, termed discriminative locality alignment (DLA) [59], is also been used as dimensionality reduction algorithms for classification. All the above methods can be thought as the direct graph embedding or its linear/kernel/tensor extensions of a specific intrinsic graph that describes certain desired statistical or geometric properties of a data set, with constraints from scale normalization or a penalty graph [167]. With the need for fast search and classification in large-scale vision applications, some recent effort has been turned to applying binary hashing techniques, which explore the approximate similarity search based on Hamming distance to effectively reduce the indexing time. Among this kind of methods, kernelized locality-sensitive hashing (KLSH) [44] has been successfully utilized for large-scale image retrieval and classification. KLSH is essentially a kernelized method of performing probabilistic dimension reduction of high-dimensional data. The basic idea is to hash the input items so that similar items are mapped to the same buckets with high probability. Beyond that, some deep learning methods are also used to learn binary coding unsupervised, *e.g.*, restricted boltzmann machine (RBM) [168].

6.3 Evolutionary Compact Embedding

In this section, the overall design of our evolutionary embedding algorithm is first introduced and then we describe how to train our GP classifier with the boosting trick.

6.3.1 Problem Formulation

Let us now consider the M-bit evolutionary compact embedding $Code = [b_1(x), \ldots, b_m(x), \ldots, b_M(x)]$, which maps the high dimensional representation into an M-dimensional string. Here, $b_m(x) \in \{1, 0\}$ is defined by: $b_m(x)$=$binary(f_{gp}(x))$, where $f_{gp}(x)$ indicates the classifier generated by GP and "$binary()$" function returns 1 if the argument is positive; returns 0 if otherwise. Thus, the result of $f_{gp}(x)$ is positive, $binary(f_{gp}(x))$ is equal to 1; otherwise, $binary(f_{gp}(x))$ is equal to 0.

Since our $f_{gp}(x)$ is originally designed for binary classification problems, here we use the pair-wise trick to transfer the multi-class classification issue to a binary one. Given a set of training samples $X = \{x_1, x_2, \ldots, x_n, \ldots, x_N\}$ with labels $Y = \{1, 2, \ldots, C\}$, we re-distribute them into a pair-wise format $X_{pair} = \{\ldots (x_n, x_p)_j, \ldots\}_{j=1}^{N^2}$ with labels $Y_{pair} = \{1, 0\}$. X_{pair} is the set of N^2 labeled training pairs such that $Y_{pair} = 1$ if pair data x_n and x_p belong to the same class, and $Y_{pair} = 0$ otherwise. Since any two sample in X should be assign together once to consist a data pair, thus for *full-possibility* we can obtain the size of X_{pair} is $N \times N = N^2$ pairs in total. Now, we can involve binary classifiers into iterative

AdaBoost learning to jointly minimizing its empirical loss:

$$L(X_{pair}, Y_{pair}, M) = \sum_{j=1}^{N^2} (binary(\sum_{m=1}^{M} [D_m \cdot F_{weak}^m(j)]) \neq Y_{pair}(j)) \tag{6.1}$$

where, $F_{weak}^m(j)$ calculates the result of data pair $(x_n, x_p)_j$ using m-th weak classifier; D_m is the m-th coefficient corresponding to F_{weak}^m. D_m controls and adjusts the pair-data classification result for each bit. Therefore, Loss function Eqn. 6.1 actually calculates the total classification error rate[1].

By adopting the AdaBoost scheme, each bit of ECE is iteratively optimized over same-labeled and differently-labeled sample pairs. Initially, each data pair is assigned the same weight value. At each iteration, incorrectly embedded samples, *i.e.*, the pairs of differently-labeled samples mistakenly regarded as from the same labels, are assigned larger weights, while the weights of correctly embedded samples are reduced. Hence, the next bits tend to correct the errors of the preceding ones.

Eqn. 6.1 reflects the final error rate on the classification of X_{pair} using ensemble weak classifiers. Minimizing Eqn. 6.1 aims at reducing the Hamming distances of embeddings between pairs of samples from the same class, while increasing the Hamming distances of embeddings between pairs of samples from different classes. However, like regular AdaBoost, it is difficult to directly get a proper $b_m(x)$ to optimize Eqn. 6.1 for multi-labeled embedding problems. Thus, in this chapter, we use genetic programming (GP) to automatically create binary classifiers for this optimization problem. Specifically, for each ECE bit, we evolve the entire GP system once to generate a relatively effective weak classifier F_{weak}^m (*i.e.*, $Error_{rate} < 0.5$) under weighted data distribution.

ECE iteratively computes each bit for samples through the GP bit optimization procedure. Based on the result (*i.e.*, error rate) calculated from each bit, boosting is then applied as a global optimization to balance the weights of different GP classifiers. Thus, the final loss function (Eqn. 6.1) will be decreased effectively by using this kind of weighted ensemble GP classifiers. In the following sub-section, we will describe our GP bit optimization and boosting global optimization algorithms.

6.3.2 Genetic Programming Bit Optimization

Genetic programming (GP) is an evolutionary computation (EC) technique that automatically solves problems without requiring the user to know or specify the form or structure

[1]For $F_{weak}^m = b_m(x_n) \otimes b_m(x_p)$, F_{weak}^m returns 0, when $b_m(x_n)$ and $b_m(x_p)$ are different, otherwise $\otimes$ returns 1.

Table 6.1 Functions in Genetic Programming

Function Name	Input	Description
$+$	x_1 and x_2	$y = x_1 + x_2$
$-$	x_1 and x_2	$y = x_1 - x_2$
$\times$	x_1 and x_2	$y = x_1 \times x_2$
$\div$	x_1 and x_2	$y = \frac{x_1}{\sqrt{1+x_2^2}}$
IF	x_1, x_2 and x_3	if $x_1 > 0$, $y = x_2$; otherwise $y = x_3$

of the solution in advance. Generally, GP programs can be represented as a tree structure during the evolution procedure. In this work, each individual in GP represents a candidate binary classifier and is evolved continuously through generations. To establish the architecture of our model, three important concepts: function set, terminal set and fitness function should be defined.

Terminal Set and Function Set

Individuals in the population are assembled from terminal and function nodes. Terminal nodes taken from the terminal set are utilized as input of the genetic program. Two kinds of terminals are always used in the terminal set: (1) feature terminals according to the image features (2) constant terminals. Similar to other example-based learning algorithms, these terminals remain unchanged throughout the learning process. In our classification model, we define pair data X_{pair} and random constant numbers between 0 and 1 as the terminal set for GP evolving. In each tree-based genetic structure, data is located at the bottom leaf of the entire tree and connects with the higher function nodes directly.

In addition, another key component of GP is the function set which constitutes the internal nodes of the tree and is typically driven by the nature of the problem. Usually for GP classification problems, '$+$', '$-$', '$\times$' and '$\div$' are adopted in the function set. The '$+$', '$-$', '$\times$' operators are used as their original meanings, *i.e.*, addition, subtraction and multiplication. However, '$\div$' is different from general division or protected division. '$\div$' in our model is called analytic division, which is proved leading to better results in GP regression problems [169]. Each of these four operators takes two arguments and returns one result. Additionally, we get another conditional function '*IF*' with three arguments. If the first value is positive one, it returns the second argument; otherwise the third argument is returned. The 'If' function allows discontinuous programs rather than insisting on smooth functions and allows a program to contain a different expression in different regions of a feature space [170]. Table 7.2 lists all these functions used in our GP classification model.

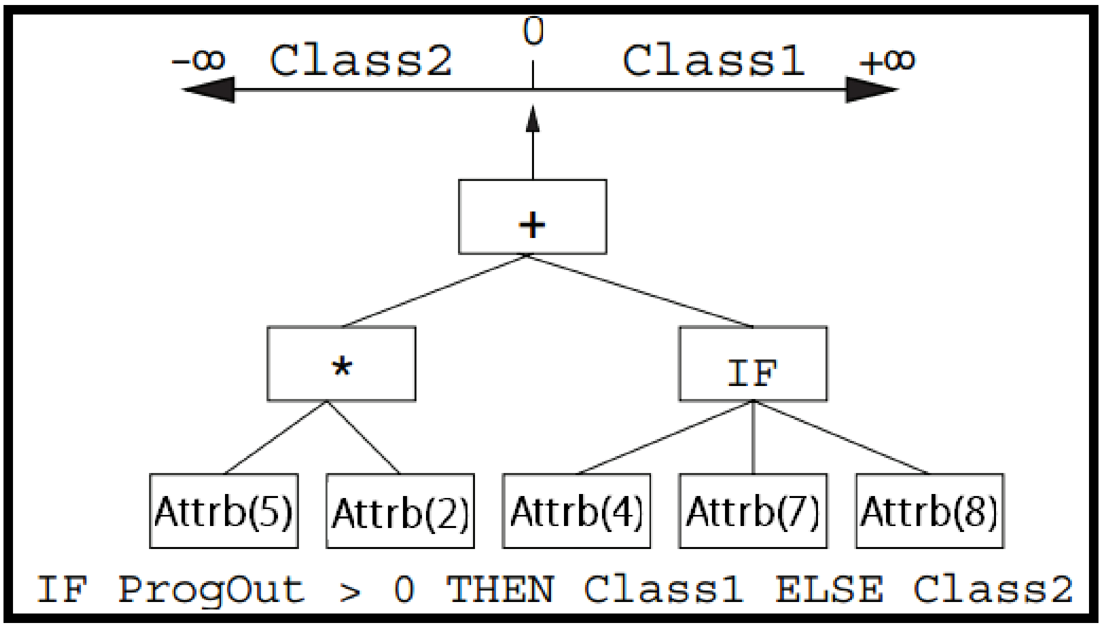


Fig. 6.1 Classification strategy using a GP program. Note: $Attrb(i)$ indicates the i-th feature of the input training vector.

GP Classification Scheme

The GP classifiers return a real value as output. In this way, there is a problem in this method of classification. This is because the task of binary classification requires a binary output rather than a continuous range of values as returned by the numeric expression representation. Therefore, a process of interpretation must be applied to convert the numeric output to a binary one. For two-class problems, the division point between the negative and non-negative numbers forms a natural boundary between the classes. Therefore, in our model, we set the zero as a boundary to separate two classes. If the GP output is positive, the example is predicted as belonging to one class, otherwise the other class. Fig. 6.1 illustrates how we use the output of a genetic program for binary classification. Numeric expressions have a hierarchical tree structure, which naturally suits the GP architecture. For numeric expressions to be evolved by the GP evolutionary search algorithm, a fitness measure must be derived.

Fitness Function

The fitness function in GP determines how well a program is able to solve the problem. For separating the pair-wise samples (*e.g.*, x_n and x_p) into positive (pairs of samples from the same class) or negative (pairs of samples from different classes), we use $F_{weak}^{m} = b_m(x_n) \otimes b_m(x_p)$

to distinguish each pair of data. $b_m(x)$ is the GP classifier for the m-th bit. Assuming a N^2 pair-wise sample dataset X_{pair} and their labels $Y_{pair} \in \{1,0\}$, we run the GP system for each bit, and the corresponding fitness function for the m-th bit is designed as follows:

$$fitness^m = [\sum_{F^m_{weak}(j) \neq Y_{pair}(j)}^{N^2} w^m_j] \times 100\% \tag{6.2}$$

where $F^m_{weak}(j)$ indicates the output of the j-th pair samples; $Y_{pair}(j)$ denotes the label of the j-th pair samples; w^m_j is the weight of the j-th pair samples for the m-th bit. This fitness function calculates the error rate by summing the weights of those wrongly classified data pairs. This is very similar to the AdaBoost by measuring the goodness of a weak hypothesis. In this way, GP can effectively get a relatively precise binary classification by continuously minimizing the value of fitness during the whole evolution procedure.

For large-scale datasets, the fitness function must be evaluated many times in each GP generation. For getting good results, a large number of generations is usually required, which leads to heavy computation. In our experiments, we implement parallel processing to speed up the GP learning algorithm. In our implementation, the large number of fitness evaluation can be performed by multiple processors at the same time, giving a tremendous reduction in the training time.

Evolutionary Parameters

For GP evolution, a lexicographic parsimony pressure has been applied as the selection method in our running. Like the original selection method, a random number of individuals are chosen from the population and then the best of them is selected. The only difference from the original selection is that, if multiple individuals are equally fit, the shortest one (the tree with the least number of nodes) is chosen as the best. Lexicographic parsimony pressure has shown its effectiveness for controlling the bloat [39] in different types of problems. In addition, we have adopted the 'totalelitism' scheme as the survival module in which all the individuals from both parents and children populations are ordered by fitness alone, regardless of being parents or children. This scheme has been demonstrated to lead to promising results in many applications. The ramped half-and-half method [171] was used for generating programs in the initial population. Table 6.2 shows these relevant parameters for GP evolving. It is noted that, since each GP classifier is evolved as a weak learner for the AdaBoost architecture, we empirically set the maximum number of generations as 50 which is proved to be enough for obtaining an acceptable weak learner (less than 50% classification error) in this case.

Table 6.2 Parameters for our GP algorithm.

Population size	300
Generation size	50
Crossover rate	75%
Mutation size	20%
Elitism rate	5%
Selection for reproduction	'lexictour'
Survival method	'totalelitism'
Stop condition	$\leq 0.1\%$

6.3.3 Boosting-Based Global Optimization

The previous section presents the theoretical algorithm for calculating each bit for the ECE code. However, we still haven't mentioned how to get the coefficient D_m for minimizing the loss function, Eqn. 6.1. To make our optimization convenient, we directly follow the Gentle AdaBoost scheme to update the D_m for each bit of ECE. Previous experiments show that Gentle AdaBoost performs slightly better than Real AdaBoost on regular data, but is considerably better on noisy data and much more resistant to outliers.

In our model, the F^m_{weak} with the lowest error rate $E_r = [\sum_{F^m_{weak}(j) \neq Y_{pair}(j)}^{N^2} w^m_j]$ via Eqn. 6.2 is selected as the best solution for the current m-th bit after GP evolving. The corresponding coefficient D_m for this F^m_{weak} can be then represented as: $D_m = 1 - 2E_r$. For the next bit GP optimization, the w^{m+1}_j for the j-th training sample pair can be updated as:

$$w^{m+1}_j = \frac{w^m_j exp(-D_m Y_{pair}(j) F^m_{weak}(j))}{\sum_{j=1}^{N^2} w^m_j exp(-D_m Y_{pair}(j) F^m_{weak}(j))} \tag{6.3}$$

Note that, for the first bit (m=1) of GP optimization, each data pair in the N^2 samples training set is initialized as the equal weight: $w^{m=1}_j = \frac{1}{N^2}$.

According to the above boosting-based global optimization, we can summarize our evolutionary compact embedding algorithm as follows: given the existing training pairs X_{pair} and their corresponding labels Y_{pair}, each bit is iteratively evolved by an individual GP optimization procedure with an updated sample weight w in the fitness function. In this way, to compute an M bit ECE code, we need to repetitively run GP M times. After ECE learning on the training set, for a new high dimensional data x, the final ECE code is represented as $Code = [b_1(x), \ldots, b_m(x), \ldots, b_M(x)]$.

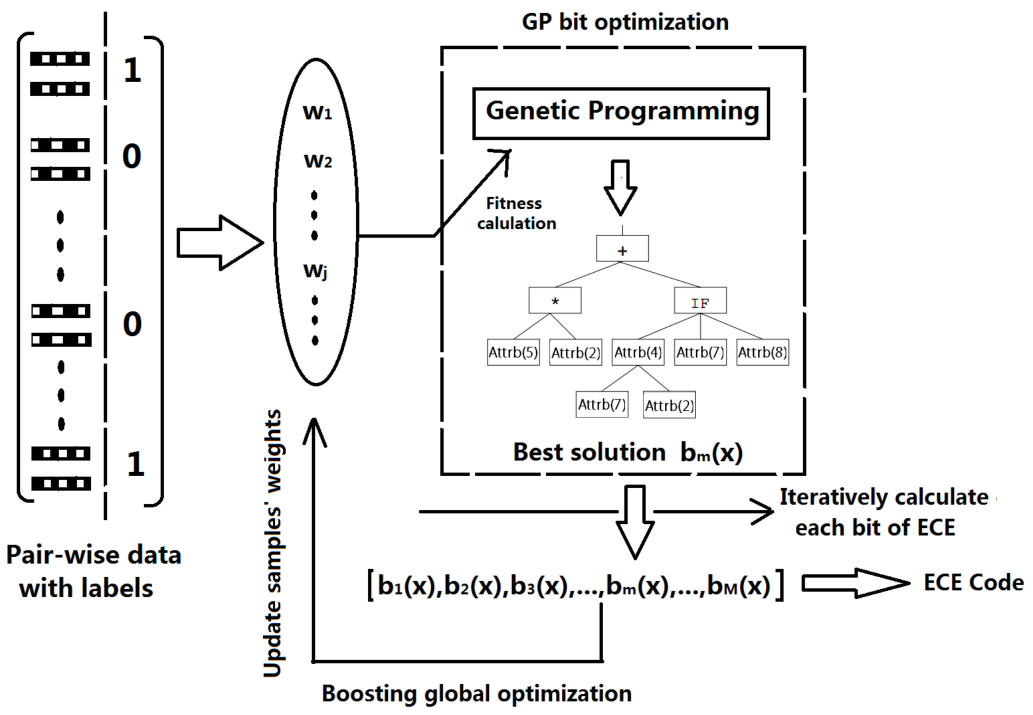


Fig. 6.2 Each bit of ECE is iteratively optimized by GP over same-labeled and differently-labeled sample pairs. Initially, each data pair is assigned the same weight. At each iteration, incorrectly embedded samples, such as pairs of differently-labeled samples mistakenly assigned to the same embedding value, are assigned a larger weight, while the weight of correctly embedded samples is reduced. Hence, the next bit tends to correct the errors of the preceding ones.

It is noteworthy that our approach can be regarded as an embedding learning method. Once our ECE embedding functions obtained via GP, they are fixed and then can be directly utilized on any new-coming data similar as a hand-crafted embedding scheme without re-learning. Fig. 6.2 visualizes the procedure of the ECE optimization scheme and the corresponding algorithm is depicted in Algorithm 4.

Algorithm 4 Evolutionary Compact Embedding

```
Input: A training set containing $N^2$ pairs of data$X_{pair} = \{\dots (x_n, x_p)_j, \dots\}_{j=1}^{N^2}$ with labels
$Y_{pair} \in \{1,0\}$, where $Y_{pair} = 1$ if pair data $x_n$ and $x_p$ belong to the same class, and $Y_{pair} = 0$
otherwise;
Aim: Learn an $M$-bits embedding code
First step
Initialize data weights: $w_j^{m=1} = \frac{1}{N^2}$ for the first bit optimization;
Second step
For $m = 1, ..., M$:
    1. Complete the GP bit optimization procedure to obtain best-performing $F_{weak}^m$ with
    the fitness function Eqn. 6.2, where $F_{weak}^m(j) = b_m(x_n) \bigotimes b_m(x_p)$;
    2. For each pair of data, the evolved weak classifier $F_{weak}^m$ calculates: $X_{pair} \to \{1,0\}$.
    The error rate is evaluated with respect to $E_r$=[$\sum_{F_{weak}^m(j) \neq Y_{pair}(j)}^{N^2} w_j^m$];
    3. If $E_r >= 0.5$, STOP loop; Otherwise,CONTINUE;
    4. Calculate the coefficient $D_m$ of this $F_{weak}^m$: $D_m = 1 - 2E_r$;
    5. Update the weights of the $N^2$ pairs of training data:
    $w_j^{m+1} = \frac{w_j^m exp(-D_m Y_{pair}(j) F_{weak}^m(j))}{\sum_{j=1}^{N^2} w_j^m exp(-D_m Y_{pair}(j) F_{weak}^m(j))}$;
End
Output:
The M-bits ECE code expression: $Code = [b_1(x), \dots, b_m(x), \dots, b_M(x)]$, where $x$ is a new
high dimensional feature .
```

6.4 Improved ECE Implementation for Large-Scale Applications

Our ECE method can theoretically reduce data of any dimension to a lower dimension compact code. However, the GP algorithm is always time-consuming for training on large-scale datasets, especially when the dimensionality of the original data is high.

To reduce the GP optimization complexity, we improve our ECE algorithm by using the random batch parallel learning (RBPL) technique. Given a training set $X = \{x_1, x_2, \dots, x_n, \dots, x_N\}$

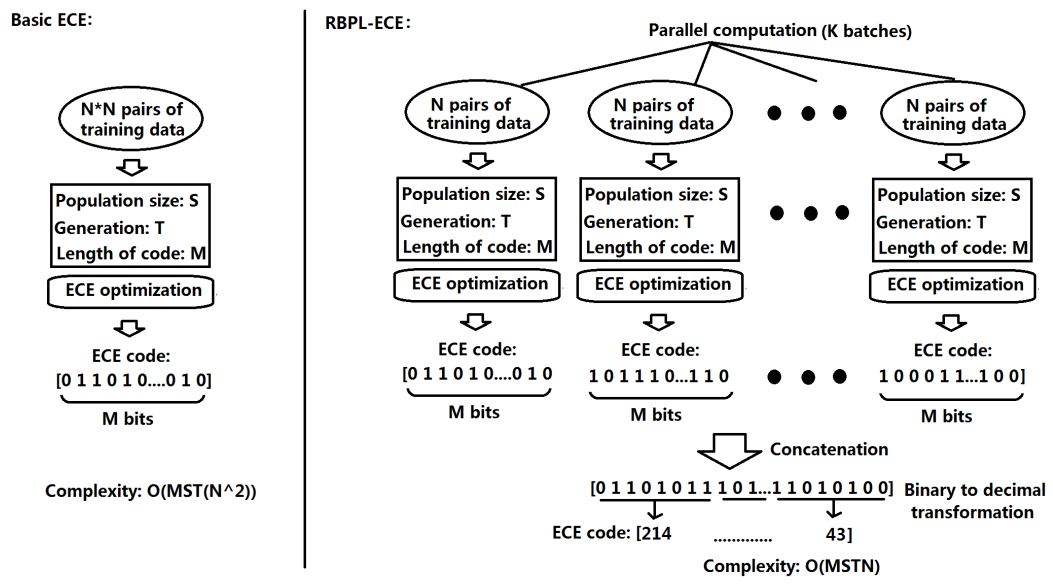


Fig. 6.3 Comparison between basic ECE and RBPL-ECE.

with labels $Y = \{1, 2, \ldots, C\}$, we randomly assemble them into N pairs$\hat{X}_{pair} = \{\ldots (x_n, x_p)_j, \ldots\}_{j=1}^{N}$ with labels $\hat{Y}_{pair} = \{1, 0\}$ using the half-half scheme[2], instead of generating a *full-possibility* N^2-sized data pair set X_{pair}. We repeat this kind of random assignment K times, so that K groups of pair-data sets are obtained: $\{\hat{X}^1_{pair}, \hat{X}^2_{pair}, \ldots, \hat{X}^K_{pair}\}$ with their corresponding pair labels $\{\hat{Y}^1_{pair}, \hat{Y}^2_{pair}, \ldots, \hat{Y}^K_{pair}\}$.

Our algorithms are implemented using Matlab 2013a on a server configured with a 6-core processor and 32G of RAM running the Linux OS. In this way, we can use parallel computation to separately learn an M-bits ECE for each $\hat{X}_{pair}$ at same time. We further concatenate these ECE codes into a long code. Although using the original *full-possibility* training set X_{pair} can theoretically learn a better ECE code than just applying any single N-pair set $\hat{X}_{pair}$, in fact the ECE codes calculated in parallel from different randomly assigned sets $\hat{X}_{pair}$ can effectively compensate each other's training errors (better resist overfitting). So the concatenated code can still keep the smallest Hamming distance for data from the same class and enlarge the Hamming distance for data from different classes. In terms of complexity, if each bit GP optimization needs a population of S individuals evolved via T generations, for embedding M bits using the basic ECE algorithm, the training complexity is $O(MSTN^2)$. Our RBPL technique can effectively reduce the basic ECE training complexity from $O(MSTN^2)$ to $O(MSTN)$.Thus, the RBPL-ECE implementation is about N times

[2]The half-half scheme aims to balance the training data by generating half of the pair-wise data belonging to label '1' and the rest of pairs belonging to label '0'. This scheme makes the training samples evenly fill the data space and effectively reduce the overfitting in the training phase.

faster than the basic ECE algorithm.

Furthermore, as the RBPL-ECE code is binary, for realistic applications, we always make a extremely compact code by transferring every 8 binary bits of RBPL-ECE code into a decimal number in the range of [0, 255]. The later experiments show that, after RBPL-ECE learning, we can efficiently compute compact codes for any data. It is proved to be fast and accurate for large-scale image classification applications by combining the SVM classifier with a RBF-Hamming-distance kernel. Fig. 6.3 illustrates our RBPL-ECE implementation.

6.5 Experiments on Image Classification

In this section, we systematically evaluate our proposed ECE with other popular dimension reduction algorithms on different datasets and relevant experimental results are compared and discussed in the following sub-sections.

6.5.1 Datasets

Three datasets are used to evaluate our evolutionary compact embedding algorithm, including handwritten digit images, face images and object images. The details of the three datasets are as follows:

The USPS handwritten digit database is described in [165], which contains 9298 16×16 handwritten digit images belonging to 10 classes. In our experiments, we split them into 7291 training images and 2007 test images. We further resize all images into 256-d vectors. In our experiments, we train all the algorithms on the first 3000, 4500, 6000, or 7291 images in the training set and evaluate on the 2007 test images.

The CMU PIE[3] face dataset contains $41,368$ images from 68 subjects (people). All the face images are captured by 13 synchronized cameras and 21 flashes under 13 different poses, 43 different illumination conditions and 4 different facial expressions. We select all the images with five near frontal poses (*i.e.*, No. 05, No. 07, No. 09, No. 27, and No. 29) under different illumination conditions and facial expressions. All the images are manually aligned and cropped. The cropped images are 32×32 pixels and converted to 1024-d vectors. Among the $11,554$ images, $8,000$ images are used as the training set and the remaining $3,554$ images are used for testing. Several cases are run with all the algorithms on the first 4000, 5000, 6000, 7000, or 8000 images in the training set.

[3]http://www.ri.cmu.edu/projects/project 418.html

The CIFAR-10 dataset[4] is a labeled subset of the 80-million tiny images dataset [166]. In particular, 60000 32×32 color images are included in CIFAR-10 with 10 classes and each of classes has 6000 samples. It consists of a total of in 10 classes, each of which has 6000 samples. Fig. 6.4 illustrates some example images from the CIFAR-10 dataset. The entire dataset is partitioned into two parts: a training set with 50000 samples and a test set with 10000 samples. We use a global GIST[5] [75] descriptor for each image, which is a 384-d vector describing the texture within localized grid cells. For all experiments on this dataset, we repetitively train all the algorithms on the first 20000, 30000, 40000, or 50000 images in the training set.

6.5.2 Compared Algorithms

The eight state-of-the-art algorithms which are compared with ECE in our experiments are listed below:

1. Principal Component Analysis (PCA), which is a mathematical procedure that uses orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

2. Linear Discriminant Analysis (LDA), which is related to Fisher's linear discriminant and provides us a baseline performance of linear algorithms for dimensionality reduction.

3. Kernel Fisher Discriminant Analysis (KDA) is regarded as the kernelized version of LDA. Using the kernel trick, KDA can be implicitly performed in a new feature space, which allows non-linear mappings to be learned.

4. Spectral Regression Kernel Discriminant Analysis (SRKDA) casts discriminant analysis into a regression framework which can improve both efficiency of computation and the use of regularization techniques.

5. Locality Preserving Projection (LPP) is linear projective maps that arise by solving a variational problem that optimally preserves the neighborhood structure of the data set. LPP should be seen as an alternative to PCA.

6. Discriminative Locality Alignment (DLA) focuses on the local patch of every sample in a training set and implements the sample weighting by the margin degree, which is used as a measure of the importance of each sample for classification.

7. Locality-Sensitive Hashing (LSH) is a randomized hashing scheme, developed with the primary goal of K neighbor search. The basic idea is to hash the input items so that similar items are mapped to the same buckets with high probability.

[4]http://www.cs.toronto.edu/ kriz/cifar.html

[5]We follow the common settings to use GIST features descriptors in this chapter. Of course, other features can also been extracted to represent images.

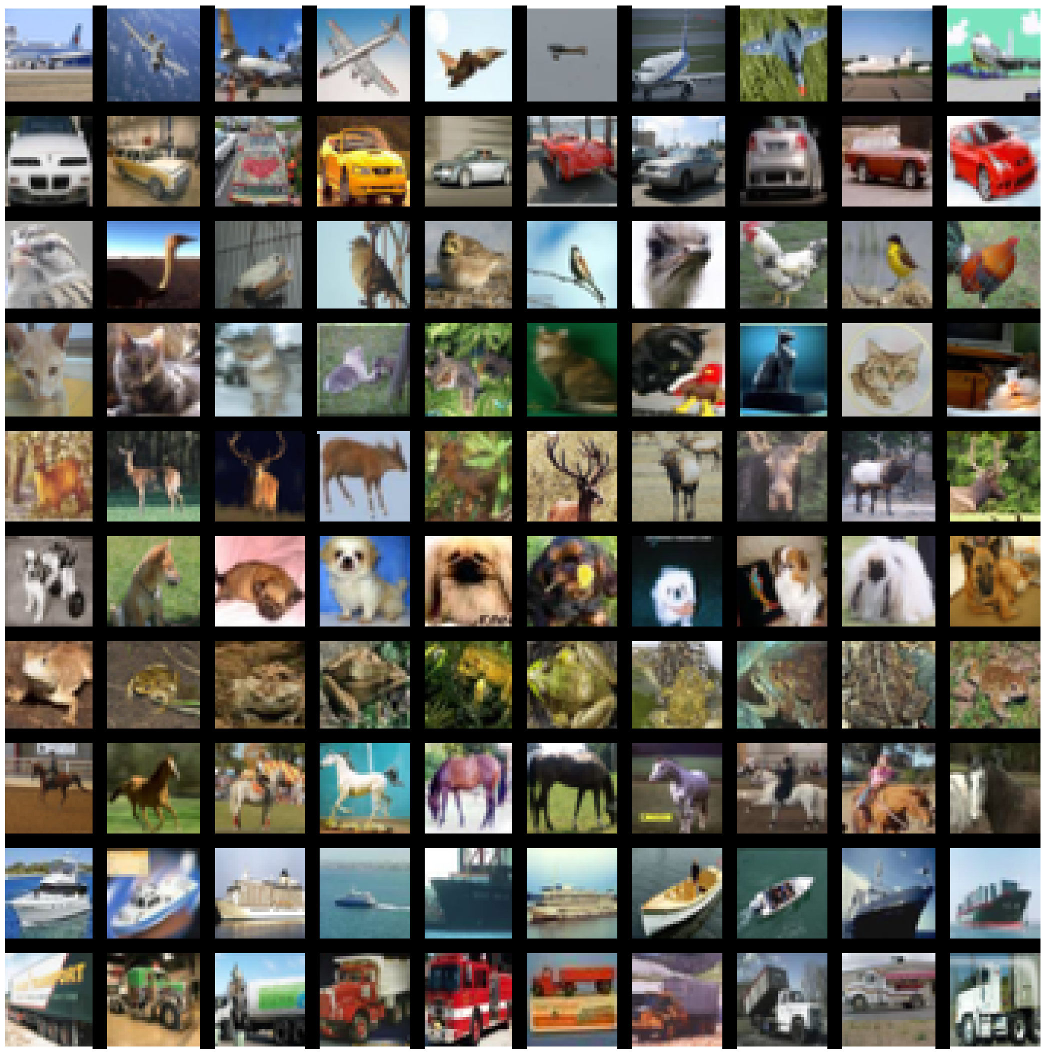


Fig. 6.4 A few example images from the CIFAR10 dataset. From top row to bottom row, the image classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck, respectively.

8. Kernelized Locality-Sensitive Hashing (KLSH) is a kernelized method of LSH performing probabilistic dimension reduction of high-dimensional data.

9. Boosted similarity sensitive coding (BSSC) is to construct an embedding similar to the one achieved in LSH, but to explicitly maximize its sensitivity to the desired similarity measure.

10. Restricted boltzmann machine (RBM) is a generative stochastic neural network that can learn a probability distribution over its set of inputs widely used in dimensionality reduction.

Note: the Gaussian RBF kernel is used for all the kernel-based methods. We manually tune the kernel width parameter σ by cross-validation on the training set to achieve best performance.

We combine all the above methods with (linear/non-linear) SVM classifier to evaluate their capability. Particularly, since LSH, KLSH, BSSC, RBM and ECE are all treated as hashing embedding methods based on Hamming distance, we apply RBF-Hamming kernel (see in Eqn. 6.4) with SVM in our experiments instead of the original Euclidean-RBF.

$$K(x_1,x_2) = exp(-\frac{HD(x_1,x_2)^2}{2\sigma^2}) \tag{6.4}$$

where $HD(x_1,x_2)$ indicates the Hamming distance between x_1 and x_2.

6.5.3 Results

We first show the results of our ECE algorithms on three datasets. Then we compare ECE with the above state-of-the-art dimensionality reduction methods.

ECE Vs. RBPL-ECE

We compare our basic ECE algorithm with RBPL-ECE on different numbers of bits as shown in Fig. 6.5. For each dataset, we use the whole training set (*i.e.*, training samples on USPS, CMU PIE and CIFAR-10 datasets are 7291, 8000 and 50000, respectively) to learn the ECE embeddings and test on the rest of each dataset for image classification.

From the results illustrated in Fig. 6.5, we can conclude that, for both basic ECE and RBPL-ECE, the classification accuracies on all three datasets at first rapidly climb up when increasing the ECE bits. Then, acceleration of the growing speeds gradually decreases and finally classification accuracies reach at a relatively stable level. We compute RBPL-ECE codes by randomly assembling K=3 parallel training batches. The best classification accuracies achieved by the basic ECE on the USPS, CMU PIE and CIFAR-10 datasets are

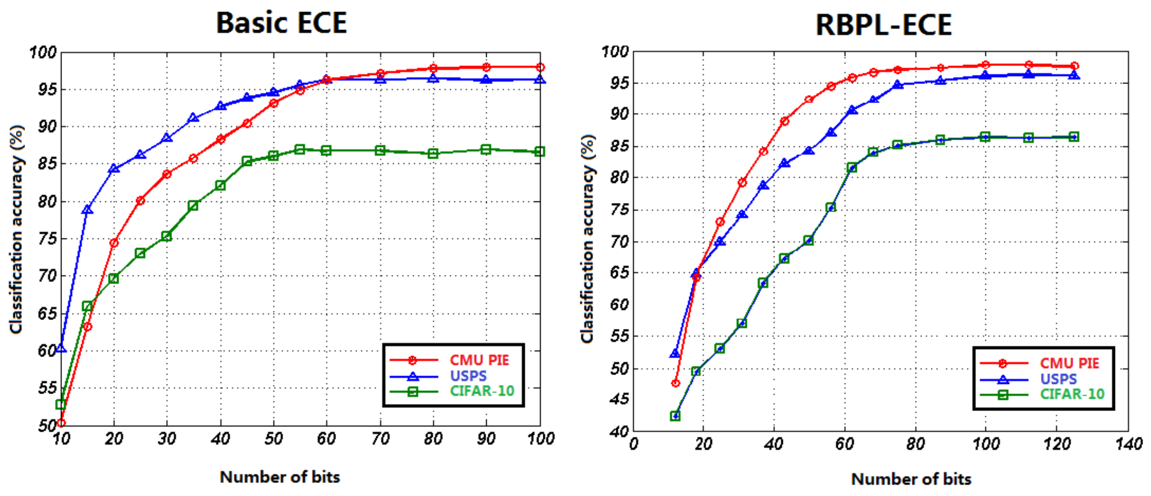


Fig. 6.5 Classification results via the basic ECE and RBPL-ECE: the left sub-figure shows the results of the basic ECE with a *full-possibility* N^2-sized data pair set for training. Each bit of the basic ECE code is a binary number - 1 or 0. The right sub-figure shows the results of RBPL-ECE, which adopts $K = 3$ random assigned N-sized learning batches to compute ECE codes in parallel.

Table 6.3 Basic ECE computational time with the highest classification accuracies.

Computational time / Datasets	Learning time	Coding time	Classification time
USPS (96.5%)	847 min(7291 training samples)	0.047s/sample	0.21s/sample
CMU PIE (97.6%)	891 min(8000 training samples)	0.056s/sample	0.21s/sample
CIFAR-10 (86.9%)	4786 min(50000 training samples)	0.051s/sample	0.54s/sample

96.5%(80-d), 97.6%(90-d) and 86.9%(90-d), respectively, while 96.1%(112-d), 98.15%(100-d) and 86.4%(87-d) are the best results produced by RBPL-ECE. It can be observed that, for both the USPS and CIFAR-10 datasets, the basic ECE just achieves slightly better results than RBPL-ECE.

Table 6.3 and Table 6.4 compare the computational costs with the highest classification accuracies for both the basic ECE and RBPL-ECE algorithms on three datasets. The results show that RBPL-ECE can vastly reduce the time during the learning phase with a tiny increase for the coding phase compared with the basic ECE. Once the evolutionary embedding functions are obtained, it is very fast for both the basic ECE and RBPL-ECE to code a new sample and classify it (always with the total time less than 0.6 seconds in the Matlab environment).

From the above evaluation, we prove that RBPL-ECE can get competitive results (*i.e.*,

Table 6.4 RBPL-ECE computational time with the highest classification accuracies.

Datasets \ Computational time	Learning time	Coding time	Classification time
USPS (96.1%)	12 min(7291 training samples)	0.071s/sample	0.21s/sample
CMU PIE (98.15%)	13 min(8000 training samples)	0.078s/sample	0.21s/sample
CIFAR-10 (86.4%)	71 min(50000 training samples)	0.073s/sample	0.53s/sample

Table 6.5 Performance (%) comparison on the USPS dataset.

Training Set \ Methods	Original feature	PCA	LDA	PCA+LDA	KDA	SRKDA	LPP	DLA	LSH	KLSH	BSSC	RBM	RBPL-ECE
3000	92.12 (256)	92.40 (49)	90.23 (9)	93.56 (98,9)	94.47 (9)	94.62 (9)	93.80 (112)	94.07(63)	89.40 (190)	92.93 (72)	92.20 (130)	91.20 (100)	**95.04** (10,87)
4500	92.46 (256)	92.57 (58)	90.48 (9)	93.72 (95,9)	94.47 (9)	95.12 (9)	94.52 (73)	94.23 (96)	91.31 (214)	94.21 (77)	93.12 (153)	93.04 (100)	**95.17** (10,87)
6000	92.81 (256)	92.64 (52)	90.08 (9)	94.43 (97,9)	94.97 (9)	95.57 (9)	94.88 (64)	94.55 (84)	93.16 (220)	95.13 (87)	94.83 (142)	94.47 (100)	**95.72** (10,100)
7291	93.04 (256)	93.13 (71)	89.74 (9)	94.78 (95,9)	95.17 (9)	95.96 (9)	95.09 (97)	94.79 (91)	93.97 (239)	95.94 (95)	94.92 (178)	95.51 (100)	**96.10** (10,112)

classification accuracies and dimensions) compared with the basic ECE but using much shorter time for learning. Thus, in the rest of experiments, we only adopt RBPL-ECE as our dimensionality reduction method to compare with the state-of-of-art.

Comparison with other methods

In this section, we compare our method against classical dimensionality reduction techniques, PCA, LDA, KDA, SRKDA [12], LPP and DLA [59] and popular binary coding methods, including BSSC [65], RBM [168] (with 100-100 two hidden layers) , LSH [172] and KLSH [44] (with RBF kernel). Each of these methods is evaluated on three datasets under different training settings. Table 6.5, Table 6.7 and Table 6.8 show the classification accuracies by SVM[6] for each method on the three datasets.

Note that the numbers in parentheses are the corresponding feature dimensions with the best results after dimensionality reduction. For PCA+LDA, the first number is the percent-

[6]The RBF kernel and RBF-Hamming-distance kernel are used. We manually tune the kernel width parameter σ by cross-validation on the training set to achieve best performance.

Table 6.6 Time complexity analysis for best performance on the USPS dataset. (Note: Coding time denote the time used for each sample

Time \ Methods	PCA	LDA	PCA+LDA	KDA	SRKDA	LPP	DLA	LSH	KLSH	BSSC	RBM	RBPL-ECE
Learning Time	12s	44s	56s	50s	89s	33s	132s	2s	6s	957s	1921s	720s
Coding Time	67ms	34ms	133ms	63ms	52ms	80ms	104ms	13ms	17ms	94ms	151ms	71ms

Table 6.7 Performance (%) comparison on the CMU PIE dataset

Methods / Training Set	Original feature	PCA	LDA	PCA+LDA	KDA	SRKDA	LPP	DLA	LSH	KLSH	BSSC	RBM	RBPL-ECE
4000	96.26 (1024)	96.26 (154)	95.86 (67)	96.76 (97,67)	96.44 (67)	96.93 (67)	96.35 (77)	96.50 (90)	89.71 (112)	92.53 (178)	91.76 (141)	93.15 (100)	**97.64** (10,87)
5000	96.59 (1024)	96.90 (161)	96.15 (67)	96.92 (91,67)	97.17 (67)	**97.75** (67)	96.49 (103)	97.03 (101)	92.24 (187)	94.76 (175)	93.75 (132)	94.79 (100)	**97.75** (10,87)
6000	96.95 (1024)	95.81 (241)	96.43 (67)	97.45 (98,67)	97.59 (67)	97.77 (67)	96.94 (96)	97.21 (94)	94.97 (139)	97.10 (175)	95.19 (197)	95.28 (100)	**97.98** (10,100)
7000	97.21 (1024)	96.89 (228)	96.60 (67)	97.78 (95,67)	97.82 (67)	97.91(67)	97.51 (132)	97.52 (89)	96.16 (172)	97.94 (181)	96.83 (111)	95.85 (100)	**98.10** (10,100)
8000	97.84 (1024)	97.04 (198)	96.65 (67)	98.10 (97,67)	97.89 (67)	97.93 (67)	97.87 (113)	97.81 (97)	96.77 (158)	98.13 (185)	97.53 (162)	96.07 (100)	**98.15** (10,100)

Table 6.8 Performance comparison (%) on the CIFAR-10 dataset

Methods / Training Set	Original feature	PCA	LDA	PCA+LDA	KDA	SRKDA	LPP	DLA	LSH	KLSH	BSSC	RBM	RBPL-ECE
20000	81.89 (384)	82.16 (164)	77.03 (9)	84.86 (95,9)	84.47 (9)	**85.02** (9)	83.35 (60)	84.10 (82)	79.04 (99)	81.13 (46)	81.79 (191)	84.38 (100)	84.77 (10,87)
30000	82.04 (384)	82.27 (171)	79.96 (9)	84.95 (95,9)	84.97 (9)	85.05 (9)	83.37 (84)	84.43 (86)	81.10 (134)	83.55 (51)	83.40 (154)	84.50 (100)	**85.02** (10,100)
40000	82.15 (384)	83.21 (154)	81.31 (9)	84.75 (95,9)	85.12 (9)	85.41 (9)	84.00 (82)	84.97 (92)	83.43(112)	84.74 (45)	84.13 (110)	85.22 (100)	**85.98** (10,100)
50000	82.27 (384)	83.19 (159)	82.22 (9)	84.98 (98,9)	85.20 (9)	85.55 (9)	83.71 (73)	85.29 (88)	84.71 (123)	86.23 (48)	84.78 (144)	85.85 (100)	**86.40** (10,87)

age of energy retained in the PCA step, and the second number is the length of the final vector. For RBPL-ECE, the first number is the K which means the number of the random batches for parallel computation, and the second number is the final feature length.

Among those methods based on graph embedding (*i.e.*, PCA, LDA, KDA, SRKDA, LPP), The SRKDA always gives the best performance for most of the cases in all the three datasets. KDA and LPP have similar abilities on the first two datasets. PCA+LDA can achieve better results than just PCA or LDA individually. When increasing the size of training sets from the smallest to the largest, the maximum gains of results for these graph embedding methods are 1.82%, 1.58% and 2.19%, respectively, on USPS, CMU PIE and CIFAR-10. On the contrary, for the binary coding based methods (*i.e.*, LSH, KLSH, RBM and BSSC), it can be observed that the classification accuracies are quite sensitive to the size of the training set compared with graph embedding methods. When adding the number of samples in their training phase, an obvious increase occurs on classification accuracies.

Compared with above methods, our RBPL-ECE is much more stable when changing the training set size compared with all other reduction methods. This property is useful in large-scale data classification, since a relatively small training set can achieve promising results. In general, our RBPL-ECE algorithm yields the best performance in large-scale image classification on all three datasets. Particularly, RBPL-ECE reaches the highest classification accuracies on the USPS dataset. For CMU PIE and CIFAR-10 datasets, only SRKDA achieves same or slightly better results than RBPL-ECE.

We also illustrate the time cost corresponding to their their best performance on the USPS dataset in Table 6.6. It is seems that our GP-based methods is still time-consuming compared with other embedding approaches for learning, even though computational complexity would not be very important in the area of GP training phase. Once the optimal embedding function is obtained from the GP training phase, the classification phase will be very efficient, as the optimized embedding function can be just directly applied on any newcoming data. Of course, with the rapid development of silicon technologies, future computers will be much faster and even the training will become less a problem.

In terms of reduction effect, LDA, KDA and SRKDA are treated as the most effective methods, which project the data from the original feature space to a subspace with the dimension of the class number minus 1. Our RBPL-ECE code is still longer than some of the embeddings we compare with, though our method is much faster for reduction. In future work, we will focus on optimizing our embedding method to make it more compact and effective.

6.5.4 Large-Scale Image Classification

Additionally, in this section, we further evaluate our RBPL-ECE on a very large dataset[7], *i.e.*, SUN397 [136], which contains 108754 scene images in total from 397 well-sampled categories with at least 100 images per category. We randomly select 5, 10, 20, 50 samples from each category respectively to construct the training set and the rest of samples are used as the test set. The same GIST feature is used to describe each image. In this way, we train our RBPL-ECE on these extracted feature vectors to learn the compact binary representation. We also compare all the above methods with ours and show the results in Fig. 6.6. From the results, we can observe that our RBPL-ECE still consistently outperforms all the compared methods in different settings.

6.6 Experiments on Image Retrieval

In this section, ECE algorithm on the high dimensional nearest neighbor search problem. Different from classification tasks, in realistic retrieval scenarios, we cannot get the precise label for each of the data point in large-scale retrieval tasks. Thus, we use an approximate scheme to obtain the weak label information. In particular, we first adopt a clustering method (e.g., K-means) to partition the data into several groups. Since this kind of clustering method is normally based on distances (e.g., Euclidean distance) to divide data into different groups, data points from the same cluster always have high similarity. Therefore, we assign pair label $Y_{pair} = 1$ if pair data x_n and x_p belong to the same cluster (group), and $Y_{pair} = 0$ if pair data x_n and x_p come from different clusters (groups).

[7]For the image classification task, the size of a large scale dataset is always over ten thousand in previous reports, such as Caltech-256 with 30607 images. A dataset containing over a hundred thousand images can be regarded as a very large dataset.

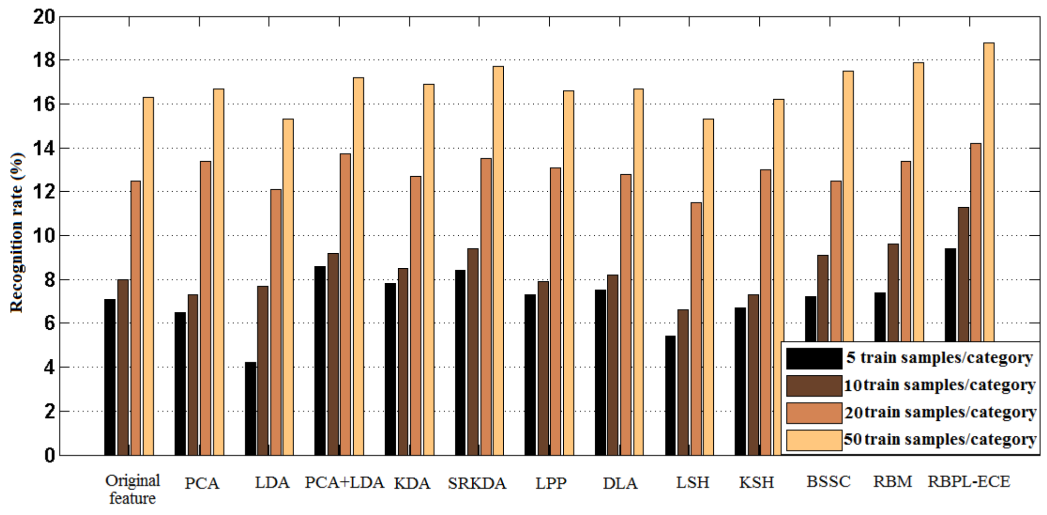


Fig. 6.6 The classification accuracies on the SUN397 scene dataset with different training-test partitions.

Following the previous reports, two large scale realistic datasets[8] are used in our experiments, *i.e.*, **SIFT 1M** and **GIST 1M** which both contain one million image features with $128-dim$ and $960-dim$ vectors, respectively.

For each data set, we randomly select 10k data points as the queries and use the remaining to form the gallery database for training. We generate the ground truth using the same criterion as in [173]. In test phase, similarly, a returned point is regarded as a true neighbor if it lies in the top 2 percentile points closest to a query. Hamming distances ranking is then used as the measurement for in our retrieval tasks, since it's fast enough with short hash codes in practice. We evaluate the retrieval results by the Mean Average Precision (MAP) and the precision-recall curve. Additionally, we also report the training time and the testing time (the average time used for each query) for all the methods. Our experiments are completed using Matlab 2013a on a server configured with a 12-core processor and 128G of RAM running the Linux OS.

6.6.1 Compared Methods and Settings

We compare our method against 10 popular hashing algorithms, *i.e.*, LSH [172], KLSH [44], RBM [67], BSSC [65], PCAH [68], SpH [69], AGH [70], STH [71], KSH [72] and CH [73]. Particularly, for KLSH and KSH, we both use the RBF-kernel and randomly sample 500 training samples to construct the empirical kernel map and set the scalar parameter σ always to an appropriate value on each dataset. To run RBM, we train it with a set of $100-100$

[8]Download here: http://corpus-texmex.irisa.fr/

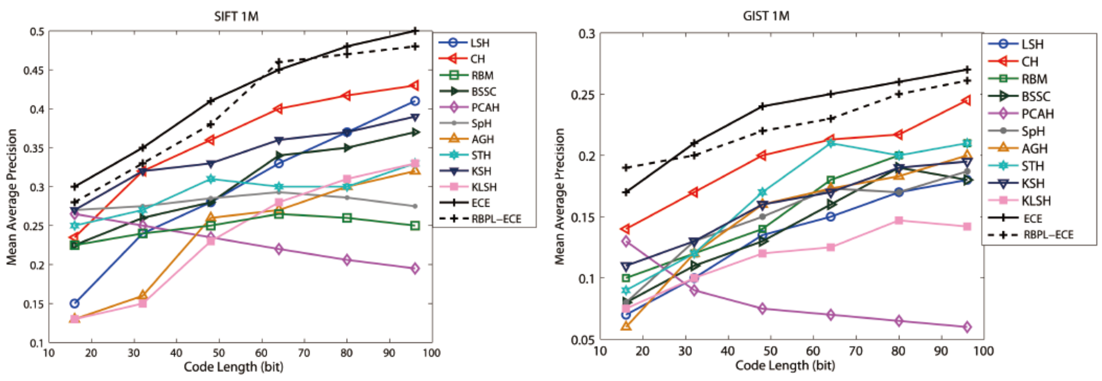


Fig. 6.7 The Mean Average Precision of all the algorithms on SIFT 1M and GIST 1M data sets.

hidden layers without fine-tuning. BSSC uses the labeled pairs scheme mentioned above in a boosting framework to learn the thresholds and weights for each hash function. AGH with two-layer is used in our comparison, which shows superior performance over AGH with one-layer [70]. We further set k=200 as the number of the anchor points and the number of nearest anchors in sparse coding as s=50. Both our CH method and the AGH need an anchor-based sparse coding step, thus same settings are applied in CH, as well. For our method RBPL-ECE, since we mainly evaluate the short hash codes, we just fix the batch number as K=4 in all experiments. The number of clusters of K-means in the proposed method for each dataset is selected from one of {600, 700, 800,. . . , 1000,. . . , 1500} with the step of 100, which yields the best performance by 10-fold cross-validation. Due to that basic ECE and RBPL-ECE are both inspired via genetic programming which is always initialized randomly, all the experiments with our methods have been repetitively carried out 10 times and the final results shown are the averages of the 10 runs with a degree of uncertainty. All of the above methods in our experiments are evaluated on six different lengths of codes (16, 32, 48, 64, 80, 96).

6.6.2 Results Comparison

Fig. 6.7 illustrates the MAP curves of all comparable algorithms on SIFT 1M and GIST 1M datasets. In its entirety, the searching accuracies on the SIFT 1M dataset are obviously higher than that on the more complicated GIST 1M dataset. In particular, for the PCAH, it has a high MAP when the code length is short. However, it fails to make significant improvements and the performance decreases as the code length increases. On the contrary, the rest of the methods keep an overall increasing or fluctuating tendency on MAP when

Table 6.9 MAP of 32 bits and 48 bits with training and testing time of all algorithms on SIFT1M and GIST1M data sets.

Methods	SIFT 1M						GIST 1M					
	32 bits			48 bits			32 bits			48 bits		
	MAP	Train time	Test Time	MAP	Train time	Test Time	MAP	Train time	Test Time	MAP	Train time	Test Time
LSH	0.240	0.3s	1.1μs	0.280	0.6s	1.9μs	0.107	1.4s	2.7μs	0.135	2.1s	3.0μs
KLSH	0.150	10.5s	14.6μs	0.230	10.7s	16.2μs	0.110	29.5s	27.2μs	0.120	30.7s	38.0μs
RBM	0.245	4.5×10^4s	3.3μs	0.262	5.8×10^4s	3.7μs	0.123	5.5×10^4s	3.4μs	0.142	6.2×10^4s	3.7μs
BSSC	0.280	2.2×10^3s	11.2μs	0.293	2.6×10^3s	13.4μs	0.112	3.2×10^3s	13.0μs	0.130	3.8×10^3s	15.1μs
PCAH	0.252	6.5s	1.2μs	0.235	7.4s	1.9μs	0.090	49.2s	2.8μs	0.075	52.3s	3.0μs
SpH	0.275	25.8s	28.3μs	0.284	88.2s	101.9μs	0.130	65.3s	40.2μs	0.148	131.1s	116.3μs
AGH	0.161	144.7s	55.7μs	0.267	184.2s	72.0μs	0.134	242.5s	83.7μs	0.160	279.4s	95.6μs
STH	0.270	1.2×10^3s	17.4μs	0.318	1.8×10^3s	19.8μs	0.123	1.9×10^3s	21.3μs	0.171	2.5×10^3s	25.2μ
KSH	0.324	2.1×10^3s	32.0μs	0.339	2.3×10^3s	37.4μs	0.136	2.6×10^3s	34.2μs	0.161	2.9×10^3s	38.1μs
CH	0.320	93.4s	53.5μs	0.360	98.2s	54.4μ	0.175	194s	64.1μs	0.206	210.5s	71.5μs
(Basic) ECE	**0.350 ±0.009**	5.5×10^4s	28.2μs	**0.412 ±0.014**	6.2×10^4s	31.7μs	**0.233 ±0.018**	6.1×10^4s	30.8μs	**0.249 ±0.017**	6.7×10^4s	35.0μs
RBPL-ECE	**0.335** ±**0.012**	917s	32.1μs	**0.387 ±0.020**	984s	33.5μs	**0.204 ±0.024**	1.7×10^3s	32.7μs	**0.213 ±0.021**	1.9×10^3s	33.9μs

The results of EBE and RBPL-EBE are mean accuracies of 10 runs with a degree of uncertainty

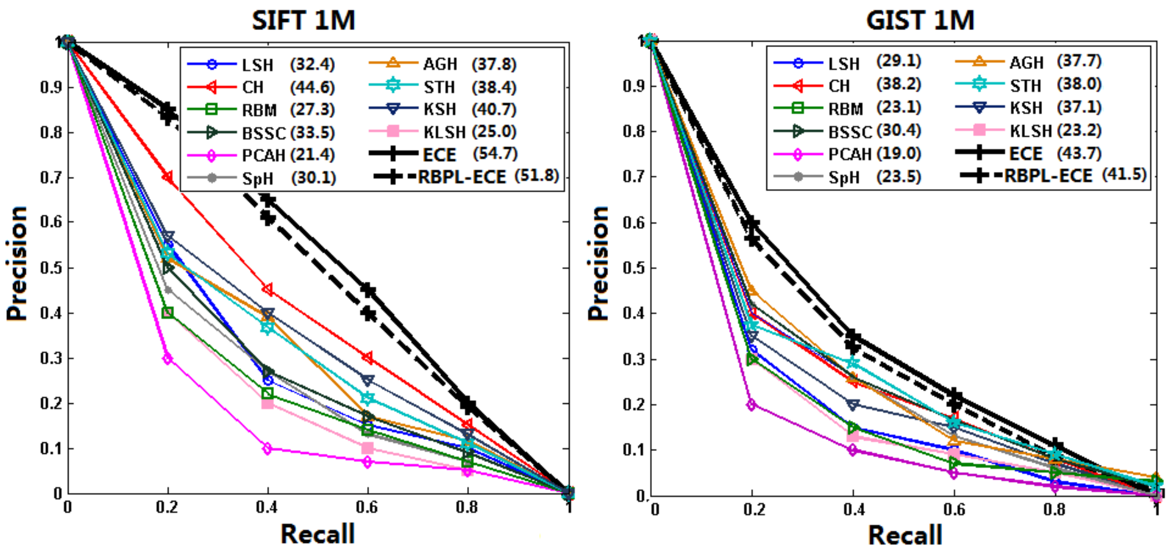


Fig. 6.8 The precision-recall curves of all algorithms on SIFT 1M and GIST 1M data sets for the codes of 48 bits. The AUC values are illustrated in the brackets.

the code length increases. Specifically, LSH and KLSH have a low MAP when the code length is short. STH and BSSC always produce competitive search accuracies on both two datasets. The performance of RBM and SpH achieves ‘rise-then-fall’ curves on the SIFT 1M dataset. Beyond those, it is obviously observed that KSH and CH always reach high search accuracies on both datasets. For our methods, both ECE and RBPL-ECE can significantly outperform the other comparable methods (also see in Table 6.9) in terms of the MAP. Fig. 6.8 also presents the precision-recall curves of all the algorithms on two data sets with the code of 48 bits. From both figures, we can further discover that, for both datasets, the (basic) ECE achieves slightly better performance than RBPL-ECE by comparing the Mean Average Precision (MAP) and Area Under the Curve (AUC). The learned GP-tree based hashing functions for embedding 16-bits binary codes on the SIFT 1M dataset are illustrated in Fig. 6.9.

We also list the training time and test time for different algorithms on two data sets in Table 6.9. Considering the training time, our (basic) ECE spends the most time to train. While, the random projection based algorithms are relatively efficient, especially the LSH. Compared with the (basic) ECE, the RBPL-ECE saves much computational cost in the training phase, which is also more efficient than RBM, BSSC, STH and KSH (details can be seen in Table 6.9). In terms of the test phase, LSH and PCAH are the most efficient methods. Both of them simply need a matrix multiplication and a thresholding to obtain the binary codes. Since the concatenation in our parallel coding, RBPL-ECE is slightly slower than the (basic) ECE. AGH and SpH are the most expensive methods for testing, due to the relatively high cost on calculating the sparse representation and computing the analytical eigenfunctions, respectively. The most expensive part of the (basic) ECE and the RBPL-ECE methods in coding is computing the algebraic weak functions generated by GP.

6.7 Summary

In this chapter, we have presented a novel framework to learn highly discriminative embedding codes for dimensionality reduction using evolutionary compact embedding (ECE). We address it as an optimization problem combining genetic programming (GP) with the boosting-based weight updating trick. For each bit of ECE, the proposed learning scheme evolves a binary classification function through GP and re-weights the training samples for the next bit to jointly minimize its empirical risk with the AdaBoost strategy. We further improve the basic ECE by using the random batch parallel learning (RBPL) technique, which has been demonstrated to be more efficient for large-scale training but can still achieve competitive results. We have systematically evaluated our method on the USPS, CMU PIE,

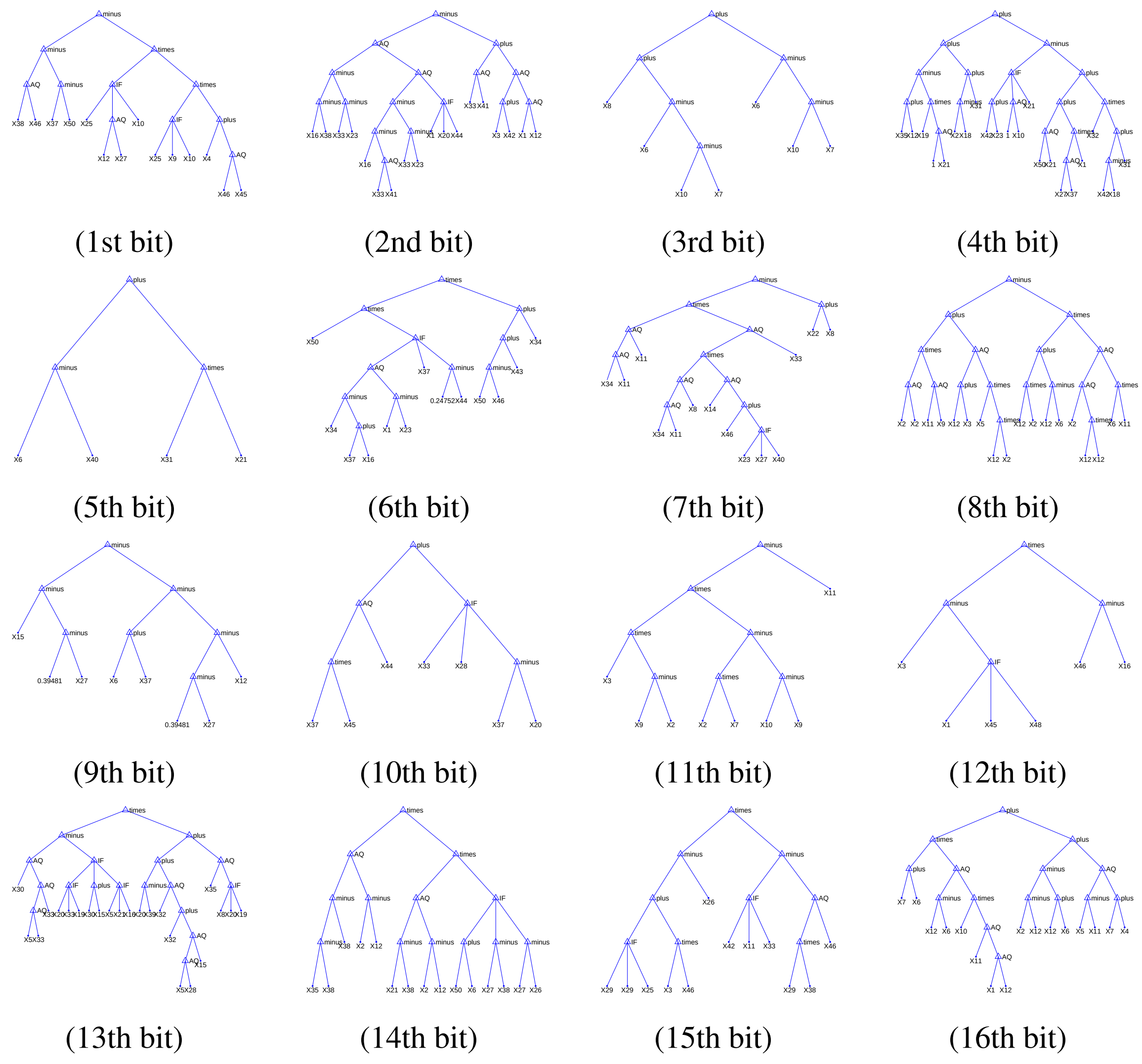


Fig. 6.9 The GP-evolved tree hash functions for embedding 16-bits binary codes on the SIFT 1M dataset. From top-left to bottom right, each tree illustrates a hashing function for a bit, respectively. (Note., the nodes 'plus', 'minus', 'times' and 'AQ' in the tree correspond with '$+$', '$-$', '$\times$' and '$\div$', respectively.)

CIFAR-10 and SUN397 scene datasets and produced the image classification accuracies of 96.1%, 98.15%, 86.4% and 18.8%, respectively. In all four datasets, our evolutionary compact embedding achieves better or competitive results compared with popular reduction methods. Furthermore, two standard datastes SIFT 1M and GIST 1M have also been evaluated for image retrieval, and show promising results compared with state-of-the-art hashing methods.

Although, the proposed method can successfully used for large-scale image retrieval, most existing hashing methods for image search are still based on global feature representations, which heavily limit the effectiveness of the hash code. In the next chapter, the Bilinear Local Feature Hashing (BLFH) has been proposed, which is a novel supervised hashing technique based on local features for image search and retrieval.

Chapter 7

Bilinear Local Feature Hashing

7.1 Overview

In the last chapter, a new evolutionary compact embedding (ECE) algorithm has been proposed for large-scale image similarity search. Although ECE can achieves significant results in terms of the final accuracies, this kind of hashing methods based on global feature representations may heavily limit the effectiveness of the hash code. Particularly, very low accuracies can be achieved for the final searching results if a large intra-class variation exists in the dataset. In this chapter, we propose the Bilinear Local Feature Hashing (BLFH), which is a novel supervised hashing technique based on local features for image search and retrieval. The proposed scheme aims to embed local feature descriptors from a high-dimensional space into a Hamming space with a low dimension, rather than using global features as in most existing hashing techniques which are susceptible to image variations such as viewpoint changes and background cluttering. Specifically, BLFH seeks two orthogonal projection matrices to preserve the pairwise attributes including labels and similarities between different local features. Meanwhile, a bipartite graph regularization item, which is constructed by images and classes, is simultaneously used to preserve the image-level intrinsic structure of local features. Since the raised problem is regarded as nonconvex and discrete, our objective function is then optimized via an alternate way with relaxation and finally converges to a near-optimal solution.

To the best of our knowledge, this is the first work specially on linear (bilinear) local feature hashing. It is worthwhile to highlight several properties of the proposed method:

(1) BLFH is linear and efficient. With the bilinear projection, the complexity of the eigen-decomposition, which is the cubic form of the dimensionality, will be significantly reduced.

(2) BLFH simultaneously preserves pairwise structure and bigraph structure which can

be regarded as the local structure and the global structure respectively in the original feature space.

(3) BLFH can be extended to the semi-supervised learning case by incorporating the optimization of unlabeled data, which is more practical for realistic applications.

7.2 Literature Review

An early work involving local feature binary coding was proposed in [174]. Specifically, they introduced two schemes to improve the standard bag-of-words (BoW) model: (1) a Hamming embedding (HE) which provides binary signatures to refine visual words; (2) a weak geometric consistency constraint with the geometrical transformation. Both methods can significantly improve the final performance for retrieval tasks. Furthermore, a coupled Multi-Index (c-MI) framework was proposed for accurate image retrieval [175]. In their framework, each keypoint in the image is described using both SIFT and color descriptors. Two distinct features are encoded by the binary coding methods in [174] and [176], respectively, and then fused together. Beyond that, a selective match kernel approach [177] has also been developed to incorporate matching kernels sharing the best properties of HE and VLAD. Based on [174], another related work can also be found in [178], which introduces a color binary descriptor and this descriptor can be achieved via either a global or a local form.

However, all the above BoW-related works mainly focus on the retrieval techniques rather than the learning procedure of the binary coding for large-scale hashing. Besides, these methods are not fully linear, which limits their efficiency and applicability for large-scale datasets.

7.3 Bilinear Local Feature Hashing

In this section, we present our new supervised hashing approach, Bilinear Local Feature Hashing (BLFH). Specifically, a bilinear projection scheme is first introduced to obtain the hashing function and then we explain how to construct our objective function via preserving the pairwise structure and the I2C distances. After that, we propose an alternate method to learn the optimal projections for hash codes. The outline of the proposed method is illustrated in Fig. 7.1.

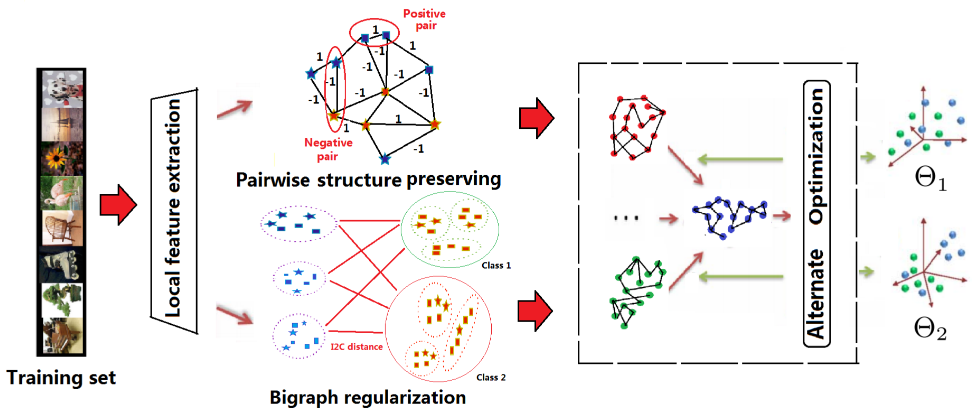


Fig. 7.1 The illustration of the working flow of BLFH learning. The algorithm intends to preserve the pairwise structure and the I2C distances and outputs the optimal bilinear projection matrices Θ_1 and Θ_2.

7.3.1 Bilinear Projection for Hashing

Bilinear projection is to multiply projection matrices on both sides of data. However, local features are mostly represented as a vector form which needs to be transformed into a matrix form. Given a local feature set $\mathscr{F} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ from training samples, where $\mathbf{x}_i \in \mathbb{R}^D$, $\forall i$, we factor integer D as $D = D_1 \times D_2$. Then we can rearrange $\mathbf{x}_i$ into a $D_1 \times D_2$ matrix X_i:

$$\mathbf{x}_i \in \mathbb{R}^D \mapsto X_i \in \mathbb{R}^{D_1 \times D_2},\ i = 1, \cdots, N.$$

In other words, we reorganize vector $\mathbf{x}_i$ into matrix X_i such that $\text{vec}(X_i) = \mathbf{x}_i$, where $\text{vec}(\cdot)$ represents the vectorization of a matrix. Then we have the inverse map of vectorization $\text{vec}^{-1}(\mathbf{x}_i) = X_i$, since the vectorization is a one-to-one correspondence if D_1 and D_2 are given. To make the transformation more efficient, we apply a bilinear formulation using two matrices $\Theta_1 \in \mathbb{R}^{D_1 \times d_1}$ and $\Theta_2 \in \mathbb{R}^{D_2 \times d_2}$ to define our hash function with the input of a vector form:

$$H(\mathbf{x}_i) = \text{sgn}\left(\text{vec}(\Theta_1^T \text{vec}^{-1}(\mathbf{x}_i)\Theta_2)\right), \tag{7.1}$$

or with the input of a matrix form:

$$H(X_i) = \text{sgn}\left(\text{vec}(\Theta_1^T X_i \Theta_2)\right). \tag{7.2}$$

In fact, we notice that

$$\text{vec}(\Theta_1^T X_i \Theta_2) = (\Theta_2^T \otimes \Theta_1^T)\text{vec}(X_i) = (\Theta_2^T \otimes \Theta_1^T)\mathbf{x}_i,$$

where $\otimes$ is the Kronecker product, thus a bilinear projection is simply a special case of the single matrix projection Θ which can be decomposed as $\Theta = \Theta_2 \otimes \Theta_1$. Besides, if Θ_1 and Θ_2 are orthogonal, i.e., $\Theta_1^T \Theta_1 = I_{d_1 \times d_1}$ and $\Theta_2^T \Theta_2 = I_{d_2 \times d_2}$, then Θ is orthogonal since $\Theta^T \Theta = (\Theta_2 \otimes \Theta_1)^T (\Theta_2 \otimes \Theta_1) = (\Theta_2^T \otimes \Theta_1^T)(\Theta_2 \otimes \Theta_1) = \Theta_2^T \Theta_2 \otimes \Theta_1^T \Theta_1 = I_{d_2 \times d_2} \otimes I_{d_1 \times d_1} = I_{d_1 d_2 \times d_1 d_2}$. One of the differences, however, is that bilinear projection leads to an eigen-decomposition on matrices with much smaller sizes $D_1 \times D_1$ and $D_2 \times D_2$ rather than $D_1 D_2 \times D_1 D_2$, which will be shown later.

7.3.2 Objective Function

To obtain meaningful hash code for local features, we construct our objective function mainly based on two parts: pairwise structure preserving and I2C distance preserving.

Pairwise Structure Preserving

Let us consider the whole property of the local feature set $\mathscr{F} = \{X_1, \cdots, X_N\}$, where N is the number of local features in training data. We are concerned about the relationship between each local feature in the high-dimensional space, which should also be retained in the lower-dimensional space. Accordingly, the pairwise label information is introduced instead of the class label. The binary property of hash function motivates us to employ the pairwise label $\{-1, +1\}$ to represent relationships between local features. However, similar local features can appear in samples from many different classes. Thus, for each pair of local features (X_i, X_j), its pairwise label is determined by its neighbors rather than the class label as follows:

$$\ell_{ij} = \begin{cases} +1, & X_i \in \mathrm{N}_k(X_j) \text{ or } X_j \in \mathrm{N}_k(X_i) \\ -1, & \text{otherwise} \end{cases},$$

where $\mathrm{N}_k(X)$ is the set of k nearest neighbors of X. We hope that the projected local features $\widehat{X}_i = \mathrm{vec}(\Theta_1^T X_i \Theta_2)$ and $\widehat{X}_j = \mathrm{vec}(\Theta_1^T X_j \Theta_2)$ have the same component-wise sign if they have pairwise label $+1$, and the different component-wise sign if they have pairwise label -1. We denote $\mathscr{P} = \{(i,j) | X_i, X_j \in \mathscr{F}\}$. Besides, to decrease the disturbance of noise, we assign a weight for each pair which is defined as follow:

$$W_{ij}^P = \exp\left(-\ell_{ij} \|X_i - X_j\|^2\right). \tag{7.3}$$

Therefore, the following function

$$
\begin{aligned}
&\sum_{(i,j)\in\mathscr{P}} W^P_{ij}\ell_{ij}\langle \mathrm{sgn}(\widehat{X}_i), \mathrm{sgn}(\widehat{X}_j)\rangle \\
=&\sum_{(i,j)\in\mathscr{P}} W^P_{ij}\ell_{ij}\mathrm{sgn}(\mathrm{vec}(\Theta_1^T X_i \Theta_2))^T \mathrm{sgn}(\mathrm{vec}(\Theta_1^T X_j \Theta_2)) \\
=&\sum_{(i,j)\in\mathscr{P}} W^P_{ij}\ell_{ij}\mathrm{tr}(\mathrm{sgn}(\Theta_1^T X_i \Theta_2)^T \mathrm{sgn}(\Theta_1^T X_j \Theta_2)) \\
=&\sum_{(i,j)\in\mathscr{P}} W^P_{ij}\ell_{ij}\mathrm{tr}(\mathrm{sgn}(\Theta_1^T X_i \Theta_2) \mathrm{sgn}(\Theta_1^T X_j \Theta_2)^T) \\
=&\sum_{(i,j)\in\mathscr{P}} W^P_{ij}\ell_{ij}\mathrm{tr}(\mathrm{sgn}(\Theta_1^T X_i \Theta_2) \mathrm{sgn}(\Theta_2^T X_j^T \Theta_1))
\end{aligned}
\tag{7.4}
$$

needs to be maximized, where $\langle\cdot,\cdot\rangle$ represents the inner product. The above function reaches its maximum value when $\ell_{ij}\mathrm{vec}(\Theta_1^T X_i \Theta_2)$ and $\mathrm{vec}(\Theta_1^T X_j \Theta_2)$ are similarly sorted due to the rearrangement inequality [179].

Bigraph Regularization

We are also concerned about a higher level connection, i.e., relationship between images and classes. Thus we consider a complete bipartite graph (a.k.a. bigraph) $G = (V_1, V_2, E)$ in which V_1 is the set of all images and V_2 is the set of all classes. The image-to-class (I2C) distance provides a feasible way to quantize the edges of E. Given the set of local features of an image $\mathscr{X}_i = \{X_{i1}, \cdots, X_{im_i}\}$, which contains all of the local features of image i, the I2C distance between image i and class c is defined as

$$
D^c_{\mathscr{X}_i} = \sum_{X\in\mathscr{X}_i} \|X - \mathrm{NN}^c(X)\|^2, \tag{7.5}
$$

where $\mathrm{NN}^c(X)$ is the nearest neighbor of the local feature X in class c and $\|\cdot\|$ is Frobenius norm.

However, searching the nearest neighbor in such a large scale space of local features of each class will still cost much time. Here, to reduce the complexity of searching, we first employ a K-means clustering algorithm on the set of local features of each class, i.e., $\bigcup_{C(\mathscr{X}_i)=c} \mathscr{X}_i$, $c = 1, \cdots, C$, where C is the number of classes and $C(\cdot) \in \{1, \cdots, C\}$ is the label information function that represents the class label of the input. And then we reduce the searching range of nearest neighbor to the cluster centers, i.e., for $c = 1, \cdots, C$, we let

$$
\mathrm{NN}^c(X) \in \text{Centroids } \{S_1, \cdots, S_K\} \text{ of } \bigcup_{C(\mathscr{X}_i)=c} \mathscr{X}_i.
$$

Via I2C distances, we construct the bigraph $G = (V_1, V_2, E)$, where V_1 and V_2 represent the set of all the images and the set of all the classes, respectively. Then for each edge in E, we define its weight W_i^c, named the I2C similarity between image i and class c in the original space, by the following Gaussian function

$$W_{ic}^D = \exp\left(\frac{-(D_{\mathscr{X}_i}^c)^2}{2\sigma^2}\right), \; i = 1, \cdots, n, \; c = 1, \cdots, C, \tag{7.6}$$

where σ is the Gaussian smooth parameter and n is the number of training samples. After applying BLFH, we have the I2C distance in Hamming space

$$\widehat{D}_{\mathscr{X}_i}^c = \sum_{X \in \mathscr{X}_i} \|\mathrm{H}(X) - \mathrm{NN}^c(\mathrm{H}(X))\|^2. \tag{7.7}$$

In order to preserve the order of projected similarity in the projected space, a reasonable objective function for bigraph regularization is to minimize

$$\sum_{i=1}^{n} \sum_{c=1}^{C} \widehat{D}_{\mathscr{X}_i}^c \cdot W_{ic}^D. \tag{7.8}$$

One of the necessary conditions of the above function reaches the minimum value is that $\{\widehat{D}_{\mathscr{X}_i}^c\}$ is order-preserved due to the rearrangement inequality [179].

In addition, to make the projected space more compact, we set orthogonality constraints on the projection matrices Θ_1 and Θ_2, i.e., $\Theta_1^T \Theta_1 = I$ and $\Theta_2^T \Theta_2 = I$. Combined with the pairwise preserving part in Eq. (7.4) and orthogonality constraints, finally, we set our optimization problem as follows:

$$\begin{aligned} \underset{\substack{\Theta_1^T\Theta_1 = I \\ \Theta_2^T\Theta_2 = I}}{\arg\max} & \sum_{(i,j) \in \mathscr{P}} W_{ij}^P \ell_{ij} \langle \mathrm{sgn}(\widehat{X}_i), \mathrm{sgn}(\widehat{X}_j) \rangle \\ & - \gamma \sum_{i=1}^{n} \sum_{c=1}^{C} \widehat{D}_{\mathscr{X}_i}^c \cdot W_{ic}^D. \end{aligned} \tag{7.9}$$

where γ is the regularization parameter.

7.3.3 Alternate Optimization via Relaxation

In this section, to gain a more optimal solution, we first relax the discrete sign function in optimization problem (7.9) to a real-valued continuous function by using its signed magnitude, i.e., $\mathrm{sgn}(x) \approx x$. In this case, the pairwise structure preserving part, i.e., Eq. (7.4) becomes

$$\sum_{(i,j)\in\mathscr{P}} W_{ij}^{P}\ell_{ij}\,\mathrm{tr}(\Theta_1^T X_i \Theta_2 \Theta_2^T X_j^T \Theta_1). \tag{7.10}$$

For simplicity, we denote $\mathrm{NN}^c(X) = X^c$. Besides, we also make a statistical approximation on the computation of projected I2C distances due to the large amount of local features. That is, we exchange the operation of NN^c and $H(\cdot)$ for all $X \in \mathscr{X}_i$, i.e., $\sum_{X\in\mathscr{X}_i}\|X - H(X)^c\|^2 \approx \sum_{X\in\mathscr{X}_i}\|X - H(X^c)\|^2$. Thus, the projected I2C distance after applying matrices Θ_1 and Θ_2 becomes

$$\begin{aligned}
\widehat{D}^c_{\mathscr{X}_i} &\approx \sum_{X\in\mathscr{X}_i}\|\Theta_1^T X\Theta_2 - \Theta_1^T X^c\Theta_2\|^2\\
&= \sum_{X\in\mathscr{X}_i}\|\Theta_1^T (X - X^c)\Theta_2\|^2\\
&= \sum_{k=1}^{m_i}\mathrm{tr}(\Theta_1^T (X_{ik} - X_{ik}^c)\Theta_2(\Theta_1^T (X_{ik} - X_{ik}^c)\Theta_2)^T)\\
&= \sum_{k=1}^{m_i}\mathrm{tr}(\Theta_1^T (X_{ik} - X_{ik}^c)\Theta_2\Theta_2^T (X_{ik} - X_{ik}^c)^T\Theta_1)\\
&:= \sum_{k=1}^{m_i}\mathrm{tr}(\Theta_1^T \Delta X_{ik}^c\Theta_2\Theta_2^T (\Delta X_{ik}^c)^T\Theta_1),
\end{aligned} \tag{7.11}$$

where $\Delta X_{ik}^c = X_{ik} - X_{ik}^c$, $k = 1,\cdots,m_i$.

Additionally, due to the difficulty of computing the optimal Θ_1 and Θ_2 simultaneously, we derive an alternate iteration algorithm in this section. Specifically, for a fixed Θ_2, we can compute the optimal Θ_1 by solving a classical eigen-decomposition problem. And for the computation of Θ_2, we can then update Θ_2 by solving another eigen-decomposition problem with the computed Θ_1. First, let us denote the objective function in optimization problem (7.9) by $\mathscr{L}(\Theta_1,\Theta_2)$ and transform it to the following form by Eq. (7.10) and (7.11)

$$\begin{aligned}
&\mathscr{L}(\Theta_1,\Theta_2)\\
&= \sum_{(i,j)\in\mathscr{P}} W_{ij}^{P}\ell_{ij}\,\mathrm{tr}(\Theta_1^T X_i\Theta_2\Theta_2^T X_j^T\Theta_1)\\
&\quad -\gamma\sum_{i=1}^{n}\sum_{c=1}^{C}\sum_{k=1}^{m_i} W_{ic}^{D}\,\mathrm{tr}(\Theta_1^T\Delta X_{ik}^c\Theta_2\Theta_2^T(\Delta X_{ik}^c)^T\Theta_1)\\
&:= \mathrm{tr}(\Theta_1^T M_2(\Theta_2)\Theta_1),
\end{aligned} \tag{7.12}$$

where

$$\begin{aligned} M_2(\Theta_2) = & \sum_{(i,j)\in\mathscr{P}} W_{ij}^P \ell_{ij} X_i \Theta_2 \Theta_2^T X_j^T \\ & -\gamma \sum_{i=1}^{n} \sum_{c=1}^{C} \sum_{k=1}^{m_i} W_{ic}^D \Delta X_{ik}^c \Theta_2 \Theta_2^T (\Delta X_{ik}^c)^T. \end{aligned} \tag{7.13}$$

And we also have the following form

$$\begin{aligned} & \mathscr{L}(\Theta_1, \Theta_2) \\ = & \sum_{(i,j)\in\mathscr{P}} W_{ij}^P \ell_{ij} \operatorname{tr}(\Theta_2^T X_j^T \Theta_1 \Theta_1^T X_i \Theta_2) \\ & -\gamma \sum_{i=1}^{n} \sum_{c=1}^{C} \sum_{k=1}^{m_i} W_{ic}^D \operatorname{tr}(\Theta_2^T (\Delta X_{ik}^c)^T \Theta_1 \Theta_1^T \Delta X_{ik}^c \Theta_2) \\ := & \operatorname{tr}(\Theta_2^T M_1(\Theta_1) \Theta_2), \end{aligned} \tag{7.14}$$

since $\operatorname{tr}(AB) = \operatorname{tr}(BA)$ if both products AB and BA exist, where

$$\begin{aligned} M_1(\Theta_1) = & \sum_{(i,j)\in\mathscr{P}} W_{ij}^P \ell_{ij} X_j^T \Theta_1 \Theta_1^T X_i \\ & -\gamma \sum_{i=1}^{n} \sum_{c=1}^{C} \sum_{k=1}^{m_i} W_{ic}^D (\Delta X_{ik}^c)^T \Theta_1 \Theta_1^T \Delta X_{ik}^c. \end{aligned} \tag{7.15}$$

Then M_1 and M_2 are two matrix-valued functions with their codomains $\mathbb{R}^{D_1 \times D_1}$ and $\mathbb{R}^{D_2 \times D_2}$, respectively. Although the number of the local features is relatively huge, the size of our final matrices M_1 and M_2 used for decomposition are small enough (D_1 and D_2 are always less than 100). This property mainly guarantees the efficiency and feasibility. Therefore, for $t = 0$, we randomly initialize $\Theta_2^{(t)}$; for t-th step, we have the update rules:

$$\begin{aligned} \Theta_1^{(t)} & \leftarrow \text{ the first } d_1 \text{ eigenvectors of } M_2(\Theta_2^{(t-1)}); \\ \Theta_2^{(t)} & \leftarrow \text{ the first } d_2 \text{ eigenvectors of } M_1(\Theta_1^{(t)}). \end{aligned}$$

For the t-th step, we have the following inequality

$$\mathscr{L}(\Theta_1^{(t-1)}, \Theta_2^{(t-1)}) \leq \mathscr{L}(\Theta_1^{(t)}, \Theta_2^{(t-1)}) \leq \mathscr{L}(\Theta_1^{(t)}, \Theta_2^{(t)}).$$

Thus $\mathscr{L}(\Theta_1^{(t)}, \Theta_2^{(t)})$ is monotonic nondecreasing as $t \to \infty$. And continuous function $\mathscr{L}(\Theta_1, \Theta_2)$ is bounded in the closed district $\{(\Theta_1, \Theta_2) | \Theta_1^T \Theta_1 = I, \Theta_2^T \Theta_2 = I\}$. Then the above alternate iteration converges. In practice, we stop the iteration when the difference $|\mathscr{L}(\Theta_1^{(t)}, \Theta_2^{(t)}) -$

$\mathscr{L}(\Theta_1^{(t-1)},\Theta_2^{(t-1)})|$ is less than a small threshold or the number of iteration reaches a maximum. The corresponding integrated BLFH algorithm is depicted in Algorithm 5.

Algorithm 5 Bilinear Local Feature Hashing (BLFH)

```
Input: Local feature set of each training image $\mathscr{X}_i = \{X_{i1}, \cdots, X_{im_i}\}$ in matrix form, the
    whole local feature set $\mathscr{F} = \bigcup \mathscr{X}_i$, the parameter $k$ for pairwise structure preserving, the
    number of centroids $K$ in K-means and the label information function $C(\cdot) \in \{1, \cdots, C\}$.
Output: The bilinear projection matrices $\Theta_1$ and $\Theta_2$.
1: Construct local feature pairing set $\mathscr{P} = \{(i,j)|X_i, X_j \in \mathscr{F}\}$ and their corresponding
   pairwise labels $\ell_{ij} = \{-1,+1\}$, where $\ell_{ij} = +1$ if $X_i \in \mathrm{N}_k(X_j)$ or $X_j \in \mathrm{N}_k(X_i)$, and
   $\ell_{ij} = -1$ otherwise;
2: Employ the K-means clustering algorithm on the set of local features of each class
   $\bigcup_{C(\mathscr{X}_i)=c} \mathscr{X}_i$, $c = 1, \cdots, C$;
3: Compute pairwise weight $W_{ij}^P$ and I2C similarity $W_{ic}^D$ by Eqs. (7.3) and (7.6) respec-
   tively;
4: Initialize $\Theta_2^{(0)}$ randomly;
5: repeat
6:     $\Theta_1^{(t)} \leftarrow$ the first $d_1$ eigenvectors of $M_2(\Theta_2^{(t-1)})$ by Eq. (7.13);
7:     $\Theta_2^{(t)} \leftarrow$ the first $d_2$ eigenvectors of $M_1(\Theta_1^{(t)})$ by Eq. (7.15);
8: until $\mathscr{L}(\Theta_1^{(t)},\Theta_2^{(t)})$ converges.
```

7.4 Semi-supervised BLFH

For more realistic scenarios, it is difficult for all the samples in the training set to be well labeled in some visual retrieval tasks. Therefore, in this section, we consider our algorithm in a new environment by taking unlabeled samples into account and incorporating them with labeled samples to a whole optimization procedure as semi-supervised BLFH (SBLFH). The unlabeled samples are attached to the original labeled sample set as $\{\mathscr{X}_1, \cdots, \mathscr{X}_n, \mathscr{X}_{n+1}, \cdots, \mathscr{X}_{n+n_U}\}$, where the first n samples are labeled and the left n_U ones are unlabeled. Then the pairwise label is determined by the k nearest neighbors of each local feature instead of class labels, i.e.,

$$\ell_{ij} = \begin{cases} +1, & X_i \in \mathrm{N}_k(X_j) \text{ or } X_j \in \mathrm{N}_k(X_i) \\ -1, & \text{otherwise} \end{cases},$$

where $\mathrm{N}_k(X)$ is the set of k nearest neighbors of X.

Besides, the I2C distance cannot be applied in unlabeled samples. Thus, similarly, we introduce image-to-image (I2I) distance for all the samples including labeled and unlabeled ones. I2I distance from image i to image j simply changes the range of nearest neighbor search to the local features in $\mathscr{X}_j$ instead of the whole class in I2C distance, which is formulated as

$$d_{ij} = \sum_{X \in \mathscr{X}_i} \|X - \mathrm{NN}^{\mathscr{X}_j}(X)\|^2, \tag{7.16}$$

where $\mathrm{NN}^{\mathscr{X}_j}(X)$ represents the nearest neighbor in image j of local feature X. However, it is easy to find that $d_{ij} \neq d_{ji}$. Thus, we define our symmetric I2I distance as their average

$$D_{ij} = \frac{1}{2}(d_{ij} + d_{ji}). \tag{7.17}$$

Then in the projected Hamming space, I2I distance becomes

$$\begin{aligned} \widehat{D}_{ij} = \frac{1}{2}\Bigg(& \sum_{X \in \mathscr{X}_i} \|\mathrm{H}(X) - \mathrm{NN}^{\mathscr{X}_j}(\mathrm{H}(X))\|^2 \\ & + \sum_{X \in \mathscr{X}_j} \|\mathrm{H}(X) - \mathrm{NN}^{\mathscr{X}_i}(\mathrm{H}(X))\|^2 \Bigg). \end{aligned} \tag{7.18}$$

And we also have the following I2I similarity between image i and image k in the original space

$$W_{ij}^{I} = \exp\left(\frac{-D_{ij}^2}{2\tau^2}\right), \; i,j = 1,\cdots,n+n_U, \tag{7.19}$$

where τ is the Gaussian smooth parameter. Therefore, the following function

$$\sum_{i=1}^{n+n_U} \sum_{j=1}^{n+n_U} \widehat{D}_{ij} \cdot W_{ij}^{I} \tag{7.20}$$

need to be minimized since when it reaches the minimum value, $\{\widehat{D}_{ij}\}$ and $\{W_{ij}^{I}\}$ are oppositely sorted, i.e., $\{\widehat{D}_{ij}\}$ and $\{D_{ij}\}$ are similarly sorted due to the rearrangement inequality [179]. Finally, combined with the pairing preserving part and orthogonal constraints, we set the semi-supervised optimization problem as follows

$$\begin{aligned}\underset{\substack{\Theta_1^T\Theta_1=I\\ \Theta_2^T\Theta_2=I}}{\arg\max} \sum_{(i,j)\in\mathscr{P}} & W_{ij}^P \ell_{ij}\langle \operatorname{sgn}(\widehat{X}_i), \operatorname{sgn}(\widehat{X}_j)\rangle \\ & -\gamma\lambda \sum_{i=1}^{n}\sum_{c=1}^{C} \widehat{D}_{\mathscr{X}_i}^c \cdot W_{ic}^D \\ & -\gamma(1-\lambda)\sum_{i=1}^{n+n_U}\sum_{j=1}^{n+n_U} \widehat{D}_{ij}\cdot W_{ij}^I. \end{aligned} \tag{7.21}$$

where γ is the regularization parameter and $\lambda \in (0,1)$ is the balanced parameter.

7.4.1 Optimizing SBLFH

Similarly, we also have relaxation $\operatorname{sgn}(x) \approx x$ in the optimization. If we denote $X_k^{ij} = X_{ik} - \mathrm{NN}^{\mathscr{X}_j}(X_{ik})$, $i, j = 1, \cdots, n+n_U$, $k = 1, \cdots, m_i$, similar to Eq. (7.11), we have the projected I2I distance

$$\begin{aligned}\widehat{D}_{ij} \approx \frac{1}{2}\Bigg(& \sum_{k=1}^{m_i} \operatorname{tr}(\Theta_1^T X_k^{ij}\Theta_2\Theta_2^T (X_k^{ij})^T\Theta_1) \\ & + \sum_{k=1}^{m_j} \operatorname{tr}(\Theta_1^T X_k^{ji}\Theta_2\Theta_2^T (X_k^{ji})^T\Theta_1)\Bigg).\end{aligned}$$

And we also need an alternate optimization for the semi-supervised algorithm. We use $\mathscr{L}^{semi}(\Theta_1, \Theta_2)$ to denote the objective function in optimization problem (7.21). By the similar derivation in Eqs. (7.12)–(7.15), we also have form

$$\mathscr{L}^{semi}(\Theta_1,\Theta_2) = \operatorname{tr}(\Theta_1^T M_2'(\Theta_2)\Theta_1) = \operatorname{tr}(\Theta_2^T M_1'(\Theta_1)\Theta_2),$$

where

$$\begin{aligned}
&M_2'(\Theta_2)\\
&= \sum_{(i,j)\in\mathscr{P}} W_{ij}^P \ell_{ij} X_i \Theta_2 \Theta_2^T X_j^T\\
&- \gamma\lambda \sum_{i=1}^{n}\sum_{c=1}^{C}\sum_{k=1}^{m_i} W_{ic}^D \Delta X_{ik}^c \Theta_2 \Theta_2^T (\Delta X_{ik}^c)^T\\
&- \frac{\gamma}{2}(1-\lambda) \sum_{i=1}^{n+n_U}\sum_{j=1}^{n+n_U} W_{ij}^I \sum_{k=1}^{m_i} X_k^{ij} \Theta_2 \Theta_2^T (X_k^{ij})^T\\
&- \frac{\gamma}{2}(1-\lambda) \sum_{i=1}^{n+n_U}\sum_{j=1}^{n+n_U} W_{ji}^I \sum_{k=1}^{m_j} X_k^{ji} \Theta_2 \Theta_2^T (X_k^{ji})^T
\end{aligned} \tag{7.22}$$

and

$$\begin{aligned}
&M_1'(\Theta_1)\\
&= \sum_{(i,j)\in\mathscr{P}} W_{ij}^P \ell_{ij} X_j^T \Theta_1 \Theta_1^T X_i\\
&- \gamma\lambda \sum_{i=1}^{n}\sum_{c=1}^{C}\sum_{k=1}^{m_i} W_{ic}^D (\Delta X_{ik}^c)^T \Theta_1 \Theta_1^T \Delta X_{ik}^c\\
&- \frac{\gamma}{2}(1-\lambda) \sum_{i=1}^{n+n_U}\sum_{j=1}^{n+n_U} W_{ij}^I \sum_{k=1}^{m_i} (X_k^{ij})^T \Theta_1 \Theta_1^T X_k^{ij}\\
&- \frac{\gamma}{2}(1-\lambda) \sum_{i=1}^{n+n_U}\sum_{j=1}^{n+n_U} W_{ji}^I \sum_{k=1}^{m_j} (X_k^{ji})^T \Theta_1 \Theta_1^T X_k^{ji}
\end{aligned} \tag{7.23}$$

are two matrix-valued functions with their codomains $\mathbb{R}^{D_1\times D_1}$ and $\mathbb{R}^{D_2\times D_2}$ respectively. Therefore, for $t = 0$, we randomly initialize $\Theta_2^{(t)}$; for t-th step, we also have the update rules:

$$\begin{aligned}
\Theta_1^{(t)} &\leftarrow \text{the first } d_1 \text{ eigenvectors of } M_2'(\Theta_2^{(t-1)});\\
\Theta_2^{(t)} &\leftarrow \text{the first } d_2 \text{ eigenvectors of } M_1'(\Theta_1^{(t)}).
\end{aligned}$$

The iteration halts when the difference $|\mathscr{L}^{semi}(\Theta_1^{(t)},\Theta_2^{(t)}) - \mathscr{L}^{semi}(\Theta_1^{(t-1)},\Theta_2^{(t-1)})|$ is less than a small threshold or the number of iteration reaches a maximum. We summarize the integrated SBLFH algorithm in Algorithm 6 as follows.

Algorithm 6 Semi-supervised BLFH (SBLFH)

```
Input: Local feature sets of each training labeled image $\{\mathscr{X}_1, \cdots, \mathscr{X}_n\}$ and unlabeled image $\{\mathscr{X}_{n+1}, \cdots, \mathscr{X}_{n+n_U}\}$ in matrix form, the whole local feature set $\mathscr{F} = \bigcup \mathscr{X}_i$, the parameter $k$ for pairwise structure preserving, the number of centroids $K$ in K-means, the label information function $C(\cdot) \in \{1, \cdots, C\}$ and the balanced parameter $\lambda$.
Output: The bilinear projection matrices $\Theta_1$ and $\Theta_2$.
1: Construct local feature pairing set $\mathscr{P} = \{(i,j)|X_i, X_j \in \mathscr{F}\}$ and their corresponding pairwise labels $\ell_{ij} = \{-1, +1\}$, where $\ell_{ij} = +1$ if $X_i \in \mathrm{N}_k(X_j)$ or $X_j \in \mathrm{N}_k(X_i)$, and $\ell_{ij} = -1$ otherwise;
2: Employ the K-means clustering algorithm on the set of local features of each class $\bigcup_{C(\mathscr{X}_i)=c} \mathscr{X}_i$, $c = 1, \cdots, C$;
3: Compute pairwise weight $W^P_{ij}$, I2C similarity $W^D_{ic}$ and I2I similarity $W^I_{ij}$ by Eqs. (7.3), (7.6) and (7.19) respectively;
4: Initialize $\Theta_2^{(0)}$ randomly;
5: repeat
6:     $\Theta_1^{(t)} \leftarrow$ the first $d_1$ eigenvectors of $M'_2(\Theta_2^{(t-1)})$ by Eq. (7.22);
7:     $\Theta_2^{(t)} \leftarrow$ the first $d_2$ eigenvectors of $M'_1(\Theta_1^{(t)})$ by Eq. (7.23);
8: until $\mathscr{L}^{semi}(\Theta_1^{(t)}, \Theta_2^{(t)})$ converges.
```

7.5 Indexing via Local Hashing Voting

Once the bilinear projection matrices $\{\Theta_1, \Theta_2\}$ are computed, we can easily embed the training data into binary hash codes by Eq. (7.2). Generally, for global hashing, a common way is to find the similar samples in the training set by using *Hamming Distance Ranking* (HDR), which returns the nearest neighbors of query in Hamming space and is proved to be efficient enough with short hash codes in practice. However, for our local feature hashing scenario, linear search (e.g., HDR) with complexity $O(N)$ is not fast any more, since N denotes the total number[1] of the local features. To achieve the local feature based visual retrieval, in this chapter, we apply a fast indexing scheme called Local Hashing Voting (LHV) as shown in Fig. 7.2. We first introduce the *Hamming lookup table* (a.k.a. the hash table) into our LHV scheme. Hamming lookup table, which is in principle similar with inverted index, builds a series of buckets containing the indices of the documents with the same hash code. Given a query, we can find the bucket of corresponding hash codes in near constant time $O(1)$, and return all the data in the bucket as the retrieval results. Nevertheless, as many hash bits with only one single lookup table used, the Hamming space becomes increasingly sparse; and very few samples fall in the same hash code bucket of

[1]For large-scale database retrieval, the total number of local features is always huge. In practice, for a training set of $10K$ images, if each image contains 300 local features, the total number N would be $300 \times 10K = 3M$.

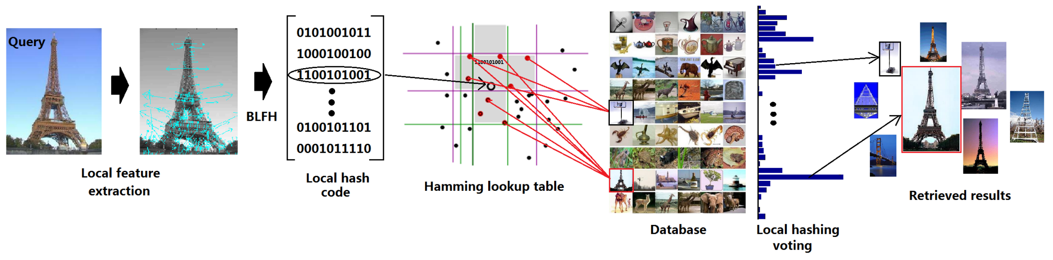


Fig. 7.2 An illustration of the proposed LHV. Specifically, given a query image, local features are first extracted and embedded into hash codes via BLFH. Then, each hash code (e.g., “1100101001”) corresponding to a local feature in the query image is then searched in the Hamming lookup table within the Hamming radius r and this gives the corresponding images' indices. Finally, we vote and accumulate the times of each image's index appearing in relevant buckets and rank them to return the retrieved results.

a query feature. Thus, in this work, we return all the nearest samples located within the Hamming radius r as the retrieval results, where r is always less than 3.

After construction of the Hamming lookup table over the training set, we store the corresponding indices for all the hash codes of local features. For instance, given a bucket with hash code “1100101001”, we store the indices of the images, which contain the same local feature hash code with this bucket. In this way, we search the hash code $\mathrm{H}(\mathbf{q}_i)$ for each of the local feature $\mathbf{q}_k \in \mathscr{Q}$ in the query image $\mathscr{Q} = \{\mathbf{q}_1, \cdots, \mathbf{q}_m\}$ over the Hamming lookup table within Hamming radius r and return the possible images' indices. It is noteworthy that the same bucket in the Hamming lookup table may store the indices from different images. Furthermore, we vote and accumulate the times of each image's index appearing in relevant buckets and then rank them in decreasing order. LHV cannot be used individually but can only be combined with the local hashing methods. The final retrieved samples are returned according to the relevant ranking generated by our LHV scheme, which is depicted in Algorithm 7.

7.6 Complexity Analysis

The cost of BLFH mainly contains three parts. The first part is for constructing the weight matrix (W_{ic}^D) to preserve the I2C distance. The time complexity of this part is $O(nCNKD)$. The second part is for calculating the bilinear projection matrices via alternate optimization. From Section 7.3.3, the update of Θ_1 has the time complexity of the eigenvalue decomposition of an $D_1 \times D_1$ matrix. It is $O(D_1^3)$. The update of Θ_2 has the similar time complexity of $O(D_2^3)$. The last part is for image indexing via the proposed Local Hashing Voting (LHV). For the hash code of each local feature, the index complexity is $O(1)$

Algorithm 7 Local Hashing Voting (LHV)

```
Input: The local feature set $\mathscr{F} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ in vector form from all training images, a local feature set of query image $\mathscr{Q} = \{\mathbf{q}_1, \cdots, \mathbf{q}_m\}$, where $\mathbf{x}_i$, $\mathbf{q}_i \in \mathbb{R}^D$, $\forall i$, Hamming radius $r$ and the learned bilinear projection matrices $\{\Theta_1, \Theta_2\}$.
Output: The retrieved images ranked by similarity with the query.
1: Embedding all the local features in $\mathscr{F}$ and $\mathscr{Q}$ into Hamming space via Eq. (7.1) with $\{\Theta_1, \Theta_2\}$;
2: Construct Hamming lookup table over the training set;
3: for $i = 1$ to $m$ do
4:     For the local query hash code $\mathrm{H}(\mathbf{q}_i)$, store all the possible image indices that fall into the Hamming lookup table within Hamming radius $r$;
5: end for
6: Vote and accumulate the times of each image's index appearing and rank them in decreasing order; return All the relevant images as the retrieved results.
```

Table 7.1 Computational complexity of BLFH and SBLFH.

Procedures	**BLFH**	**SBLFH**
I2C distance with centroids	$O(nCNKD)$	$O(nCNKD)$
I2I distance	—	$O\big((n+n_U)^2 \sum_{i=1}^{n+n_U} m_i^2 D\big)$
Alternate optimization	$O(D_1^3 + D_2^3)N_T$	$O(D_1^3 + D_2^3)N_T$
Overall (training)	$O(N) + O(D^{\frac{3}{2}})$	$O(N^2) + O(D^{\frac{3}{2}})$
LHV	$O(m)$	$O(m)$

using the Hamming lookup table. Thus, it costs $O(m)$ for a query with m local features in the search phase according to Section 7.5. In total, the time complexity of BLFH is at most $O(nCNKD) + (O(D_1^3) + O(D_2^3)) \times N_T + O(m)$, where N_T is the number of the iteration for alternate optimization. While, for SBLFH the total time complexity can be denoted as $O(nCNKD) + O((n+n_U)^2 \sum_{i=1}^{n+n_U} m_i^2 D) + (O(D_1^3) + O(D_2^3)) \times N_T + O(m)$, where $O((n+n_U)^2 \sum_{i=1}^{n+n_U} m_i^2 D)$ indicates the complexity of the I2I item in Eq. (7.21). Empirically, D_1 and D_2 are approximately equal to $D^{1/2}$. N_T is always less than 10, i.e., BLFH/SBLFH converges within 10 rounds.

7.7 Experiments and Results

In this section, we evaluate the proposed method on image classification and searching tasks respectively. The relevant results are also compared with the state-of-the-art methods.

7.7.1 Experiments on Efficient Image Classification

We first evaluate our binary codes for image classification tasks. A standard static object dataset Caltech-256 [180] has been used. We use the naive Bayes nearest neighbor (NBN-N) classifier to achieve our classification with a varied number of training sample for each object category, i.e., $\{5, 10, 15, 20, 25, 30\}$. Particularly, we compare the classification accuracies between using the original 128-d SIFT feature and the low-dimensional binary codes via the proposed BLFH. Fig. 7.3 shows the comparison results in terms of the image classification performance on the Caltech-256 dataset. We compute the accuracies on three different lengths, i.e., $\{16, 32, 64\}$, using our BLFH embedding method. The results manifest that combining the SIFT and the proposed BLFH can achieve better classification performance than just using the original SIFT feature on lengths of 32 and 64. When the length of codes decreases to 16, SIFT+BLFH gets less precision than the original SIFT, since the discriminative information cannot be preserved well enough in the binary low-dimensional space. Apparently, due to consideration of the pairwise data structure with bipartite graph regularization, the proposed BLFH can significantly improve the classification accuracy with reduced dimensionality of data representations.

Different from the original NBNN algorithm, in our experiments, the Hamming distance is used in NBNN when applying the BLFH. Obviously, SIFT+BLFH costs much less time than only applying the original SIFT. Because computing the Hamming distance (X-OR operation) in NBNN is much faster than using the Euclidean distance. Therefore, we can conclude that in image classification tasks, the proposed BLFH can effectively achieve higher performance while costing far less time.

7.7.2 Experiments on Image Searching

In this section, the proposed BLFH algorithm and its semi-supervised version SBLFH are evaluated for the image similarity search problem. Three different datasets are used in our experiments, i.e., Caltech-256 [180], SUN397 [136] and NUS-WIDE [181]. The **Caltech-256** dataset consists of 30607 images associated with 256 object categories. By following the experimental setting in [182], we randomly select 1000 images as the query set and the rest of dataset is regarded as the training set. The **SUN397** dataset contains $108,754$ scene

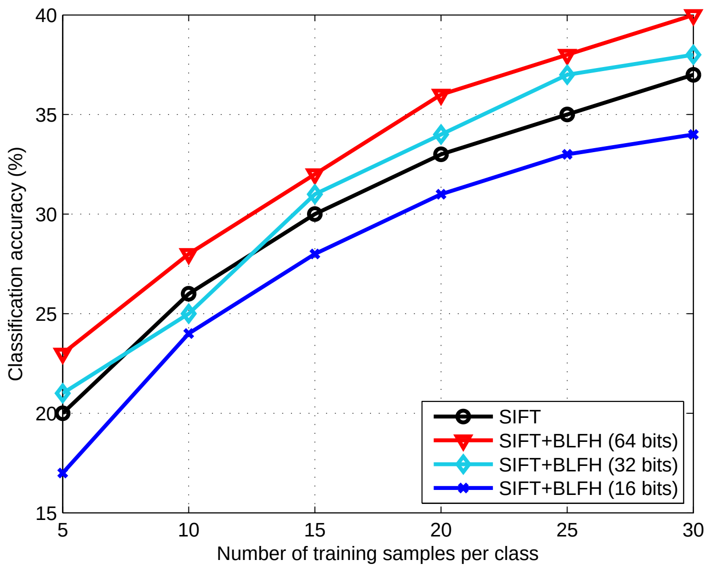
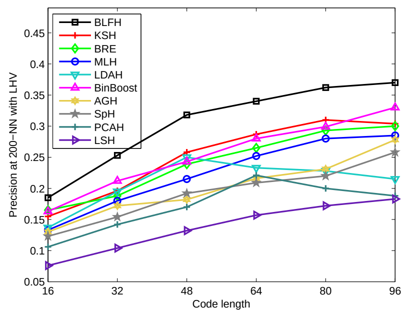


Fig. 7.3 Performance comparison of NBNN on the Caltech256 dataset.

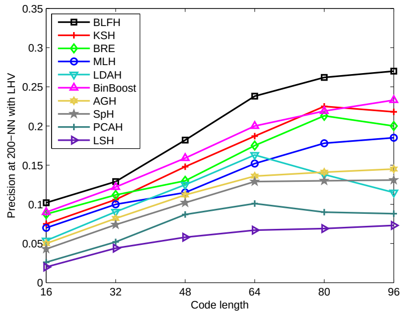
images in total from 397 well-sampled categories with at least 100 images per category. We randomly select 70 samples from each category to construct the training set and the rest of samples are the query set. Thus, there are total numbers of 27790 and 80964 in the training set and query set, respectively. For the **NUS-WIDE** dataset, it contains around $270,000$ web images associated with 81 ground truth concept classes. As in [70], we only use the most frequent 21 concept classes, each of which has abundant relevant images ranging from $5,000$ to $30,000$. Unlike other datasets, each image in the NUS-WIDE dataset is assigned with multiple semantic labels (tags). In this work, two images belong to the same class, only if they share at least one common tag. We further sample uniformly 100 images from each of the selected 21 tags to form a query set of $2,100$ images with the rest serving as the training set. Furthermore, given an image, we would like to describe it with a set of local features extracted from it. In our experiments, we adopt SIFT[2] [8] as the local feature to describe the images and then learn to hash these local descriptors with all compared methods.

In the querying phase, a returned point is regarded as a true neighbor if it lies in the top ranked 200, 200 and 500 points in LHV for Caltech-256, SUN397 and NUS-WIDE, respectively. Specifically in LHV, we just consider the local hash codes lying in the buckets that fall within a small Hamming radius $r = 2$ (following [69]) in the Hamming lookup table which is constructed using the training set codes. We evaluate the retrieval results by the

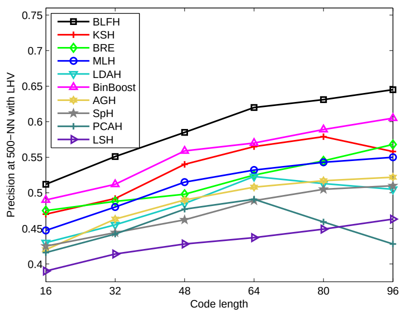
[2] A 16×16 local patch is divided into sub-blocks by a 4×4 grid, and in each sub-block a histogram of 8 orientation bins is computed. Thus, the total length of a SIFT feature is $4 \times 4 \times 8 = 128$. Assuredly our approach can also work with any other legitimate local features.



(a) Caltech256

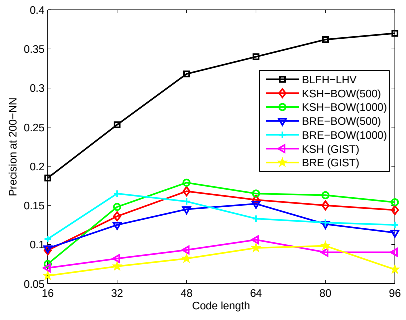


(b) SUN397

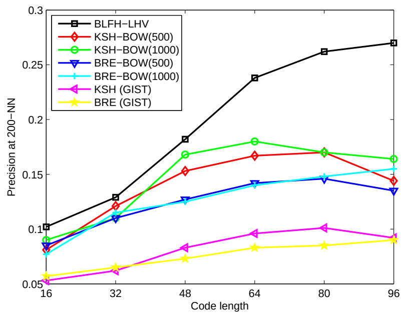


(c) NUS-WIDE

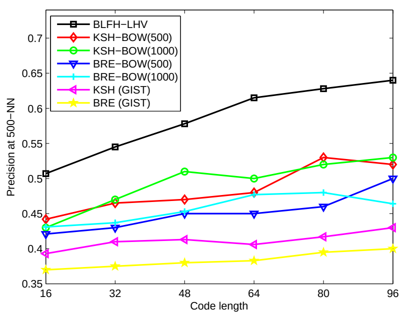
Fig. 7.4 Performance comparison with different numbers of bits.



(a) Caltech256



(b) SUN397



(c) NUS-WIDE

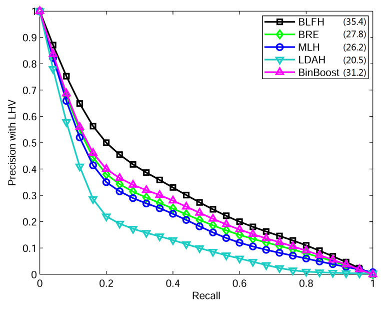
Fig. 7.5 Comparison performance (MAP) of our BLFH and other global hashing schemes. Note: KSH and BRE are the top-performing supervised hashing methods.

Mean Average Precision (MAP) and the precision-recall curve by changing the number of top ranked points in LHV. Additionally, we also report the training time and the test time (the average searching time used for each query) for all the methods.

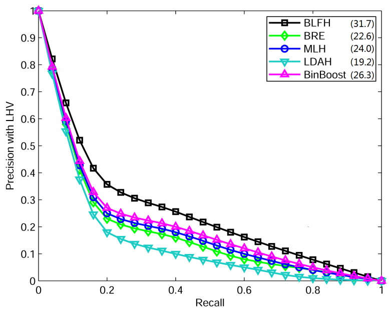
Our experiments are completed using Matlab 2013a on a server configured with a 12-core processor and 128G of RAM running the Linux OS.

Compared Methods and Settings

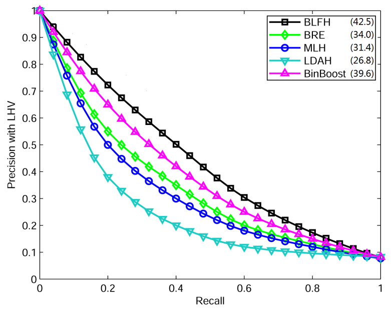
Since our BLFH is the pioneer work for local feature hashing, we can only adopt existing global hashing methods for comparison. Thus, in our experiments, we compare the proposed method against nine general hashing algorithms, including five supervised methods: LDAH [183], BRE [184], MLH [185], KSH [72] and BinBoost descriptor [76], and four unsupervised methods: LSH [172], PCAH [68], SpH [69] and AGH [70]. All the above methods (except BinBoost) are computed on the extracted SIFT local features to show their capability. It is noteworthy that we treat BinBoost as a binary feature learning method which



(a) Caltech256

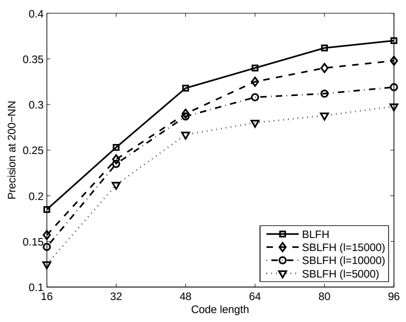


(b) SUN397

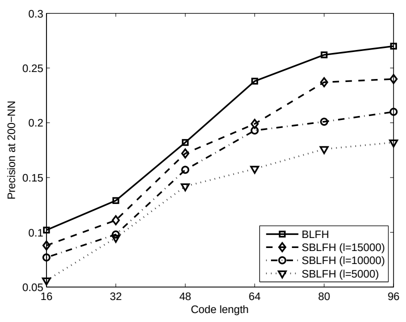


(c) NUS-WIDE

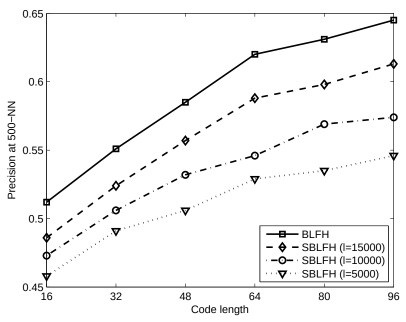
Fig. 7.6 The precision-recall curves of all compared algorithms on the three datasets with the code length of 96 bits. The corresponding AUC values of compared methods are illustrated in the brackets.



(a) Caltech256



(b) SUN397



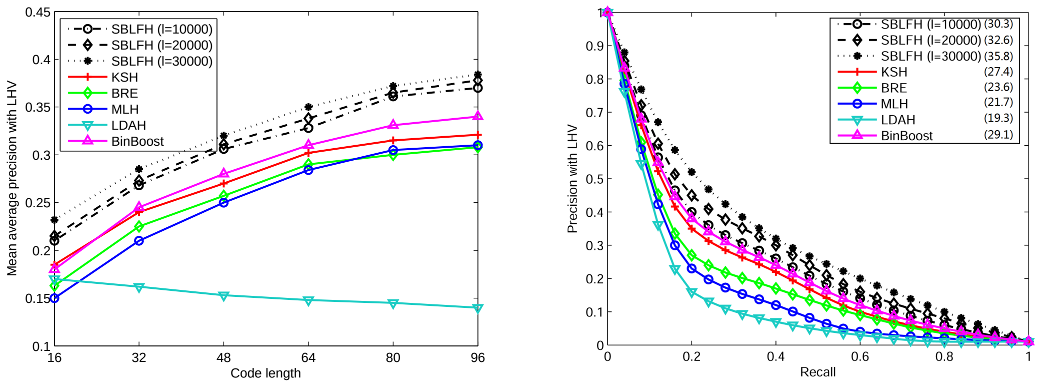
(c) NUS-WIDE

Fig. 7.7 Semi-supervised performance comparison with different numbers of bits. l denotes the number of labeled examples for semi-supervised hashing training methods.

Table 7.2 MAP of 32 bits with training and test time (with LHV) on three datasets.

Methods	**Caltech-256**			**SUN397**			**NUS-WIDE**		
	MAP (200-NN)	**Training time**	**Test time**	**MAP (200-NN)**	**Training time**	**Test time**	**MAP (500-NN)**	**Training time**	**Test time**
KSH	0.196	4741.1s	57.4μs	0.106	8384.2s	63.8μs	0.492	9987.0s	62.7μs
BRE	0.188	2.84×10^4s	32.9μs	0.112	5.17×10^4s	48.8μs	0.488	5.83×10^4s	53.1μs
MLH	0.180	1.19×10^4s	18.3μs	0.100	3.48×10^4s	25.0μs	0.480	3.32×10^4s	28.4μs
LDAH	0.195	15.3s	11.2μs	0.091	41.4s	24.8μs	0.455	56.3s	26.1μs
BinBoost	0.212	9345.0s	134.7μs	0.122	2.12×10^4s	164.5μs	0.512	2.91×10^4s	174.8μs
AGH	0.172	764.2s	97.4μs	0.082	1832.8s	105.1μs	0.463	2018.0s	112.0μs
SpH	0.154	282.7s	103.4μs	0.074	883.0s	101.8μs	0.444	944.3s	109.6μs
PCAH	0.142	17.4s	16.1μs	0.052	42.4s	18.4μs	0.442	58.1s	19.2μs
LSH	0.104	9.1s	8.3μs	0.044	23.6s	11.5μs	0.414	31.7s	10.8μs
BLFH	**0.253**	1341.4s	49.3μs	**0.129**	3181.2s	54.1μs	**0.551**	3451.5s	57.0μs
SBLFH (l=10000)	0.235	1903.0s	53.1μs	0.098	4892.0s	65.2μs	0.506	5082.1s	68.5μs
SBLFH (l=15000)	0.240	1982.0s	57.2μs	0.111	4939.0s	68.0μs	0.524	5173.1s	69.1μs

(The test time indicates the average searching time used for each query.)



(a) MAP vs. code length (b) Precision-recall with the AUC values

Fig. 7.8 The retrieval results on the Flickr 1M dataset.

learns to hash from the original local patches[3], however, the other methods construct the binary codes from extracted SIFT descriptors. We used the publicly available codes of BRE, MLH, LDAH, SpH and AGH, and implemented LSH, PCAH, KSH and BinBoost ourselves. All of the above methods are then evaluated on six different lengths of codes (16, 32, 48, 64, 80, 96). Under the same experimental setting, all the parameters used in the compared methods have been strictly chosen according to their original papers.

For our BLFH/SBLFH, we set $k = 15$ for pairwise data structure preserving. In this way, we assign the pairwise label $\ell_{ij} = +1$ if they are 15 nearest neighbors of each other in Euclidean space, for others, we assign the pairwise label $\ell_{ij} = -1$. Besides, we set $D_1 = 16$ and $D_2 = 8$ for the transformation of SIFT local features (see Section 7.3.1)[4]. We further adopt $K = 300$ in the K-means clustering. For BLFH/SBLFH, the optimal regularization parameter γ for each dataset is selected from one of $\{0.05, 0.1, 0.15, 0.2, \ldots, 0.45, 0.5\}$ with the step of 0.05, which yields the best performance by 10-fold cross-validation. In addition, for SBLFH, the $\lambda \in \{0.1, 0.2, 0.3, \ldots, 0.9, 1\}$ is determined by 10-fold cross-validation, as well.

Results Comparison

Fig. 7.4 illustrates the MAP curves of all compared algorithms on Caltech-256, SUN397 and NUS-WIDE datasets. All MAP values are calculated using the proposed LHV ranking algorithm under the same setting. In its entirety, the searching accuracies on the NUS-WIDE

[3]Each patch is with the size of 16×16 located on the keypoints detected by SIFT.

[4]Generally, the factorization for the dimension of local features is varied. However, the complexity reaches the minimum value when $D_1 = D_2$ due to the inequality $D_1^3 + D_2^3 \geq 2\sqrt{D_1^3 D_2^3}$. Thus we let D_1 and D_2 be close enough.

dataset are obviously higher than that on the other two datasets with more categories. Specifically, the supervised methods, KSH, BRE, MLH, LDAH and BinBoost, always achieve the better performance than unsupervised methods, since the label information is involved in the learning phases. BinBoost has the best performance compared with all other existing methods and AGH achieves the best performance among all of the unsupervised methods. Furthermore, the results of KSH always climb up then go down when the length of codes increases. The same tendency also appears with BRE, LDAH and PCAH. LSH consistently brings the worst actuaries on all these datasets. In general, our BLFH significantly outperforms all other compared methods. Besides, we also compare our method with global hash schemes. We first use the 'Bag-of-Words' scheme with 500 and 1000 codebook sizes to encode SIFT features into global representations. After that, two best performing[5] hashing methods, KSH and BRE, are used to learn the hash codes on these global features. Besides, we also list the search performance via directly using the global feature 'GIST' with KSH and BRE. In Fig. 7.5, one can see that our local hashing method BLFH with LHV achieves better results than the compared global hashing schemes. Moreover, the precision-recall curves of all the comparable methods on three datasets with the code length of 96 bits are presented in Fig. 7.6, as well. From all these figures, we can further discover that, for all three datasets, BLFH achieves better performance than other methods for both the Mean Average Precision (MAP) and Area Under the Curve (AUC).

In addition, we illustrate the relevant results of our semi-supervied version SBLFH in Fig. 7.7. We have systematically evaluated SBLFH by using different numbers of labeled data. Particularly, we learn SBLFH using the training sets with only 15000, 10000 and 5000 data labeled, respectively and then compare their retrieval actuaries with the fully supervised BLFH under the same experimental setting. It can be obviously observed that, the more labeled data being used, the better performance can be achieved. All the detailed information can be seen in Fig. 7.4–7.7.

Finally, the training time and test time for different algorithms on three datasets are illustrated in Table 7.2. Considering the training time, supervised methods always need more time for the hash learning except LDAH. In particular, BRE and MLH spend the most time to train hashing functions. BinBoost also spends much time for training since hash functions are learned from the pixel-level local patches. While, the random projection based algorithms are relatively efficient, especially the LSH. Our BLFH costs significantly less time than KSH, BRE, MLH and BinBoost but more than other methods for training. BLFH takes less time compared with SBLFH due to the extra cost item in its objective function. In terms of the test phase, LSH, LDAH and PCAH are the most efficient methods,

[5]BinBoost is not considered here, since it cannot be directly applied on extracted features.

while AGH has the competitive searching time with SpH. BinBoost needs the most time for testing. More details can be also seen in Table 7.2.

7.7.3 Large-scale Image Retrieval

To further evaluate the scalability of the proposed BLFH approach, a larger image dataset named Flickr 1M[6] is used in our experiments. This dataset contains one million images collected from the Flickr and the SIFT features are extracted from each image. To control the computational complexity, we limit the number of the SIFT features for each image to around 300. In a realistic scenario, we cannot obtain all of the label information in a huge dataset for image retrieval tasks. Thus, on this dataset, only the semi-supervised BLFH (S-BLFH) is used to enable a realistic setting. We randomly select $1K$ images as the queries and use the remaining to form the gallery database. Considering the huge cost of computation, in this experiment, only $25,000$ randomly selected data points from the gallery database form the training set. Furthermore, due to the semi-supervised setting, we test the proposed SBLFH algorithm with only $10,000$, $20,000$, $30,000$ labeled data in the training phase, respectively. Later, we evaluate the retrieval results by the Mean Average Precision (MAP) and the precision-recall curve on the Flickr 1M as shown in Fig. 7.8. It can be obviously observed that the higher results are obtained when more label information is involved in our hashing code learning stage. In addition, compared with other fully supervised hashing techniques, our SBLFH can achieve significantly better results with different lengths of the codes. Meanwhile, the AUCs of the proposed SBLFH are also consistently higher than those of compared methods.

7.8 Summary

In this chapter, we have presented a novel supervised framework, BLFH, to learn highly discriminative binary codes on the local descriptors for large-scale image similarity search. We address it as a nonconvex optimization problem to seek orthogonal projection matrices for hashing, which can successfully preserve the pairwise similarity between different local features and simultaneously take image-to-class (I2C) distances into consideration. For more realistic scenarios, we further extend BLFH to the semi-supervised version, S-BLFH, which only utilizes a few labeled samples in the hashing learning phase but can still achieve competitive results. We have systematically evaluated our methods on Caltech-256, SUN397, NUS-WIDE and Flickr 1M datasets and show promising results compared with

[6]http://www.multimedia-computing.de/wiki/Flickr1M

state-of-the-art hashing methods. In the next chapter, a brief conclusion is conducted and the potential future work is listed, as well.

Chapter 8

Conclusions and Future Work

In this chapter, the contributions of this thesis have been first briefly concluded. Furthermore, the possible future work have been also discussed.

8.1 Conclusions

In the first part, a boosted discriminative key poses selection scheme is first introduced for improved human action recognition. In particular, Poses in video frames are described by the proposed extensive pyramidal features (EPFs), which include the Gabor, Gaussian, and wavelet pyramids. These features are able to encode the orientation, intensity, and contour information and therefore provide an informative representation of human poses. Due to the fact that not all poses in a sequence are discriminative and representative, we further utilize the AdaBoost algorithm to learn a subset of discriminative poses. Given the boosted poses for each video sequence, a new classifier named weighted local naive Bayes nearest neighbor is proposed for the final action classification, which is demonstrated to be more accurate and robust than other classifiers, e.g., support vector machine (SVM) and naive Bayes nearest neighbor.

In the second part, a novel feature learning method is addressed for gesture recognition. we successfully apply an evolutionary method-genetic programming (GP) to synthesize machine learned spatio-temporal descriptors for automatic gesture recognition instead of using hand-crafted descriptors. In our architecture, a set of primitive low-level 3D operators are first randomly assembled as tree-based combinations, which are further evolved generation-by generation through the GP system, and finally a well performed combination will be selected as the best descriptor for high-level gesture recognition. To the best of our knowledge, this is the first report of using GP to evolve spatio-temporal descriptors for gesture recognition.

Inspired by the previous work, in the third part, a set of primitive 2-D operators are randomly combined to construct feature descriptors for image classification through the MOGP evolving and then evaluated by two objective fitness criteria, i.e., the classification error and the tree complexity. After the entire evolution procedure finishes, the best-so-far solution selected by the MOGP is regarded as the (near-)optimal feature descriptor obtained. This MOGP-based method can effectively control the over-fitting and proved to be a good solution to an NP-hard problem in an acceptable amount of computing time. Experimental results verify that the proposed method significantly outperforms other hand-crafted features.

In the fourth part, a novel linear unsupervised dimensionality reduction algorithm is proposed, termed Discriminative Partition Sparsity Analysis (DPSA), explicitly considering different probabilistic distributions that exist over the data points, meanwhile preserving the natural locality relationship among the data. Specifically, the Gaussian mixture model (GMM) is first applied to partition all samples into several clusters. In each cluster, a number of sparse sub-graphs are computed via the ℓ-norm constraint to optimally represent the intrinsic data structure. Such sub-graphs are demonstrated to be robust to data noise, automatically sparse and adaptive to the neighborhood. All the sub-graphs from the clusters are then combined into a whole discriminative optimization framework for final reduction.

In the fifth part, to solve many large-scale image classification and retrieval tasks, we propose a novel framework to automatically learn the task-specific compact coding, called evolutionary compact embedding (ECE), which can be regarded as an optimization algorithm combining genetic programming (GP) and a boosting trick. As an evolutionary computation methodology, GP can solve problems inspired by natural evolution without any prior knowledge of the solutions. In our evolutionary architecture, each bit of ECE is iteratively computed using a binary classification function,which is generated through GP evolving by jointly minimizing its empirical risk with the AdaBoost strategy on a training set. We address this as greedy optimization leading to small Hamming distances for similar samples and large distances for dissimilar samples. The relevant result shows the accurate and robust performance of our method for large-scale image classification/retrieval.

In the last part, the Bilinear Local Feature Hashing (BLFH) is proposed, which is a novel supervised hashing technique based on local features for image search and retrieval. The proposed scheme aims to embed local feature descriptors from a high-dimensional space into a Hamming space with a low dimension, rather than using global features as in most existing hashing techniques which are susceptible to image variations such as viewpoint changes and background cluttering. Specifically, BLFH seeks two orthogonal projection matrices to preserve the pairwise attributes including labels and similarities between different local features. Meanwhile, a bipartite graph regularization item, which is constructed by images and classes, is simultaneously used to preserve the image-level intrinsic structure of local features. Since the raised problem is regarded as nonconvex and discrete, our objective function is then optimized via an alternate way with relaxation and finally converges to a near-optimal solution. Furthermore, BLFH is extended to semi-supervised BLFH (SBLFH) for more realistic applications. Extensive experiments consistently prove the superiority of the proposed methods compared to the state-of-the-art hashing techniques.

8.2 Future Work

Considering the current jobs their potential extension in computer vision community, in the following part, several topics that can be researched in the future time will be briefly illustrated.

- **Hashing based on feature merging:** Most existed hashing embedding methods is based on three schemes: (1) hashing via feature thresholding; (2) embedding with learning projections; (3) embedding using regression algorithms, e.g., SVM and logistic regression. However, these schemes above always rely on complex mathematical analysis such as full-matrix-decomposition, which could cause significantly computational usages. Particularly for the hashing problem, when applying on large-scale visual data, how to efficiently obtain the meaningful binary embedding becomes the most important issue. Inspired by the recently jobs using feature merging for efficient dimensionality reduction, we aim to develop a more scalable and efficient merging based hashing scheme to solve extremely large-scale (usually more than 10 million) image retrieval tasks. In detail, we will consider both the correlation between each features in on representation and the geometry information for all of representations in our objective function.

- **Semantic local hashing for human action retrieval:** For the large-scale video based retrieval tasks, it is very difficult to represent a video sequences meaningfully with just one global feature, since cluttering and complex background always exist in these videos. Inspired by previous work on local feature based hashing for efficient image search, in the video retrieval tasks, this technique can be easily extended to carry on videos. To further improve the retrieval capability, the semantic information of videos is also included in our scheme as a constraint. Specifically, we need to match human actions with their particular scenario information. For instance, running Vs. field (✓); swimming Vs. water (✓); running Vs. water (×). We will add this semantic constraint as a regularization item in our objective function to obtain more unique and meaningful hash codes for human action retrieval tasks.

- **Deep learning and its application:** Deep learning nowadays has become one of the most attractive topic in both research and industry area. Deep learning currently provide the best solutions to many problems in image recognition, speech recognition, and natural language processing. In future work, particularly, we will focus on two aspects with deep learning: (1) deep learning for objects detection; (2) deep learning for unsupervised feature learning. For detection application, deep learned networks can efficiently and precisely to classify the positive samples from negative ones and in the application of feature learning, deep networks have also shown their powerful capabilities for visual recognition tasks.

References

[1] G. Dahl, A.-r. Mohamed, G. E. Hinton *et al.*, "Phone recognition with the mean-covariance restricted boltzmann machine," in *Advances in neural information processing systems*, 2010, pp. 469–477.

[2] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[3] L. Deng, M. Seltzer, D. Yu, A. Acero, A. rahman Mohamed, and G. Hinton, "Binary coding of speech spectrograms using a deep auto-encoder," in *Interspeech*, 2010, pp. 57–72.

[4] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649.

[5] O. Chapelle, P. Haffner, and V. Vapnik, "Support vector machines for histogram-based image classification," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1055–1064, 1999.

[6] A. Vailaya, M. Figueiredo, A. Jain, and H. Zhang, "Image classification for content-based indexing," *IEEE Transactions on Image Processing*, vol. 10, no. 1, pp. 117–130, 2001.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[8] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[9] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns," *European Conference on Computer Vision*, pp. 469–481, 2004.

[10] V. Gaede and O. Günther, "Multidimensional access methods," *ACM Computing Surveys (CSUR)*, vol. 30, no. 2, pp. 170–231, 1998.

[11] L. Cayton and S. Dasgupta, "A learning framework for nearest neighbor search," in *Neural Information Processing Systems*, 2007, pp. 233–240.

[12] D. Cai, X. He, and J. Han, "Speed up kernel discriminant analysis," *VLDB*, vol. 20, no. 1, pp. 21–33, 2011.

[13] D. Peng, D. Huijun, D. Ligeng, T. Linmi, and X. Guangyou, “Group interaction analysis in dynamic context,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 38, no. 1, pp. 275–282, 2008.

[14] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, “Behavior recognition via sparse spatio-temporal features,” in *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005, pp. 65–72.

[15] Y. Xu, D. Xu, S. Lin, T. Han, X. Cao, and X. Li, “Detection of sudden pedestrian crossings for driving assistance systems,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, no. 42, pp. 729–739, 2012.

[16] J. Niebles, H. Wang, and L. Fei-Fei, “Unsupervised learning of human action categories using spatial-temporal words,” *International Journal of Computer Vision*, vol. 79, no. 3, pp. 299–318, 2008.

[17] W. Bian, D. Tao, and Y. Rui, “Cross-domain human action recognition,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 298–307, 2012.

[18] T. Ko, “A survey on behavior analysis in video surveillance for homeland security applications,” in *IEEE Applied Imagery Pattern Recognition Workshop*, 2008, pp. 1–8.

[19] S. Zhang, M. H. Ang, W. Xiao, and C. K. Tham, “Detection of activities for daily life surveillance: Eating and drinking,” in *International Conference on e-health Networking, Applications and Services*, 2008, pp. 171–176.

[20] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, “Learning realistic human actions from movies,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[21] P. Scovanner, S. Ali, and M. Shah, “A 3-dimensional sift descriptor and its application to action recognition,” in *Proceedings of the 15th international conference on Multimedia*, 2007, pp. 357–360.

[22] A. Klaser and M. Marszalek, “A spatio-temporal descriptor based on 3d-gradients,” in *British Machine Vision Conference*, 2008, pp. 120–129.

[23] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (surf),” *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[24] I. Laptev, “On space-time interest points,” *International Journal of Computer Vision*, vol. 64, no. 2, pp. 107–123, 2005.

[25] A. Oikonomopoulos, I. Patras, and M. Pantic, “Spatiotemporal salient points for visual recognition of human actions,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 36, no. 3, pp. 710–719, 2005.

[26] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, “An efficient k-means clustering algorithm: Analysis and implementation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[27] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," in *International Conference on Machine Learning*, 2010, pp. 495–502.

[28] H. Lu, G. Fang, X. Shao, and X. Li, "Segmenting human from photo images based on a coarse-to-fine scheme." *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics: a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 42, no. 3, pp. 889–899, 2012.

[29] K. Schindler and L. Van Gool, "Action snippets: How many frames does human action recognition require?" in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[30] S. Baysal, M. Kurt, and P. Duygulu, "Recognizing human actions using key poses," in *International Conference on Pattern Recognition*, 2010, pp. 1727–1730.

[31] X. Cao, B. Ning, P. Yan, and X. Li, "Selecting key poses on manifold for pairwise action recognition," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 168–177, 2012.

[32] L. Shao, D. Wu, and X. Chen, "Action recognition using correlogram of body poses and spectral regression," in *18th IEEE International Conference on Image Processing*, 2011, pp. 209–212.

[33] S. Cheema, A. Eweiwi, C. Thurau, and C. Bauckhage, "Action recognition by learning discriminative key poses," in *IEEE International Conference on Computer Vision Workshops*, 2011, pp. 1302–1309.

[34] Y. Freund and R. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," *Computational Learning Theory*, vol. 1, pp. 23–37, 1995.

[35] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[36] S. McCann and D. G. Lowe, "Local naive bayes nearest neighbor for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3650–3656.

[37] Y. Bai, L. Guo, L. Jin, and Q. Huang, "A novel feature extraction method using pyramid histogram of orientation gradients for smile recognition," in *IEEE International Conference on Image Processing*, 2009, pp. 3305–3308.

[38] S. Belongie, J. Malik, and J. Puzicha, "Shape context: A new descriptor for shape matching and object recognition," *Advances in neural information processing systems*, pp. 831–837, 2001.

[39] R. Poli, W. Langdon, and N. McPhee, *A field guide to genetic programming*, 2008.

[40] L. Liu, L. Shao, and X. Li, "Building holistic descriptors for scene recognition: a multi-objective genetic programming approach," in *Proceedings of the 21st ACM international conference on Multimedia*, 2013, pp. 997–1006.

[41] G.-F. Lu and Y. Wang, "Feature extraction using a fast null space based linear discriminant analysis algorithm," *Information Sciences*, vol. 193, pp. 72–80, 2012.

[42] X. He and P. Niyogi, "Locality preserving projections," in *Neural information processing systems*, vol. 16, 2003, pp. 153–161.

[43] C.-C. Lin, S.-C. Chen, and N.-L. Hsueh, "Adaptive embedding techniques for vq-compressed images," *Information Sciences*, vol. 179, no. 1, pp. 140–149, 2009.

[44] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 6, pp. 1092–1104, 2012.

[45] H. Cheng, K. Vu, and K. A. Hua, "Subspace projection: A unified framework for a class of partition-based dimension reduction techniques," *Information Sciences*, vol. 179, no. 9, pp. 1234–1248, 2009.

[46] S. Jones and L. Shao, "Content-based retrieval of human actions from realistic video databases," *Information Sciences*, vol. 236, no. 7, pp. 56–65, 2013.

[47] U. Jayaraman, A. K. Gupta, and P. Gupta, "Boosted geometric hashing based indexing technique for finger-knuckle-print database," *Information Sciences*, vol. 275, pp. 30–44, 2014.

[48] S. Maldonado, R. Weber, and J. Basak, "Simultaneous feature selection and classification using kernel-penalized support vector machines," *Information Sciences*, vol. 181, no. 1, pp. 115–128, 2011.

[49] M.-L. Zhang, J. M. Peña, and V. Robles, "Feature selection for multi-label naive bayes classification," *Information Sciences*, vol. 179, no. 19, pp. 3218–3229, 2009.

[50] L. Shao, T. Kadir, and M. Brady, "Geometric and photometric invariant distinctive regions detection," *Information Sciences*, vol. 177, no. 4, pp. 1088–1122, 2007.

[51] H. Guan, J. Zhou, B. Xiao, M. Guo, and T. Yang, "Fast dimension reduction for document classification based on imprecise spectrum analysis," *Information Sciences*, vol. 222, pp. 147–162, 2013.

[52] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[53] L. Shao, L. Liu, and X. Li, "Feature learning for image classification via multiobjective genetic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 7, pp. 1359–1371, 2014.

[54] F. Zhu and L. Shao, "Weakly-supervised cross-domain dictionary learning for visual recognition," *International Journal of Computer Vision*, pp. 1–18, 2014.

[55] L. Liu, L. Shao, X. Zhen, and X. Li, "Learning discriminative key poses for action recognition," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1860–1870, 2013.

[56] S. Chakrabarti, S. Roy, and M. V. Soundalgekar, “Fast and accurate text classification via multiple linear discriminant projections,” *The VLDB Journal*, vol. 12, no. 2, pp. 170–185, 2003.

[57] K. Fukunaga, *Introduction to statistical pattern recognition*, 1990.

[58] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Mullers, “Fisher discriminant analysis with kernels,” in *Neural Networks for Signal Processing IX, IEEE Signal Processing Society Workshop.*, 1999, pp. 41–48.

[59] T. Zhang, D. Tao, and J. Yang, “Discriminative locality alignment,” in *European Conference on Computer Vision*, 2008, pp. 725–738.

[60] M. Belkin and P. Niyogi, “Laplacian eigenmaps and spectral techniques for embedding and clustering,” in *Conference on Neural Information Processing Systems*, 2001, pp. 585–591.

[61] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[62] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet, “Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering,” in *Conference on Neural Information Processing Systems*, vol. 16, 2003, pp. 177–184.

[63] X. He, D. Cai, S. Yan, and H. Zhang, “Neighborhood preserving embedding,” in *IEEE International Conference on Computer Vision*, 2005, pp. 1208–1213.

[64] D. Cai, H. Bao, and X. He, “Sparse concept coding for visual analysis,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 2905–2910.

[65] G. Shakhnarovich, “Learning task-specific similarity,” Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[66] R. Salakhutdinov and G. Hinton, “Semantic hashing,” *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[67] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[68] J. Wang, S. Kumar, and S. Chang, “Semi-supervised hashing for large scale search,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 107–120, 2012.

[69] Y. Weiss, A. Torralba, and R. Fergus, “Spectral hashing,” in *Neural Information Processing Systems*, 2008, pp. 1753–1760.

[70] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, “Hashing with graphs,” in *International Conference on Machine Learning*, 2011, pp. 1–8.

[71] D. Zhang, J. Wang, D. Cai, and J. Lu, “Self-taught hashing for fast similarity search,” in *Proceedings of International Conference on Research and Development in Information Retrieval*, 2010, pp. 18–25.

[72] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, “Supervised hashing with kernels,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2074–2081.

[73] Y. Lin, R. Jin, D. Cai, S. Yan, and X. Li, “Compressed hashing,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 446–451.

[74] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors),” *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[75] A. Oliva and A. Torralba, “Modeling the shape of the scene: A holistic representation of the spatial envelope,” *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[76] T. Trzcinski, C. M. Christoudias, P. Fua, and V. Lepetit, “Boosting binary keypoint descriptors,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2874–2881.

[77] J. Ye, R. Janardan, Q. Li *et al.*, “Two-dimensional linear discriminant analysis,” in *Conference on Neural Information Processing Systems*, 2004, pp. 1569–1576.

[78] C. Thurau and V. Hlavác, “Pose primitive based human action recognition in videos or still images,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[79] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[80] J. Niebles and L. Fei-Fei, “A hierarchical model of shape and appearance for human action classification,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[81] W. Yang, Y. Wang, and G. Mori, “Recognizing human actions from still images with latent poses,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2030–2037.

[82] S. M. M. Huang, J.and Kumar, W. Zhu, and R. Zabih, “Image indexing using color correlograms,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 762–768.

[83] M. Cooper and J. Foote, “Discriminative techniques for keyframe selection,” in *IEEE International Conference on Multimedia and Expo*, pp. 4–12.

[84] Z. Zhao and A. Elgammal, “Information theoretic key frame selection for action recognition,” in *British Machine Vision Conference*, 2008, pp. 1–10.

[85] Y. Zhuang, Y. Rui, T. Huang, and S. Mehrotra, “Adaptive key frame extraction using unsupervised clustering,” in *International Conference on Image Processing*, vol. 1, 1998, pp. 866–870.

[86] A. Fathi and G. Mori, “Action recognition by learning mid-level motion features,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[87] L. Shen and L. Bai, “Adaboost gabor feature selection for classification,” *Image and Vision Computing*, pp. 77–83, 2004.

[88] R. E. Schapire, “The boosting approach to machine learning: An overview,” in *Nonlinear estimation and classification*, 2003, pp. 149–171.

[89] C. Shan, S. Gong, and P. W. McOwan, “Facial expression recognition based on local binary patterns: A comprehensive study,” *Image and Vision Computing*, vol. 27, no. 6, pp. 803–816, 2009.

[90] G. Zhang, X. Huang, S. Z. Li, Y. Wang, and X. Wu, “Boosting local binary pattern (lbp)-based face recognition,” in *Advances in biometric person authentication*, 2005, pp. 179–186.

[91] M. Zhou and H. Wei, “Face verification using gaborwavelets and adaboost,” in *International Conference on Pattern Recognition*, vol. 1, 2006, pp. 404–407.

[92] A. Treptow and A. Zell, “Combining adaboost learning and evolutionary search to select features for real-time object detection,” in *Congress on Evolutionary Computation*, vol. 2, 2004, pp. 2107–2113.

[93] S. Paisitkriangkrai, C. Shen, and J. Zhang, “Fast pedestrian detection using a cascade of boosted covariance features,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 8, pp. 1140–1151, 2008.

[94] I. Laptev and P. Pérez, “Retrieving actions in movies,” in *International Conference on Computer Vision*, 2007, pp. 1–8.

[95] M. Riesenhuber and T. Poggio, “Hierarchical models of object recognition in cortex,” *nature neuroscience*, vol. 2, pp. 1019–1025, 1999.

[96] C. Yang, M. Schmalz, W. Hu, and G. Ritter, “Center-surround filters for the detection of small targets in cluttered multispectral imagery: background and filter design,” in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, 1995, pp. 637–648.

[97] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, “Image coding using wavelet transform,” *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 205–220, 1992.

[98] I. Daubechies, “The wavelet transform, time-frequency localization and signal analysis,” *IEEE Transactions on Information Theory*, vol. 36, no. 5, pp. 961–1005, 1990.

[99] A. Cohen, I. Daubechies, and J. Feauveau, “Biorthogonal bases of compactly supported wavelets,” *Communications on pure and applied mathematics*, vol. 45, no. 5, pp. 485–560, 1992.

[100] G. De’ath and K. Fabricius, “Classification and regression trees: a powerful yet simple technique for ecological data analysis,” *Ecology*, vol. 81, no. 11, pp. 3178–3192, 2000.

[101] C. Schuldt, I. Laptev, and B. Caputo, “Recognizing human actions: A local svm approach,” in *International Conference on Pattern Recognition*, 2004, pp. 32–36.

[102] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, “Actions as space-time shapes,” in *International Conference on Computer Vision*, 2005, pp. 1395–1402.

[103] D. Weinland, E. Boyer, and R. Ronfard, “Action recognition from arbitrary views using 3d exemplars,” in *IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–7.

[104] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “Hmdb: A large video database for human motion recognition,” in *13th IEEE International Conference on Computer Vision*, Barcelona, Spain, 2011, pp. 2556 – 2563.

[105] A. Yao, J. Gall, and L. Van Gool, “A hough transform-based voting framework for action recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2061–2068.

[106] G. Taylor, R. Fergus, Y. LeCun, and C. Bregler, “Convolutional learning of spatio-temporal features,” in *Europen Conference on Computer Vsion*, 2010, pp. 140–153.

[107] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, “A biologically inspired system for action recognition,” in *International Conference on Computer Vision*, 2007, pp. 1–8.

[108] Q. Le, W. Zou, S. Yeung, and A. Ng, “Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis,” in *Conference on Computer Vision and Pattern Recognition*, 2011, pp. 3361–3368.

[109] J. Liu and M. Shah, “Learning human actions via information maximization,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[110] M. Yang, F. Lv, W. Xu, K. Yu, and Y. Gong, “Human action detection by boosting efficient motion features,” in *International Conference on Computer Vision Workshops*, 2009, pp. 522–529.

[111] M. Varma and B. Babu, “More generality in efficient multiple kernel learning,” in *International Conference on Machine Learning*, 2009, pp. 1065–1072.

[112] X. Wu, D. Xu, L. Duan, and J. Luo, “Action recognition using context and appearance distribution features,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 489–496.

[113] D. Weinland, R. Ronfard, and E. Boyer, “Free viewpoint action recognition using motion history volumes,” *Computer Vision and Image Understanding*, vol. 104, no. 2-3, pp. 249–257, 2006.

[114] S. Mitra and T. Acharya, “Gesture recognition: A survey,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 3, pp. 311–324, 2007.

[115] M. Holte, T. Moeslund, and P. Fihl, “View-invariant gesture recognition using 3d optical flow and harmonic motion context,” *Computer Vision and Image Understanding*, vol. 114, no. 12, pp. 1353–1361, 2010.

[116] R. Cutler and M. Turk, “View-based interpretation of real-time optical flow for gesture recognition,” in *Third IEEE International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 416–416.

[117] L. Campbell, D. Becker, A. Azarbayejani, A. Bobick, and A. Pentland, “Invariant features for 3-d gesture recognition,” in *The Second International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 157–157.

[118] L. Bretzner, I. Laptev, and T. Lindeberg, “Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering,” in *Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, 2002, pp. 423–428.

[119] L. Trujillo and G. Olague, “Synthesis of interest point detectors through genetic programming,” in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 2006, pp. 887–894.

[120] R. Torres, A. Falcão, M. Gonçalves, J. Papa, B. Zhang, W. Fan, and E. Fox, “A genetic programming framework for content-based image retrieval,” *Pattern Recognition*, vol. 42, no. 2, pp. 283–292, 2009.

[121] R. Poli, “Genetic programming for image analysis,” in *Proceedings of the First Annual Conference on Genetic Programming*, 1996, pp. 363–368.

[122] D. Howard, S. Roberts, and R. Brankin, “Target detection in SAR imagery by genetic programming,” *Advances in Engineering Software*, vol. 30, no. 5, pp. 303–311, 1999.

[123] R. Davis, A. Charlton, S. Oehlschlager, and J. Wilson, “Novel feature selection method for genetic programming using metabolomic HNMR data,” *Chemometrics and Intelligent Laboratory Systems*, vol. 81, no. 1, pp. 50–59, 2006.

[124] D. Fogel, “What is evolutionary computation?” *Spectrum, IEEE*, vol. 37, no. 2, pp. 26–28, 2000.

[125] D. Song and D. Tao, “Biologically inspired feature manifold for scene classification,” *IEEE Transactions on Image Processing*, vol. 19, no. 1, pp. 174–184, 2010.

[126] M. Riesenhuber, T. Poggio *et al.*, “Hierarchical models of object recognition in cortex,” *Nature neuroscience*, vol. 2, pp. 1019–1025, 1999.

[127] T. Kim, S. Wong, and R. Cipolla, “Tensor canonical correlation analysis for action classification,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[128] X. Shen, Z. Lin, J. Brandt, and Y. Wu, “Dynamic hand gesture recognition: An exemplar-based approach from motion divergence fields,” *Image and Vision Computing*, vol. 30, no. 3, pp. 227–235, 2012.

[129] S.-F. Wong and R. Cipolla, “Real-time interpretation of hand motions using a sparse bayesian classifier on motion gradient orientation images,” in *British Machine Vision Conference*, 2005, pp. 160–172.

[130] J. Davis, “Hierarchical motion history images for recognizing human motion,” in *IEEE Workshop on Detection and Recognition of Events in Video*, 2001, pp. 39–46.

[131] A. Kläser, M. Marszałek, and C. Schmid, “A spatio-temporal descriptor based on 3d-gradients,” in *British Machine Vision Conference*, 2008, pp. 120–129.

[132] P. Scovanner, S. Ali, and M. Shah, “A 3-dimensional sift descriptor and its application to action recognition,” in *International Conference on Multimedia*, 2007, pp. 357–360.

[133] L. Fei-Fei, R. Fergus, and P. Perona, “Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories,” *Computer Vision and Image Understanding*, vol. 106, no. 1, pp. 59–70, 2007.

[134] T. Sim, S. Baker, and M. Bsat, “The cmu pose, illumination, and expression (pie) database,” in *IEEE International Conference on Automatic Face and Gesture Recognition.*, 2002, pp. 46–51.

[135] J. Triesch and C. Von Der Malsburg, “A system for person-independent hand posture recognition against complex backgrounds,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 12, pp. 1449–1453, 2001.

[136] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, “Sun database: Large-scale scene recognition from abbey to zoo,” in *IEEE Computer Vision and Pattern Recognition*, 2010, pp. 3485–3492.

[137] L.-J. Li and L. Fei-Fei, “What, where and who? classifying events by scene and object recognition,” in *International Conference on Computer Vision*, 2007, pp. 1–8.

[138] J. Vogel and B. Schiele, “A semantic typicality measure for natural scene categorization,” *Pattern Recognition*, pp. 195–203, 2004.

[139] G. Hinton, S. Osindero, and Y. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[140] Y. LeCun, K. Kavukcuoglu, and C. Farabet, “Convolutional networks and applications in vision,” in *IEEE International Symposium on Circuits and Systems*, 2010, pp. 253–256.

[141] J. Koza, “Evolution of subsumption using genetic programming,” in *the First European Conference on Artificial Life*, 1992, pp. 110–119.

[142] W. Tackett, “Genetic programming for feature discovery and image discrimination,” in *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, 1993, pp. 303–309.

[143] M. Bot, “Feature extraction for the k-nearest neighbour classifier with genetic programming,” *Genetic Programming*, pp. 256–267, 2001.

[144] J. Sherrah, R. Bogner, and B. Bouzerdoum, “Automatic selection of features for classification using genetic programming,” in *Australian and New Zealand Conference on Intelligent Information Systems*, 1996, pp. 284–287.

[145] D. Atkins, K. Neshatian, and M. Zhang, "A domain independent genetic programming approach to automatic feature extraction for image classification," in *IEEE Congress on Evolutionary Computation*, 2011, pp. 238–245.

[146] H. Guo, L. Jack, and A. Nandi, "Feature generation using genetic programming with application to fault classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 35, no. 1, pp. 89–99, 2005.

[147] Y. Zhang and P. Rockett, "Feature extraction using multi-objective genetic programming," *Multi-objective machine learning*, pp. 75–99, 2006.

[148] U. Watchareeruetai, T. Matsumoto, Y. Takeuchi, and N. OHNISHI, "Multi-objective genetic programming with redundancy-regulations for automatic construction of image feature extractors," *IEICE transactions on Information and Systems*, vol. 93, no. 9, pp. 2614–2625, 2010.

[149] T. Liddle, M. Johnston, and M. Zhang, "Multi-objective genetic programming for object detection," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1–8.

[150] G. Olague and L. Trujillo, "Interest point detection through multiobjective genetic programming," *Applied Soft Computing*, vol. 12, no. 8, pp. 2566–2582, 2012.

[151] T. Lee and D. Mumford, "Hierarchical bayesian inference in the visual cortex," *JOSA A*, vol. 20, no. 7, pp. 1434–1448, 2003.

[152] P. Viola and M. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[153] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *IEEE Computer Vision and Pattern Recognition*, vol. 2, 2001, pp. 416–423.

[154] J. Wu and J. Rehg, "Centrist: A visual descriptor for scene categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1489–1501, 2011.

[155] S. Luke and L. Panait, "Lexicographic parsimony pressure," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2002, pp. 829–836.

[156] G. McLachlan and D. Peel, *Finite mixture models*, 2004.

[157] B. Cheng, J. Yang, S. Yan, Y. Fu, and T. Huang, "Learning with ℓ^1-graph for image analysis," *IEEE Transactions on Image Processing*, pp. 858–866, 2010.

[158] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1, pp. 37–52, 1987.

[159] A. M. Martínez and A. C. Kak, "Pca versus lda," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 228–233, 2001.

[160] D. Cai, X. He, and J. Han, "Semi-supervised discriminant analysis," in *IEEE International Conference on Computer Vision*, 2007, pp. 1–7.

[161] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–38, 1977.

[162] J. A. Bilmes *et al.*, “A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models,” *International Computer Science Institute*, vol. 4, no. 510, p. 126, 1998.

[163] L. Zhang, P. Zhu, Q. Hu, and D. Zhang, “A linear subspace learning approach via sparse coding,” in *IEEE International Conference on Computer Vision*, 2011, pp. 755–761.

[164] S. Yan and H. Wang, “Semi-supervised learning by sparse representation,” in *SIAM International Conference on Data Mining*, 2009, pp. 792–801.

[165] J. J. Hull, “A database for handwritten text recognition research,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[166] A. Torralba, R. Fergus, and W. T. Freeman, “80 million tiny images: A large data set for nonparametric object and scene recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1958–1970, 2008.

[167] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, “Graph embedding and extensions: a general framework for dimensionality reduction,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40–51, 2007.

[168] A. Fischer and C. Igel, “An introduction to restricted boltzmann machines,” in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, 2012, pp. 14–36.

[169] J. Ni, R. H. Drieberg, and P. I. Rockett, “The use of an analytic quotient operator in genetic programming,” *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 1, pp. 146–152, 2013.

[170] U. Bhowan, M. Zhang, and M. Johnston, “Genetic programming for image classification with unbalanced data,” in *International Conference on Image and Vision Computing*, 2009, pp. 316–321.

[171] J. R. Koza, *Genetic Programming: On the programming of computers by means of natural selection*. MIT press, 1992, vol. 1.

[172] A. Gionis, P. Indyk, R. Motwani *et al.*, “Similarity search in high dimensions via hashing,” in *VLDB*, vol. 99, 1999, pp. 518–529.

[173] J. Wang, S. Kumar, and S.-F. Chang, “Sequential projection learning for hashing with compact codes,” in *International Conference on Machine Learning*, 2010, pp. 1127–1134.

[174] H. Jegou, M. Douze, and C. Schmid, “Hamming embedding and weak geometric consistency for large scale image search,” in *European Conference on Computer Vision*, 2008, pp. 304–317.

[175] L. Zheng, S. Wang, Z. Liu, and Q. Tian, “Packing and padding: Coupled multi-index for accurate image retrieval,” *CoRR*, vol. abs/1402.2681, 2014.

[176] W. Zhou, Y. Lu, H. Li, and Q. Tian, “Scalar quantization for large scale image search,” in *ACM Multimedia*, 2012, pp. 169–178.

[177] G. Tolias, Y. S. Avrithis, and H. Jégou, “To aggregate or not to aggregate: Selective match kernels for image search,” in *International Conference on Computer Vision*, 2013, pp. 1401–1408.

[178] C. Wengert, M. Douze, and H. Jégou, “Bag-of-colors for improved image search,” in *ACM Multimedia*, 2011, pp. 1437–1440.

[179] G. H. Hardy, J. E. Littlewood, and G. Pólya, *Inequalities.* Cambridge university press, 1952.

[180] G. Griffin, A. Holub, and P. Perona, “Caltech-256 object category dataset,” 2007.

[181] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, “Nus-wide: a real-world web image database from national university of singapore,” in *Proceedings of the ACM international conference on image and video retrieval*, 2009, p. 48.

[182] S. Kim, Y. Kang, and S. Choi, “Sequential spectral learning to hash with multiple representations,” in *European Conference on Computer Vision*, 2012, pp. 538–551.

[183] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua, “Ldahash: Improved matching with smaller descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 66–78, 2012.

[184] B. Kulis and T. Darrell, “Learning to hash with binary reconstructive embeddings,” in *Conference on Neural Information Processing Systems*, 2009, pp. 1042–1050.

[185] M. Norouzi and D. M. Blei, “Minimal loss hashing for compact binary codes,” in *International Conference on Machine Learning*, 2011, pp. 353–360.