sensors

Article

Fast Approximation for Sparse Coding with Applications to Object Recognition

Zhenzhen Sun and Yuanlong Yu *

The College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China; zhenzhen_sun@foxmail.com
* Correspondence: yu.yuanlong@fzu.edu.cn

Abstract: Sparse Coding (SC) has been widely studied and shown its superiority in the fields of signal processing, statistics, and machine learning. However, due to the high computational cost of the optimization algorithms required to compute the sparse feature, the applicability of SC to real-time object recognition tasks is limited. Many deep neural networks have been constructed to low fast estimate the sparse feature with the help of a large number of training samples, which is not suitable for small-scale datasets. Therefore, this work presents a simple and efficient fast approximation method for SC, in which a special single-hidden-layer neural network (SLNNs) is constructed to perform the approximation task, and the optimal sparse features of training samples exactly computed by sparse coding algorithm are used as ground truth to train the SLNNs. After training, the proposed SLNNs can quickly estimate sparse features for testing samples. Ten benchmark data sets taken from UCI databases and two face image datasets are used for experiment, and the low root mean square error (RMSE) results between the approximated sparse features and the optimal ones have verified the approximation performance of this proposed method. Furthermore, the recognition results demonstrate that the proposed method can effectively reduce the computational time of testing process while maintaining the recognition performance, and outperforms several state-of-the-art fast approximation sparse coding methods, as well as the exact sparse coding algorithms.

Keywords: sparse coding; fast approximation; homotopy iterative hard thresholding; object recognition

Citation: Sun, Z.; Yu, Y. Fast Approximation for Sparse Coding with Applications to Object Recognition. *Sensors* **2021**, *21*, 1442. https://doi.org/10.3390/s21041442

Academic Editor: Antonio Guerrieri
Received: 19 December 2020
Accepted: 17 February 2021
Published: 19 February 2021

1. Introduction

Object recognition is a fundamental problem in machine learning, and has been widely researched for many years. The performance of object recognition methods largely relies on feature representation. Traditional methods used handcrafted features to represent objects, i.e., scale-invariant feature transform (SIFT) [1], histograms of oriented gradients (HOG) [2], etc. Inspired by biological finding [3,4], learning sparse representation is more beneficial for object recognition, because mapping features from low-dimensional space to a high-dimensional space makes the features more likely to be linearly separable. Therefore, many sparse coding (SC) algorithms have been proposed to learn a good sparse representation for natural signals [5–7].

In general, SC is the problem of reconstructing input signal using a linear combination of an over-complete dictionary with sparse coefficients, i.e., for an observed signal $\boldsymbol{x} \in R^p$ an over-complete dictionary $\boldsymbol{D} \in R^{p \times K} (p \ll K)$, SC aims to find a representation $\boldsymbol{\alpha} \in R^K$ to reconstruct $\boldsymbol{x}$ by using only a small number of atoms chosen from $\boldsymbol{D}$. The problem of SC is formulated as

$$\min_{\boldsymbol{\alpha}} : ||\boldsymbol{x} - \boldsymbol{D}\boldsymbol{\alpha}||_2^2 + \lambda ||\boldsymbol{\alpha}||_0, \quad (1)$$

where the l_0-norm is defined as the number of non-zero elements of $\boldsymbol{\alpha}$, and λ is the regularization factor. Several optimization algorithms have been proposed for the numerical solution of (1). However, the high computational cost induced by these optimization algorithms is a major drawback for real-time applications, especially when a large-sized dictionary is used.

To get rid of this problem, many works focusing on fast approximation for sparse coding have been proposed. Kavukcuoglu et al. [8] proposed a method named Predictive Sparse Decomposition (PSD) that used a non-linear regressor to approximate the sparse feature, and applied this method to objection recognition. However, the predictor is simplistic and produces crude approximation, and the regressor training procedure is somewhat time-consuming because of the gradient descent training method. Recently, deep learning showed its widespread success on many inference problems, which provides another way to design fast approximation methods for sparse coding algorithms. The idea is first proposed by Gegor et al. [9] who constructed two deep learning networks to approximate the iterative soft thresholding and coordinate descent algorithms, leading to the so-called LISTA and LCoD methods, respectively. LISTA showed its superiority on calculation and approximation, and many recent variants of LISTA have been proposed for miscellaneous applications, see [10,11] for some examples. Inspired by [9], many fast approximation sparse coding methods based on deep learning have been proposed and shown their effectiveness on unfolding the corresponding sparse coding algorithms, i.e., LAMP [12], LVAMP [12], etc.

Though these methods perform well in large-scale datasets, there are three defects. First, they are not suitable for small-scale datasets, in which the number of training samples is far less then ten thousand. The performance of deep neural network is sensitive to the scale of training data, when the number of training samples is small, the deep network model is over-parameterized and may result in over-fitting. Second, deep networks involve lots of hyper-parameters, whose training requires large computational and storage resources because of the gradient-based back-propagation method, and is easy to get stuck in a local optimal solution. Last but not least, each deep network architecture is designed only for the corresponding sparse coding algorithm that cannot be generalized to other algorithms. Therefore, the extendibility of these methods are limited.

To solve the problems mentioned above, a simple and effective fast approximation sparse coding method is proposed for small-scale datasets object recognition task in this paper. Differing from the deep learning-based methods, a special single-hidden-layer neural network (SLNNs) is constructed to perform the approximation task, and the training process of this SLNNs can be easily implemented by the least squared method. The proposed method includes two steps. In the first step, the optimal sparse features of training samples are exactly computed by sparse coding algorithm (in this paper, the homotopy iterative hard thresholding (HIHT) algorithm [13] is used), and in the second step the optimal sparse features are used as ground truth to train the especially constructed SLNNs. After training, the input layer and hidden layer of this SLNNs can be used to implement the nonlinear feature mapping from the input space to sparse feature space, which only involves simple inner product calculation with a non-linear activation function. Therefore, the sparse features of new samples can be estimated quickly. Ten benchmark datasets taken from UCI databases and two face image datasets are used to validate the proposed method, and the root mean square error (RMSE) results on testing data have verified the approximation performance of this proposed method. Furthermore, the approximated sparse features have been applied to object recognition task, and the recognition results demonstrate that this proposed approximation sparse coding method is beneficial for object recognition in terms of recognition accuracy and testing time.

The main contributions of this paper can be concluded as

1. A fast approximation sparse coding method is proposed for small-scale datasets object recognition task, which can quickly estimate the sparse features for testing samples.
2. A special SLNNs architecture has been constructed to perform the approximation task, whose parameters can be optimized easily by the least squared method, avoiding the multifarious procedure induced by the gradient-based back-propagation training.

3. Experiment results on ten benchmark UCI datasets and two face image datasets show that our approach is more effective than current state-of-the-art deep learning-based fast approximation sparse coding methods both in RMSE, recognition accuracy and testing time.

The remainder of this paper is organized as follows. Section 2 briefly reviews the sparse coding algorithms and fast approximation sparse coding methods. Section 3 details the proposed method. Section 4 describes implementation details and presents experimental results. Finally, conclusions are given in Section 5.

2. Related Work

2.1. Sparse Coding Algorithms

As described in Section 1, the problem of SC can be formulated as problem (1). However, problem (1) is NP-hard, which is difficult to be solved. There are three common methods for approximations/relaxations of this problem: (1) iterative greedy algorithms [14–16]; (2) l_1-norm convex relaxation methods (which are called basis pursuit) [17]; (3) l_p-norm ($0 < p < 1$) relaxation methods [18–22]. Among these methods, BP has been studied more widely, in which the l_0 norm is replaced by l_1 norm to make a convex relaxation for the problem (1), i.e.,

$$\min_{\boldsymbol{\alpha}} : ||\boldsymbol{x} - \boldsymbol{D}\boldsymbol{\alpha}||_2^2 + \lambda||\boldsymbol{\alpha}||_1, \tag{2}$$

where the l_1-norm is defined as the sum of absolute values of all elements of $\boldsymbol{\alpha}$.

BP methods were proven to give the same solutions to (1) when the dictionary satisfies the Restricted Isometry Property (RIP) condition [23,24]. Many research works focusing on efficiently solving problem (2) have been proposed, [25] provides a comprehensive review of five representative algorithms, namely *Gradient Projection* (GP) [26,27], *Homotopy* [28,29], *Iterative Soft Shrinkage-Thresholding Algorithm* (ISTA) [30–32], *Proximal Gradient* (PG) [33,34], and *Augmented Lagrange Multiplier* (ALM) [35]. Among these algorithms, ISTA is the most popular algorithm, and lots of heuristic strategies have been proposed to reduce the computational time of ISTA, i.e., TwIST [36], FISTA [33], etc. Recently, a kind of pathwise coordinate optimization method called PICASSO [37–39] has been proposed to solve the l_p ($0 < p \leq 1$) least squared problem, which showed superior empirical performance compared with other state-of-the-art sparse coding algorithms mentioned above.

Although satisfactory results can be achieved by using the approximation/relaxation methods, the l_0-norm is more desirable from the sparsity perspective. In recent years, researchers have attempted to solve problem (1) directly, with iterative hard thresholding (IHT) [13,40,41] being the most popular method. The IHT methods have strong theoretical guarantees, and the extensive experimental results show that the IHT methods can improve the sparse representation reconstruction results.

2.2. Fast Approximation for Sparse Coding

The sparse coding algorithms mentioned in Section 2.1 involve a lot of iterative operations, which induces high computational cost and prohibits them from real-time applications. To get rid of this problem, some research focusing on fast approximation for sparse coding was proposed. Kavukcuoglu et al. [8] proposed the PSD method to approximate sparse coding algorithms using a non-linear regressor. In inspired by this, Chalasani et al. [42] extended PSD to estimate convolutional sparse features. However, the approximation performance of non-linear regressor is limited. As the development of deep learning, some researchers have constructed deep networks to solve the fast approximation sparse coding problem. Given a large set of training examples $\{(\boldsymbol{x}_i, \boldsymbol{\alpha}_i)\}_{i=1}^N$, a many-layer neural network is optimized to minimize the reconstruction mean squared error between network outputs and $\{\boldsymbol{\alpha}_i\}_{i=1}^N$. After training, the approximation of sparse representation for a new signal $\boldsymbol{x}_{new}$ can be quickly predicted by the deep network. The idea is first proposed by Gregor et al. [9] who constructed two deep learning networks to approximate the iterative soft thresholding and coordinate descent algorithms, leading to the so-called LISTA and LCoD methods, respectively. Inspired by [9], Xin et al. [43] translated the iterative hard thresholding algorithm into a deep learning framework. Borgerding et al. [12] proposed two deep neural-network architectures to unfold the approximate message passing (AMP) algorithm [44] and "vector AMP" (VAMP) algorithm [45] respectively, namely LAMP and LVAMP. In [46], the authors proposed a deep learning framework for the approximation of sparse representation of a signal with the aid of a correlated signal, the so-called side information. The learned deep networks perform steps similar to those implemented by corresponding sparse coding algorithms; however, the trained network can reduce the computational cost when calculating the sparse representation of new samples effectively, which is critical in large-scale data settings and real-time applications.

3. Materials and Methods

3.1. Homotopy Iterative Hard Thresholding Algorithm

The homotopy iterative hard thresholding (HIHT) [13] is an extension of IHT for the l_0-norm regularized problem

$$\min_{\boldsymbol{\alpha}} : \theta_\lambda(\boldsymbol{\alpha}) = f(\boldsymbol{\alpha}) + \lambda||\boldsymbol{\alpha}||_0, \tag{3}$$

where $f(\boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{x} - \boldsymbol{D\alpha}||_2^2$ is a differentiable convex function, whose gradient $\nabla f(\boldsymbol{\alpha})$ satisfies the Lipschitz continuous condition with parameter $L_f > 0$. Therefore, $f(\boldsymbol{\alpha})$ can be approximately iteratively updated by the projected gradient method

$$\boldsymbol{\alpha}^{k+1} = argmin\ f(\boldsymbol{\alpha}^k) + \bigtriangledown f(\boldsymbol{\alpha}^k)^T(\boldsymbol{\alpha} - \boldsymbol{\alpha}^k) + \frac{L}{2}||\boldsymbol{\alpha} - \boldsymbol{\alpha}^k||_2^2, \tag{4}$$

where $L \geq 0$ is a constant, which should satisfies the condition of $L \geqslant L_f$.

Adding $\lambda||\boldsymbol{\alpha}||_0$ into both side of (4), the solution of (3) can be obtained by iteratively solving the subproblem

$$\begin{aligned}\boldsymbol{\alpha}^{k+1} &= arg\min\ p_{L,\lambda}(\boldsymbol{\alpha}^k, \boldsymbol{\alpha}) \\ &= f(\boldsymbol{\alpha}^k) + \nabla f(\boldsymbol{\alpha}^k)^T(\boldsymbol{\alpha} - \boldsymbol{\alpha}^k) + \frac{L}{2}||\boldsymbol{\alpha} - \boldsymbol{\alpha}^k||_2^2 + \lambda||\boldsymbol{\alpha}||_0.\end{aligned} \tag{5}$$

The optimization of (5) is the same as follows (by removing or adding some constant items which are independent on $\boldsymbol{\alpha}$):

$$\min_{\boldsymbol{\alpha}} \frac{L}{2}\left[||\boldsymbol{\alpha} - (\boldsymbol{\alpha}^k - \frac{1}{L}\nabla f(\boldsymbol{\alpha}^k))||_2^2 + \frac{2\lambda}{L}||\boldsymbol{\alpha}||_0\right]. \tag{6}$$

If denote

$$T_L(\boldsymbol{\alpha}^{k+1}) = arg\min_{\boldsymbol{\alpha}}\ p_{L,\lambda}(\boldsymbol{\alpha}^k, \boldsymbol{\alpha}), \tag{7}$$

then the closed form solution of $T_L(\boldsymbol{\alpha}^{k+1})$ is given by the following lemma.

Lemma 1. *[32,41] The solution $T_L(\boldsymbol{\alpha}^{k+1})$ of (7) is give by*

$$[T_L(\boldsymbol{\alpha}^{k+1})]_i = \begin{cases} [s_L(\boldsymbol{\alpha}^k)]_i, & if\ [s_L(\boldsymbol{\alpha}^k)]_i^2 > \frac{2\lambda}{L}; \\ 0, & if\ [s_L(\boldsymbol{\alpha}^k)]_i^2 \leq \frac{2\lambda}{L}. \end{cases} \tag{8}$$

where $s_L(\boldsymbol{\alpha}) = \boldsymbol{\alpha} - \frac{1}{L}\nabla f(\boldsymbol{\alpha})$, and $[.]_i$ refers to the i-th element of a vector.

In (8), the parameter L needs to be tuned. The upper bound on Lipschitz constant L_f is unknown or may not be easily calculated, thus we use the line search method to search L as suggested in [41] until the objective value descends.

Homotopy Strategy: many works [13,26,39] have verified that the sparse coding approaches benefit from a good starting point. Therefore, we use a recursive process automatically tunes regularization factor λ. This process begins from a large initial value λ^0. At the end of each λ-tuning iterations indexed by k, an optimal solution $\boldsymbol{\alpha}^k$ is obtained given λ^k. Then λ is updated as $\lambda^{k+1} = \rho\lambda^k$, where $\rho \in [0,1]$, and $\boldsymbol{\alpha}^k$ is used as the initial solution for the next iteration $k+1$. The process stops once λ is small enough (given a positive lower-bound target, the stop condition is $\lambda_k \leq \lambda_{target}$). An outline of HIHT algorithm is described as Algorithm 1.

Algorithm 1 $\{\boldsymbol{\alpha}^\star, L^\star\} \leftarrow HIHT(L_0, \lambda_0, \boldsymbol{\alpha}^0)$

```
(Input:) $L_0, \lambda_0, \boldsymbol{\alpha}^0, \boldsymbol{D}, L_{min}, L_{max}; // L_0 \in [L_{min}, L_{max}]$
(Output:) $\boldsymbol{\alpha}^\star, L^\star;$
initialize $\rho \in (0,1), \eta > 0, \gamma > 1, \epsilon > 0, k \leftarrow 0;$
repeat
  $i \leftarrow 0;$
  $\boldsymbol{\alpha}^{k,0} = \boldsymbol{\alpha}^k;$
  $L_{k,0} \leftarrow L_k;$
  repeat
   An L-tuning iteration indexed by i
    $\boldsymbol{\alpha}^{k,i+1} \leftarrow T_{L_{k,i}}(\boldsymbol{\alpha}^{k,i});$
    while $\theta_{\lambda_k}(\boldsymbol{\alpha}^{k,i}) - \theta_{\lambda_k}(\boldsymbol{\alpha}^{k,i+1}) < \frac{\eta}{2}||\boldsymbol{\alpha}^{k,i} - \boldsymbol{\alpha}^{k,i+1}||^2$ do
      $L_{k,i} \leftarrow min\{\gamma L_{k,i}, L_{max}\};$
      $\boldsymbol{\alpha}^{k,i+1} \leftarrow T_{L_{k,i}}(\boldsymbol{\alpha}^{k,i});$
    end while
    $L_{k,i+1} \leftarrow L_{k,i};$
    $i \leftarrow i+1;$
  until$||\boldsymbol{\alpha}^{k,i} - \boldsymbol{\alpha}^{k,i+1}||_2^2 \leq \epsilon$
  $\boldsymbol{\alpha}^{k+1} \leftarrow \boldsymbol{\alpha}^{k,i};$
  $L_{k+1} \leftarrow L_{k,i}.$
  $\lambda_{k+1} \leftarrow \rho\lambda_k;$
  $k \leftarrow k+1;$
until $\lambda_{k+1} \leq \lambda_{target}$
$\boldsymbol{\alpha}^\star \leftarrow \boldsymbol{\alpha}^k;$
$L^\star \leftarrow L_k.$
```

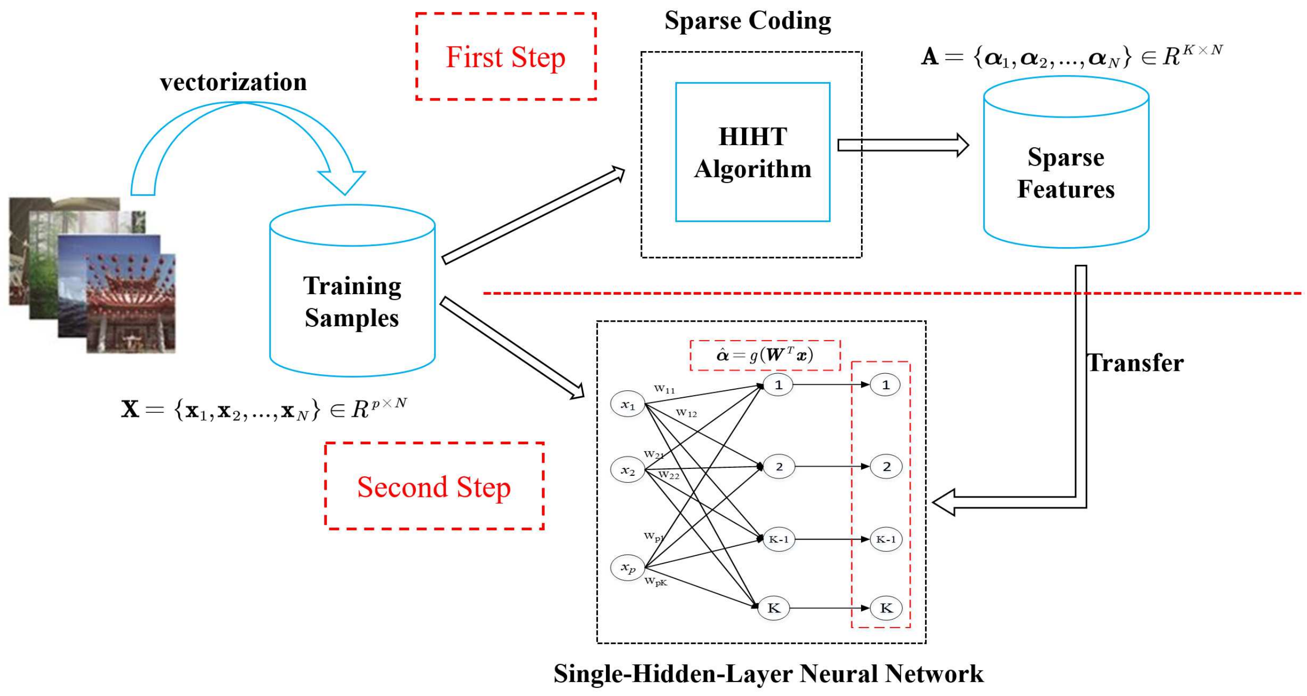
3.2. Proposed Method

Figure 1 illustrates the schematic diagram of this proposed method. As it can be seen, for the given training dataset $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\} \in R^{p*N}$ and the over-complete dictionary $\boldsymbol{D}$, the HIHT algorithm described in Section 3.1 is used to calculate the optimal sparse features $\boldsymbol{A} = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_N\} \in R^{K*N}$ of training data in the first step. After that, these optimal sparse features are used to train the SLNNs in second step.

As Figure 1 shows, the architecture of the neural network consists of an input layer, a feature layer and an output layer. The number of hidden neurons is the same as that of output neurons, which is set as the dimension of the sparse feature. Each hidden neuron is only connected to its corresponding output neuron with weight 1. Our goal is to obtain a optimal input weights $\hat{\boldsymbol{W}}$ to make the outputs of hidden layer as equal to $\boldsymbol{A}$ as possible, that is

$$||g(\hat{\boldsymbol{W}}^T\boldsymbol{X}) - \boldsymbol{A}||_F \to 0, \tag{9}$$

where $g(.)$ refers to a non-linear activation function.



First Step
Sparse Coding
$\mathbf{A} = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_N\} \in R^{K \times N}$
vectorization
HIHT Algorithm
Sparse Features
Training Samples
$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} \in R^{p \times N}$
Transfer
$\hat{\boldsymbol{\alpha}} = g(\boldsymbol{W}^T \boldsymbol{x})$
Second Step
Single-Hidden-Layer Neural Network

Figure 1. The schematic diagram of this proposed method.

There are two strategies to optimize the input weights $\boldsymbol{W}$:

(1) If the activation function is known, we chose *tanh* function as the activation function, where $g(x) = tanh(x)$. We firstly calculate $arctanh(\boldsymbol{A})$, and denote the result as $\boldsymbol{Z}$, that is

$$\boldsymbol{Z} = arctanh(\boldsymbol{A}), \tag{10}$$

then we formulate the objective function of the SLNNs as

$$\text{Minimize:} \quad \tfrac{1}{2}\|\boldsymbol{Z} - \boldsymbol{W}^T\boldsymbol{X}\|_F^2 + \tfrac{C_1}{2}\|\boldsymbol{W}\|_F^2, \tag{11}$$

where constant C_1 refers to the regularization factor used to control the trade-off between the smoothness of the mapping function and the closeness to $\boldsymbol{Z}$.

By setting the derivative of (11) with respect to $\boldsymbol{W}$ to zero and solve this equality, then the optimal solution of $\boldsymbol{W}$ is obtained as follows:

$$\hat{\boldsymbol{W}} = (\frac{\boldsymbol{I}}{C_1} + \boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{Z}^T. \tag{12}$$

In addition, the sparse feature of a testing sample $\boldsymbol{x}_{test}$ can be quickly estimated as

$$\hat{\boldsymbol{\alpha}}_{test} = tanh(((\frac{\boldsymbol{I}}{C_1} + \boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{Z}^T)^T\boldsymbol{x}_{test}). \tag{13}$$

(2) If the activation function is unknown, a kernel trick based on Mercer's condition can be used to calculate the approximated sparse feature of testing data $\boldsymbol{x}_{test}$ directly instead of training the weights $\boldsymbol{W}$,

$$\hat{\boldsymbol{\alpha}}_{test} = g(\boldsymbol{W}^T * \boldsymbol{x}_{\boldsymbol{test}}) = Ker(\boldsymbol{x}_{\boldsymbol{test}}, \boldsymbol{X})(\frac{\boldsymbol{I}}{C_1} + \Omega_{train})^{-1}\boldsymbol{A}, \tag{14}$$

where $\Omega_{train} == Ker(\boldsymbol{X}, \boldsymbol{X})$ and Ker stands for the kernel function.

In this proposed method, Gaussian function is used as the kernel function *Ker*:

$$Ker(\boldsymbol{x}_1, \boldsymbol{x}_2) = exp(-\frac{||\boldsymbol{x}_1 - \boldsymbol{x}_2||^2}{\sigma^2}), \tag{15}$$

where σ denotes the standard deviation of the Gaussian function.

4. Results and Discussion

4.1. Data Sets Description

Ten benchmark datasets taken from *UCI Machine Learning Repository* [47] and two image datasets: the Extended YaleB [48] and the AR dataset [49], are used to validate the proposed method. The ten UCI datasets include 5 binary-classification cases and 5 multi-classification cases. The details of these datasets are shown in Table 1. In this table, column "Random Perm" shows whether the training and testing data are randomly assigned or not. In the experiments, $\frac{2}{3}$ of samples per class are randomly selected for training, and the rest samples are responsible for testing if "Random Perm" is Yes.

Table 1. UCI Data sets Used in Our Experiments.

Datasets	Training	Testing	Features	Classes	Random Perm
Australian Credit	459	231	14	2	Yes
Diabetes	511	257	8	2	Yes
Glass	140	74	9	6	Yes
Image segmentation	1540	770	19	7	Yes
LiverDisorders	229	116	6	2	Yes
Madelon	2000	600	500	2	No
Satimage	4435	2000	36	6	No
Vehicle	562	284	18	4	Yes
Wine	118	60	13	3	Yes
Wisconsin Breast Cancer	379	190	30	2	Yes

The extended YaleB dataset [48] contains 38 different people with 2414 frontal face images, and each class has about 64 samples. This dataset is challenging from varying expressions and illumination conditions, see Figure 2 for some examples. The random face feature descriptor generated in [7] is used as raw feature, in which a cropped image with 192 × 168 pixels was projected onto a 504-dimensional vector by a random normal distributed matrix. In the experiment, 50% of samples per class are randomly selected for training and the rest are responsible for testing.

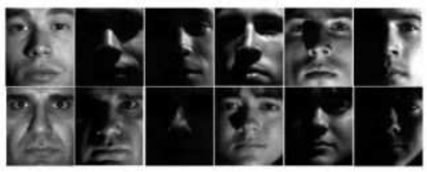


Figure 2. Extended YaleB.

The AR face dataset contains over 126 people with more than 4000 face images. There are 26 images per person taken during two different sessions. The images have large variations in terms of disguise, facial expressions, and illumination conditions. A few samples from the AR dataset are shown in Figure 3 for illustration. A subset of 2600 images pertaining to 50 males and 50 females objects are used for experiment. For each object, 20 samples are randomly chosen for training and the rest for testing. The images with 165 × 120 pixels were projected onto a 540-dimensional vector by using a random projection matrix.

Figure 3. AR Face.

4.2. Implementation Details

The experiments are mainly divided into two parts: (1) The RMSE between the approximated sparse features and the optimal features of testing data is calculated to verify the approximation performance of this proposed method, and the results of several state-of-the-art fast approximation sparse coding methods are also reported for comparison. (2) Classification experiments are implemented to validate the recognition performance of the approximated sparse features estimated by the proposed SLNNs. The compared methods can be categorized as follows: (a) Different representation learning methods: ELM [50] with random feature mapping, and ScELM [51] with optimal sparse features computed by HIHT; (b) Different fast approximation sparse coding methods: PSD [8], LISTA [9], LAMP [12], and LVAMP [12], detailed descriptions to these methods are provided in Section 2.

Implementations of ELM, ScELM, PSD, and this proposed method are based on Matlab codes and others are based on Python. A random normal distributed matrix is used as the dictionary in each sparse coding algorithm, and the number of atoms or hidden nodes K is set to 100 if the dimension of dataset is less than 100, otherwise 1000. The parameter C_1 is searched for in the grid of $\{2^{-25}, 2^{-20}, ..., 2^{25}\}$, and the σ is searched for in $\{100, 200, ..., 1000\}$. The number of hidden layers of LISTA, LAMP, and LVAMP are set as 6, 5, and 4, respectively, if not stated otherwise. Other parameters are default as the authors suggested. For the randomly training-testing assigned datasets, ten repeated trials are carried out in the following experiments, and the average result and standard deviation are recorded.

In object recognition experiments, the trained network of each method is used to compute the approximated sparse features for training and testing samples, and the approximated sparse features are used as the input of the classifier. The ridge regression model is used as the classifier in our experiments, whose objective function is

$$\text{Minimize:} \quad \tfrac{1}{2}\|\boldsymbol{Y} - \boldsymbol{\beta}^T \hat{\boldsymbol{A}}\|_F^2 + \tfrac{C_2}{2}\|\boldsymbol{\beta}\|_F^2, \tag{16}$$

where $\boldsymbol{Y}$ is the label matrix of training data $\boldsymbol{X}$, and $\boldsymbol{\beta}$ is the weights of the classifier model. For a testing sample $\boldsymbol{x}_{test}$, the predicted label for it is calculated as

$$identity(\boldsymbol{x}_{test}) = argmax_i(\boldsymbol{\beta}^T \hat{\boldsymbol{\alpha}}_{test}). \tag{17}$$

The hyper-parameter of the classifier C_2 is searched for in the grid of $\{2^{-25}, 2^{-20}, ..., 2^{25}\}$, and a value with best validation accuracy is selected. We compare our method with others in terms of recognition accuracy and testing time, where the recognition accuracy is defined as the ratio of the number of correctly classified testing samples to that of all testing samples, and the testing time refers to the total spending time of testing samples' feature calculation and classification.

A standard PC is used in our experiments and its hardware configuration as follows:

1. CPU: Intel(R) Pentium(R) CPU G2030 @3.40GHz;
2. Memory: 32.00GB;
3. Graphics Processing Unit (GPU): None.

4.3. Root Mean Square Error Results

For testing data $\boldsymbol{X}_{test}$, whose optimal sparse features computing by sparse coding algorithm is denoted as $\boldsymbol{A}_{test}$, and the approximated sparse features computing by the fast approximation method is denoted as $\hat{\boldsymbol{A}}_{test}$, the RMSE between $\boldsymbol{A}_{test}$ and $\hat{\boldsymbol{A}}_{test}$ is defined as

$$RMSE(\boldsymbol{A}_{test}, \hat{\boldsymbol{A}}_{test}) = \sqrt{\frac{1}{N_{test}K}||\boldsymbol{A}_{test} - \hat{\boldsymbol{A}}_{test}||_F^2}, \tag{18}$$

where N_{test} denotes the amount of testing samples.

Some UCI datasets are used in this experiment, and we reported the results of our method, LISTA, LAMP and LVAMP to compare their approximation performance, Table 2 shows the results. As it can be seen from this table, our approach can achieve a lower RMSE result than other methods on the most datasets, which indicates that the approximated sparse features estimated by our approach are more closer to the optimal ones than that estimated by the compared methods. For the Glass dataset, our method has achieved a significant improvement, and for LiverDisorders, though the result of our approach is not the best one, it is very close to the best one.

Table 2. The root mean square error of compared methods on UCI datasets (bold one represents the best result).

Datasets	**LISTA**	**LAMP**	**LVAMP**	**Ours**	
				Tanh	**Kernel**
Australian Credit	0.0418	0.0614	0.0497	0.0591	**0.0391**
Diabetes	0.0460	0.0356	0.0446	0.0494	**0.0350**
Glass	0.0527	0.0436	0.0550	**0.0111**	0.0144
Image segmentation	0.0324	0.0406	0.0462	0.0375	**0.0317**
LiverDisorders	0.0440	**0.0403**	0.0435	0.0476	0.0597
Vehicle	0.0267	0.0442	0.0496	0.0141	**0.0122**
Wine	0.0415	0.0394	0.0462	0.0149	**0.0094**
Wisconsin Breast Cancer	0.0295	0.0551	0.0427	0.0222	**0.0179**

4.4. Objection Recognition Results

4.4.1. The Evaluation of HIHT

The existing literature on sparse coding only compared different sparse coding algorithms in terms of reconstruction error and convergence speed, but did not compare their classification performance when applying these algorithms in object recognition. To show why this paper uses the HIHT algorithm to compute the optimal sparse features, we implemented some experiments to validate the superiority of HIHT compared with several state-of-the-art sparse coding algorithms when used in object recognition. The compared methods include IHT, homotopy GPSR (HGPSR) [26], PGH [34], and PICASSO [39].

(1) *Effectiveness on Object Recognition*: the binary-classification datasets listed in Table 1 are used in this experiment. Firstly, the sparse coding algorithms are used to compute sparse features for the experimental datasets using the same dictionary, and the measure of *cross entropy* is used to show how different the sparse features are between class 1 and class 2. A higher value means that the sparse features computed by corresponding algorithm are more discriminative and more beneficial for object recognition. The measure of *cross entropy* is estimated as follows: we accumulate a histogram $\boldsymbol{h}(\boldsymbol{\alpha}_k|v)$ along feature dimensions over all sparse features $\boldsymbol{\alpha}_k$ that belongs to the same class v ($v \in \{1,2\}$), then normalize the histogram as the probability $\boldsymbol{p}(v)$ of class v, the *cross entropy* between class 1 and class 2 is estimated as

$$cross\ \ entropy(\boldsymbol{p}(1), \boldsymbol{p}(2)) = -\sum_{k=1}^{K} p_k(1) \log \frac{1}{p_k(2)}, \tag{19}$$

where $p_k(v)$ is the p-th element of the probability $\boldsymbol{p}(v)$.

Table 3 shows the *cross entropy* results. It can be seen that the HIHT algorithm can achieve the best result on the most datasets than the other four algorithms. It indicated that the sparse features computed by HIHT can distinguish different classes more effectively, which is more useful for classification, especially when a simple linear classifier is used.

Table 3. Cross entropy of sparse features between different classes.

Datasets	HGPSR	IHT	PGH	PICASSO	HIHT
Austrain Credit	6.840	6.785	6.933	6.299	**7.268**
Diabetes	4.381	6.164	3.899	4.118	**6.410**
LiverDisorders	3.087	5.461	3.267	5.894	**6.346**
Madelon	8.040	9.335	8.076	9.280	**9.680**
Wisconsin Breast Cancer	4.652	**6.256**	4.643	5.632	5.878

Subsequently, we use these sparse coding algorithms to compute the optimal sparse features of training data to train the proposed SLNNs, and compare the final recognition results, which is shown in Table 4. From this table it can be seen that these sparse coding algorithms can achieve similar classification performance on most datasets when used in the proposed method, while HIHT outperforms the other three algorithms in some datasets (i.e., Glass and Vehicle) significantly. From the view of standard deviation, the results show that the optimal sparse features computed by HIHT are more robust to classification than other algorithms.

Table 4. Recognition Results of the Proposed Method Using different sparse coding algorithms.

Datasets	HGPSR	IHT	PICASSO	HIHT
Australian Credit	85.59 ± 1.83	85.87 ± 1.87	85.67 ± 1.87	86.08 ± 1.73
Diabetes	76.06 ± 1.96	76.73 ± 1.68	74.96 ± 2.28	76.99 ± 1.66
Glass	60.97 ± 4.83	61.64 ± 3.38	64.59 ± 4.63	66.60 ± 3.20
Image segmentation	90.08 ± 1.02	90.81 ± 0.97	90.91 ± 0.89	91.90 ± 0.73
LiverDisorders	68.59 ± 3.62	72.34 ± 4.27	69.57 ± 3.96	73.52 ± 2.56
Madelon	59.67	60.88	58.33	60.67
Satimage	82.12	83.37	83.00	84.30
Vehicle	78.46 ± 2.47	78.68 ± 2.07	76.20 ± 1.81	81.00 ± 1.51
Wine	96.80 ± 1.92	98.53 ± 1.47	97.33 ± 1.81	98.56 ± 1.07
Wisconsin Breast Cancer	93.87 ± 1.68	95.31 ± 1.09	96.00 ± 2.89	95.92 ± 1.81

(2) *Parameter Sensitivity*: In HIHT algorithm, different values of the regularization factor λ_{target} and dictionary $\boldsymbol{D}$ will product different sparse features, which will cause the proposed method to estimate different approximated sparse features and influence final recognition result. In this experiment, the sensitivities of λ_{target} and $\boldsymbol{D}$ in final recognition performance are verified, and the two face image datasets are used for testing.

Firstly, the influence of λ_{target} is investigated. By fixing other parameters (i.e., dictionary, parameters of the classifier), λ_{target} is searched for in the grid of $\{10^{-10}, 10^{-8}, ..., 10^{2}\}$, and the corresponding recognition accuracy is recorded. From the results in Figure 4, we can conclude that the final recognition result is not very sensitive to the λ_{target}, so it is no need to spend much time turning λ_{target} when uses HIHT to compute the optimal sparse features in this proposed method.

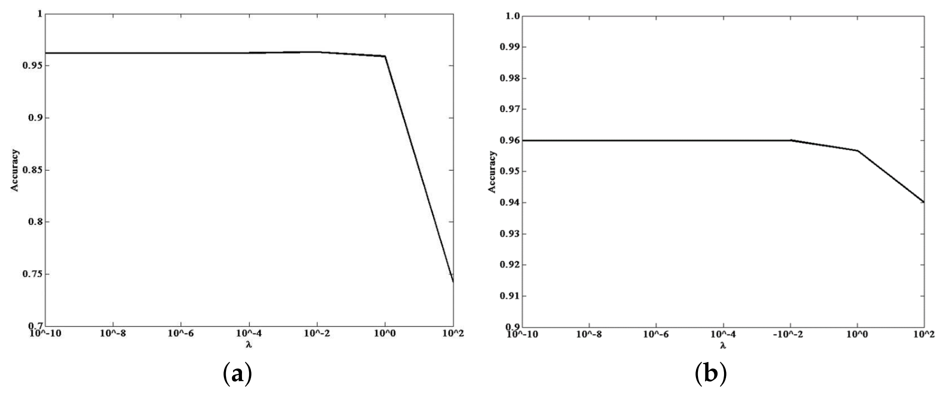


Figure 4. The influence of the target value λ_{target} of HIHT on the recognition performance. (**a**) Extended YaleB. (**b**) AR Face.

Subsequently, we investigate the influence of $\boldsymbol{D}$. An unsupervised learned dictionary by the Lagrangian dual method [52] is used to compare with a random dictionary generated by normal distribution. The number of iterations in dictionary learning is set as 5, and 10 times with random selection of training and testing data are repeated, the average accuracy is recorded for comparison. As Table 5 shows, the final recognition accuracy achieved by using learned dictionary are close to that by using random dictionary. However, the computational time of optimal sparse features calculation with dictionary learning is five times (equal to the number of iterations) that with the random dictionary. Thus, in the following experiments we use random dictionary to compute optimal sparse features in HIHT algorithm.

Table 5. Comparison of final recognition performance with or without dictionary learning in HIHT algorithm.

Datasets	Learned Dictionary	Random Dictionary
YaleB	95.95 ± 0.65	96.34 ± 0.69
AR	96.63 ± 0.77	97.37 ± 0.39

4.4.2. Evaluation on UCI Datasets

The average recognition accuracies on UCI datasets are listed in Tables 6 and 7 presents the testing time. From these two tables we can conclude that the proposed approach outperforms other methods in terms of accuracy and testing time simultaneously. For most datasets, the approximated sparse features estimated by our approach can obtain the highest accuracy, and is approximately 100 times faster than ScELM (exact sparse coding algorithm), especially in high-dimensional datasets. Compared with other approximation sparse coding methods, our approach can achieve higher recognition accuracy with simpler network training, and the testing time of the proposed method and PSD are much less than LISTA, LAMP and LVAMP. It is worth noting that the performances of activation function *tanh* and kernel function of this approach are similar, but kernel function outperforms *tanh* when the dataset is a litter complex, (i.e., Satimage, Madelon), which will be confirmed in next experiments.

Table 6. The average accuracy of compared methods on UCI datasets (red is the best result and blue is the second one).

Datasets	ELM	ScELM	PSD	LISTA	LAMP	LVAMP	Ours	
							Tanh	Kernel
Australian Credit	86.13	83.52	86.22	81.97	80.26	-	86.08	86.15
Diabetes	65.32	69.26	65.01	73.98	68.19	74.59	76.99	75.14
Glass	57.30	62.38	63.46	67.50	60.50	61.75	66.60	67.23
Image segmentation	77.32	88.77	89.30	88.13	87.39	90.63	91.90	91.86
LiverDisorders	67.45	65.52	69.17	70.34	73.39	66.10	73.52	74.25
Madelon	58.17	59.17	59.35	51.67	50.92	56.93	60.67	64.83
Satimage	71.70	80.25	78.55	78.10	77.60	74.25	84.30	89.15
Vehicle	73.99	75.30	78.85	74.22	78.61	72.83	81.00	80.56
Wine	95.13	94.00	95.00	93.25	96.83	94.29	98.56	98.00
Wisconsin Breast Cancer	87.54	94.40	94.80	93.23	91.56	93.75	95.92	96.48

Table 7. The testing time of compared methods on UCI datasets.

Datasets	ELM	ScELM	PSD	LISTA	LAMP	LVAMP	Ours	
							Tanh	Kernel
Australian Credit	0.0079	0.6696	2.006×10^{-4}	0.0728	0.0811	0.0728	3.181×10^{-4}	0.0033
Diabetes	0.0136	1.7844	6.704×10^{-5}	0.0756	0.1332	0.0927	3.536×10^{-4}	0.0057
Glass	0.0013	0.9932	5.282×10^{-5}	0.0527	0.0714	0.0505	1.276×10^{-4}	6.061×10^{-4}
Image segmentation	0.0029	22.226	2.258×10^{-4}	0.1833	0.4511	0.4998	9.128×10^{-4}	0.0365
LiverDisorders	0.0011	0.7055	6.198×10^{-5}	0.0797	0.2499	0.5070	2.052×10^{-4}	9.227×10^{-4}
Madelon	0.0494	148.59	0.0062	1.2150	1.2560	0.9656	0.0293	0.073
Satimage	0.0134	39.218	7.130×10^{-4}	0.4272	0.3601	0.3287	0.0028	0.3018
Vehicle	0.0015	1.2062	1.013×10^{-4}	0.1249	0.3214	0.0991	4.392×10^{-4}	0.0047
Wine	7.232×10^{-7}	0.2457	5.687×10^{-5}	0.1042	0.0783	0.0556	4.392×10^{-4}	4.291×10^{-4}
Wisconsin Breast Cancer	0.0021	2.7363	$2.086e \times 10^{-4}$	0.0625	0.0929	0.0731	3.099×10^{-4}	0.0134

Figure 5 (The Tanh and Kernel mean the *tanh* version and kernel version of our method, respectively.) shows the confusion matrices obtained by this proposed method, PSD, LISTA, LAMP and LVAMP on Satimage dataset, in which the kernel version of this proposed method achieved a much better recognition result than others. It can be seen from this figure that all methods almost fail to correctly classify the test samples of class 4 except the kernel version of our method. It indicates that the features computed by this proposed method are more discriminative than that of other approximation methods. Figure 6 shows two examples of the receiver operating characteristic (ROC) curves of the approximation methods, where the red lines report the performance of our approach. It is clear that the Areas Under ROC curves (AUC) of our approach is much higher than others.

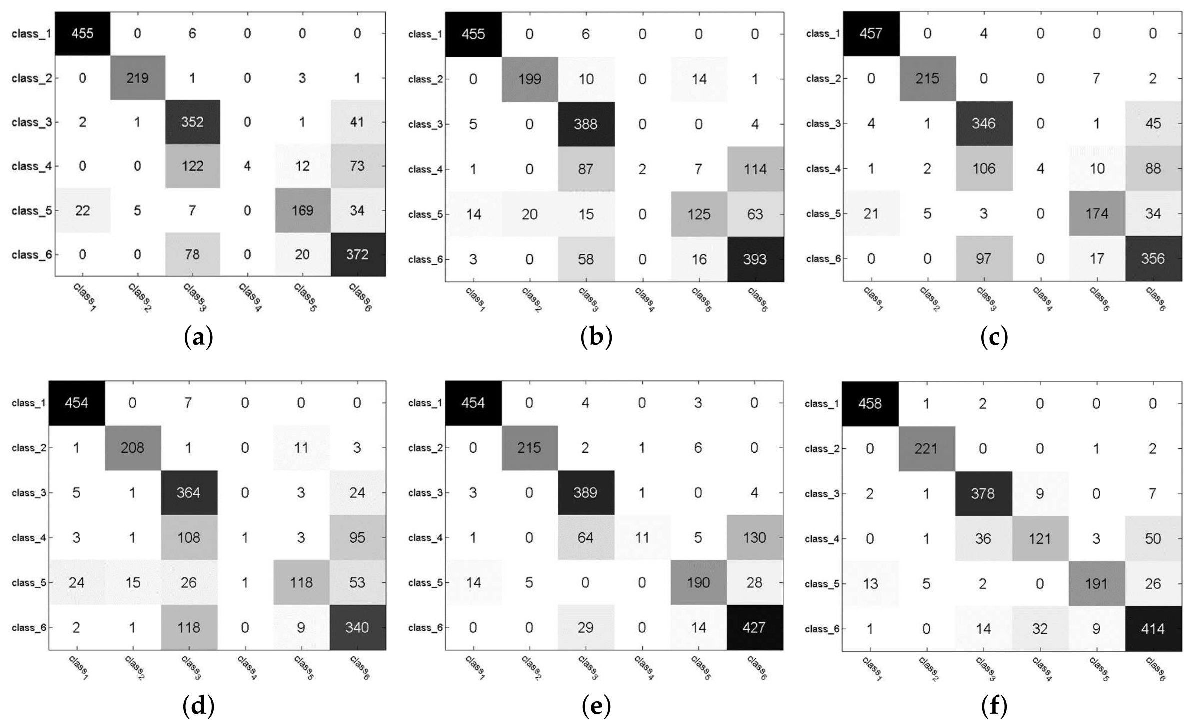


Figure 5. Confusion matrices on Satimage dataset. (**a**) PSD. (**b**) LISTA. (**c**) LAMP. (**d**) LVAMP. (**e**) Tanh. (**f**) Kernel.

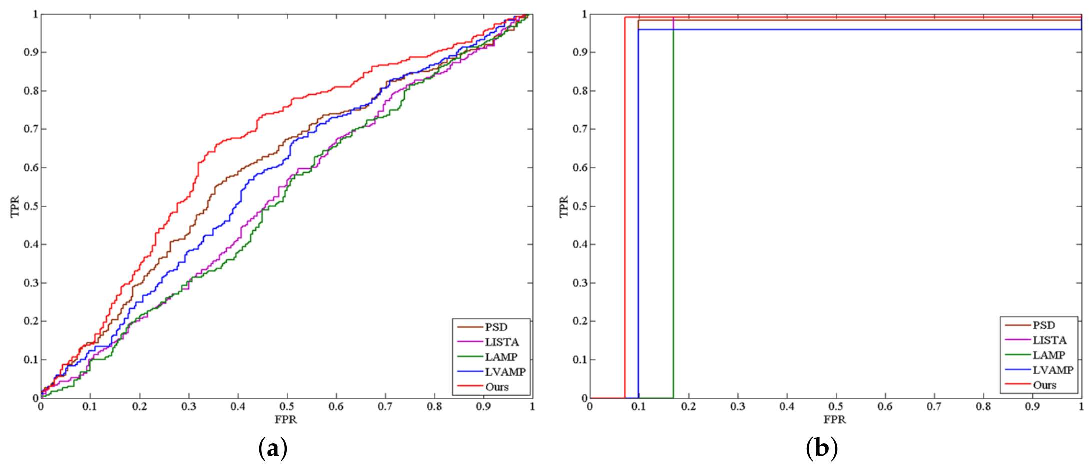


Figure 6. The receiver operating characteristic (ROC) curves of the approximate methods. (**a**) Madelon. (**b**) Breast.

4.4.3. Evaluation on Extended YaleB Dataset

Table 8 lists the recognition accuracies and testing time on Extended YaleB dataset, in which the famous sparse representation-based face recognition algorithm SRC [53] and collaborative representation-based classification (CRC) [54] are also used for comparison. Furthermore, a result obtained by raw features is set as the error bar (denoted as Baseline), and we set different number of hidden layers (denoted as T) for LISTA to show its influence on object recognition performance. As Table 8 shows, all methods beat the Baseline, indicating the benefit of feature learning. The kernel version of this proposed method obtains the best result with the value of 98.33%, and is 1.89% higher than the second one. In testing process, the proposed method is approximately 21 times faster than the deep learning-based approximation methods, and 182 times faster than SRC, also much faster than CRC. For LISTA, if the number of layers is small ($T = 2$), the recognition performance will degrade much, and as the number of layers increases, the recognition results tend to be stable. Thus, the recognition performance is somewhat sensitive to the number of layers of deep network.

Figure 7 shows the patterns of confusion across classes obtained by this proposed method, in which coordinates in X-axis and Y-axis represent 38 face classes. Color at coordinates (x, y) represents the number of test samples whose ground truth are x while machine's output labels are y. From this figure it can be seen that our approach shows fewer points in the non-diagonal region (i.e., fewer false positives and false negatives), indicting that the proposed method can classify most testing samples correctly.

Table 8. Average Recognition Accuracy with Random-Face Features on the Extended YaleB Database.

Method	Accuracy (%)	Time (s)
Baseline	91.54 ± 1.14	0.0009
SRC	96.27 ± 0.64	16.1676
CRC	96.82 ± 0.58	1.5006
ELM	96.44 ± 0.60	0.0256
ScELM	92.43 ± 0.93	168.3158
PSD	93.01 ± 0.69	0.057
LISTA (T = 2)	92.19 ± 1.21	2.0129
LISTA (T = 6)	95.16 ± 1.18	2.2535
LISTA (T = 10)	95.34 ± 0.82	4.6385
LAMP	94.07 ± 0.72	2.0119
LVAMP	94.62 ± 1.12	1.900
Tanh	96.34 ± 0.69	0.0256
kernel	**98.33 ± 0.40**	0.0888

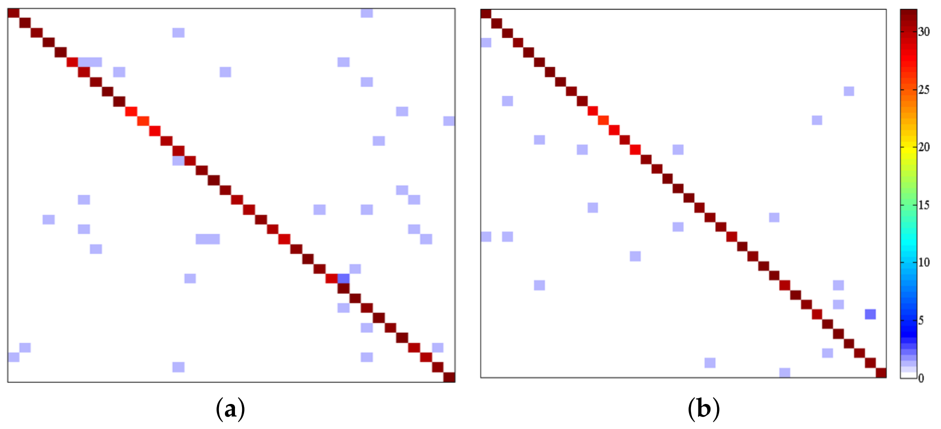


Figure 7. Patterns of confusion on Extended YaleB. (**a**) Tanh. (**b**) Kernel.

4.4.4. Evaluation on AR Dataset

For the AR dataset, a protocol (e.g., only five training samples per class or all training samples are used) is established in our experiments, and the corresponding results are list in Table 9. As we can see, the kernel version of this proposed method achieves the best result in both cases. In addition, the *tanh* version of this method gets comparable result with LAMP and ELM, but still better than SRC when all training samples were used. In terms of testing time, the proposed method is approximately 12 times faster than the deep learning-based approximation methods, 300 times faster than SRC, and 24 times faster than CRC. It is worth noting that the computational speed of kernel version is a little slower than that of *tanh* version, since it needs to compute the kernel matrix between testing samples and training samples, while it is still much faster than the deep learning-based approximation methods.

We use a confusion matrix to give the detailed evaluation at the class-level. Figure 8 shows the results, in which coordinates in x- and y-axis denote 100 face classes. Red point with coordinates (x, y) represents the misclassified test samples. It can be seen from this figure that this proposed method shows rare points in the non-diagonal region than other methods, indicating that this proposed method performs better than other methods in object recognition.

Table 9. Average Recognition Accuracy with Random-Face Features on the AR Face Database. The four column is the result when only 5 training samples per class are used.

Method	Accuracy (%)	Time (s)	Accuracy (%)
Baseline	93.00 ± 0.84	0.0011	78.67
SRC	95.42 ± 0.59	27.1421	84.00
CRC	96.92 ± 0.76	1.9682	84.83
ELM	97.05 ± 0.78	0.0178	83.67
ScELM	94.50 ± 1.09	62.1872	73.50
PSD	96.80 ± 0.66	0.0305	77.17
LISTA (T=3)	95.57 ± 0.85	0.8888	-
LISTA (T=6)	96.47 ± 0.61	1.2335	80.33
LISTA (T=8)	96.80 ± 0.73	3.1065	-
LAMP	97.37 ± 0.59	1.0505	81.67
LVAMP	95.30 ± 0.97	1.0314	84.50
Tanh	97.37 ± 0.39	0.0189	85.33
Kernel	**98.40 ± 0.46**	0.0815	**86.67**

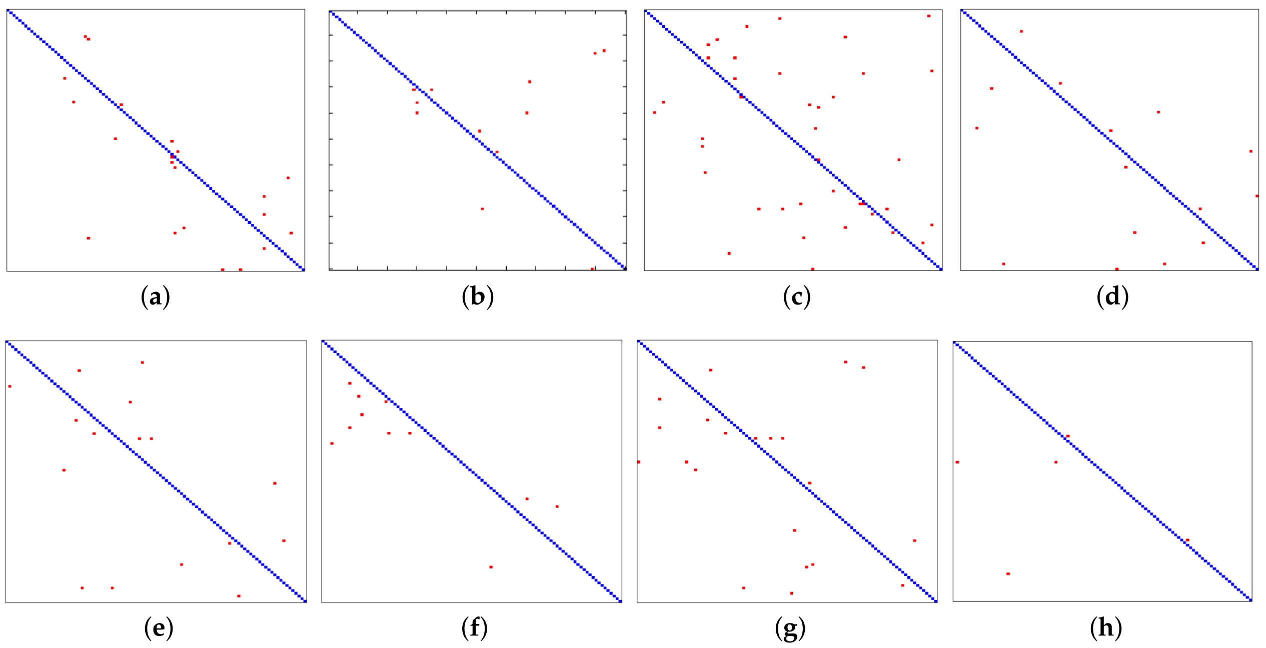


Figure 8. Patterns of confusion on AR dataset. (**a**) SRC. (**b**) CRC. (**c**) ScELM. (**d**) PSD. (**e**) LISTA. (**f**) LAMP. (**g**) LVAMP. (**h**) Kernel.

To give an intuitive illustration, Figure 9 shows all misclassified images obtained by LAMP method (which achieves the second best result) and this proposed method (kernel version). It can be seen that images with exaggerated facial expressions is the main reason causing misclassification for both methods. Another interesting point can be seen that most images with facial "disguises" are misclassified by LAMP method while they are correctly recognized by this proposed method. It indicates that the approximate sparse features estimated by this proposed method is robustness to facial occlusion or corruption than LAMP.

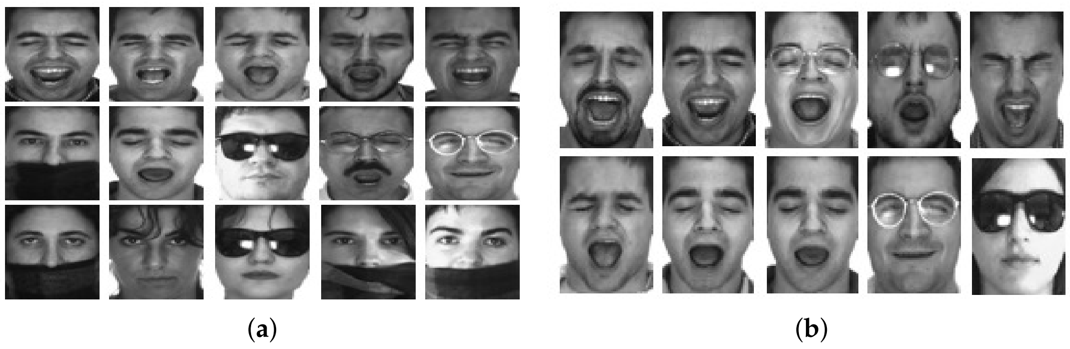


Figure 9. All misclassified images produced by (**a**) LAMP and (**b**) Kernel.

5. Conclusions

This paper proposes a simple fast approximation sparse coding method for small-scale datasets object recognition task, in which the optimal sparse features of training data computed by HIHT algorithm are used as ground truth to train a succinct and special SLNNs, thus make the representation learning in object recognition task more practical and efficient. Extensive experimental results on publicly available datasets show that this approach outperforms the compared approximation methods in terms of approximation performance, recognition accuracy and computational time. The high recognition and computational efficiency makes the proposed method very promising for real-time applications. Moreover, experimental results have demonstrated that this proposed method is robust to parameters on recognition performance, that make it more practical. Future work includes supervised sparse coding algorithms and autonomously finding an over-complete dictionary.

Author Contributions: Conceptualization, Z.S. and Y.Y.; methodology, Z.S.and Y.Y.; validation, Z.S.; data curation, Z.S.; writing—original draft preparation, Z.S.; writing—review and editing, Z.S. and Y.Y.; funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (NSFC) under Grant No. 61873067.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. These datasets can be found here: http://archive.ics.uci.edu/ml/index.php.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lowe, D.G. Distinctive Image Features from Scale-Invariant Key-points. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
2. Dalal, N.; Triggs, B. Histograms of Oriented Gradients for Human Detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Diegol, CA, USA, 20–26 June 2005. [CrossRef]
3. Hubel, D.; Wiesel, T. Receptive fields of signal neurons in the cat's striate cortex. *J. Physiol.* **1959**, *148*, 574–591. [CrossRef] [PubMed]
4. Roll, E.; Tovee, M. Sparseness of the neuronal representation of stmuli in the primate temporal visual cortex. *J. Neurophysiol.* **1992**, *173*, 713–726. [CrossRef]
5. Mairal, J.; Bach, F.; Ponce, J.; Sapiro, G.; Zisserman, A. Discriminative learned dictionaries for local image analysis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, 23–28 June 2008. [CrossRef]

6. Wang, J.; Yang, J.; Yu, K.; Lv, F.; Huang, T.; Gong, Y. Locality-constrained Linear Coding for image classification. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 3360–3367. [CrossRef]
7. Jiang, Z.; Lin, Z.; Davis, L. Label consistent K-SVD: Learning a discriminative dictionary for recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 2651–2663. [CrossRef]
8. Kavukcuoglu, K.; Ranzato, M.; LeCun, Y. Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition. In *Technical Report CBLL-TR-2008-12-01, Computational and Biological Learning Lab, Courant Institute, NYU*; New York University: New York, NY, USA, 2008.
9. Gregor, K.; LeCun, Y. Learning fast approximations of sparse coding. In Proceedings of the International Conference on Machine Learning, Haifa, Israel, 21–24 June 2010.
10. Deng, X.; Dragotti, P.L. Deep Coupled ISTA Network for Multi-Modal Image Super-Resolution. *IEEE Trans. Image Process.* **2020**, *29*, 1683–1698. [CrossRef]
11. Qian, Y.; Xiong, F.; Qian, Q.; Zhou, J. Spectral Mixture Model Inspired Network Architectures for Hyperspectral Unmixing. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 7418–7434. [CrossRef]
12. Borgerding, M.; Schniter, P.; Rangan, S. Amp-inspired deep networks for sparse linear inverse problem. *IEEE Trans. Signal Process.* **2017**, *65*, 4293–4348. [CrossRef]
13. Dong, Z.; Zhu, W. Homotopy methods based on l_0-norm for compressed sensing. *IEEE Trans. Neural Networks Learn. Syst.* **2018**, *29*, 1132–1146. [CrossRef]
14. Mallat, S.; Zhang, Z. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Process.* **1993**, *41*, 3397–3415. [CrossRef]
15. Pati, Y.; Rezaiifar, R.; Krishnaprasad, P. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In Proceedings of the IEEE International Conference on Signal, Systems and Computers, Pacific Grove, CA, USA, 1–3 November 1993; pp. 40–44. [CrossRef]
16. Loza, C. RobOMP: Robust variants of Orthogonal Matching Pursuit for sparse representations. *PeerJ Comput. Sci.* **2019**, *5*, e192. [CrossRef]
17. Chen, S.; Donoho, D.; Saunders, M. Atomic decomposition by basis pursuit. *SIAM Rev.* **2001**, *43*, 129–159. [CrossRef]
18. Chartrand, R. Exact Reconstruction of Sparse Signals via Nonconvex Minimization. *IEEE Signal Process. Lett.* **2007**, *14*, 707–710. [CrossRef]
19. Xu, Z.; Chang, X.; Xu, F.; Zhangm, H. $L_{1/2}$ regularization: A thresholding representation theory and a fast solver. *IEEE Trans. Neural Networks Learn. Syst.* **2012**, *23*, 1013–1027. [CrossRef]
20. Chen, X.; Ng, M.K.; Zhang, C. Non-Lipschitz ł$_p$-Regularization and Box Constrained Model for Image Restoration. *IEEE Trans. Image Process.* **2012**, *21*, 4709–4721. [CrossRef] [PubMed]
21. Qin, L.; Lin, Z.C.; She, Y.; Chao, Z. A comparison of typical L_p minimization algorithms. *Neurocomputing* **2013**, *119*, 413–424. [CrossRef]
22. Qiu, Y.; Jiang, H.; Ching, W.; Ng, M.K. On predicting epithelial mesenchymal transition by integrating RNA-binding proteins and correlation data via $L_{1/2}$-regularization method. *Artif. Intell. Med.* **2019**, *95*, 96–103. [CrossRef]
23. Donoho, D.; Elad, M. Optimally sparse representation in general (nonorthogonal) dictionaries via l^1 minimization. *Proc. Natl. Acad. Sci. USA* **2003**, *100*, 2197–2202. [CrossRef]
24. Candes, E.J.; Tao, T. Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies? *IEEE Trans. Inf. Theory* **2006**, *52*, 5406–5425. [CrossRef]
25. Yang, A.; Ganesh, A.; Zhou, Z.; Sastry, S.; Ma, Y. A Review of Fast l_1 -Minimization Algorithm for Robust Face Recognition. In Proceedings of the International Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 12–15 July 2010; pp. 1–36.
26. Figueiredo, M.; Nowak, R.; Wright, S. Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and other Inverse Problems. *IEEE J. Sel. Top. Signal Process.* **2007**, *1*, 586–597. [CrossRef]
27. Kim, S.J.; Koh, K.; Boyd, S. An Interior-Point Method for Large-Scale l_1 -Regularized Least Squares. *IEEE J. Sel. Top. Signal Process.* **2007**, *1*, 606–617. [CrossRef]
28. Osborne, M.R.; Presnell, B.; Turlach, B.A. A new approach to variable selection in least squares problems. *IMA J. Numer. Anal.* **2000**, *20*, 389–404. [CrossRef]
29. Malioutov, D.M.; Cetin, M.; Willsky, A.S. Homotopy continuation for sparse signal representation. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Philadelphia, PA, USA, 18–23 March 2005. [CrossRef]
30. Combettes, P.; Wajs, V. Signal recovery by proximal forward-backward splitting. *SIAM Multiscale Model. Simul.* **2005**, *4*, 1168–1200. [CrossRef]
31. Hale, E.; Yin, W.; Zhang, Y. *A Fixed-Point Continuation Method for l_1-Regularized Minimization with Applications to Compressed Sensing*; CAAM Tech Report TR07-07; Rice University: Houston, TX, USA, 7 July 2007; pp. 1–45. [CrossRef]
32. Wright, S.J.; Nowak, R.D.; Figueiredo, M.A. Sparse reconstruction by separable approximation. *IEEE Trans. Signal Process.* **2009**, *57*, 2479–2493. [CrossRef]
33. Beck, A.; Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.* **2009**, *2*, 183–202. [CrossRef]

34. Lin, X.; Tong, Z. A proximal-gradient homotopy method for the sparse least-squares problem. *SIAM J. Optim.* **2013**, *23*, 1062–1091. [CrossRef]
35. Yang, J.; Zhang, Y. Alternating direction algorithms for l_1-problems inbcompressive sensing. *SIAM J. Sci. Comput.* **2011**, *31*, 250–278. [CrossRef]
36. Bioucas-Dias, J.M.; Figueiredo, M.A.T. A New TwIST: Two-Step Iterative Shrinkage/Thresholding Algorithms for Image Restoration. *IEEE Trans. Image Process.* **2007**, *16*, 2992–3004. [CrossRef]
37. Li, X.G.; Ge, J.; Jiang, H.M.; Wang, M.D.; Hong, M.Y.; Zhao, T. *Boosting Pathwise Coordinate Optimization in High Dimensions: Sequential Screening and Proximal Sub-Sampled Newton Algorithm*; Technical Report; Georgia Tech: Atlanta, GA, USA, 2017.
38. Zhao, T.; Liu, H.; Zhang, T. Pathwise coordinate optimization for nonconvex sparse learning: Algorithm and theory. *Ann. Stat.* **2018**, *46*, 180–218. [CrossRef]
39. Ge, J.; Li, X.; Jiang, H.; Liu, H.; Zhang, T.; Wang, M.; Zhao, T. Picasso: A sparse learning library for high dimensional data analysis in R and Python. *J. Mach. Learn. Res.* **2019**, *20*, 1–5.
40. Blumensath, T.; Davies, M. Iterative thresholding for sparse approximations. *Fourier Anal. Appl.* **2008**, *14*, 629–654. [CrossRef]
41. Lu, Z. Iterative hard thresholding methods for l_0 regularized convex cone programming. *Math. Program.* **2014**, *147*, 125–154. [CrossRef]
42. Chalasani, R.; Principe, J.C.; Ramakrishnan, N. A fast proximal method for convolutional sparse coding. In Proceedings of the 2013 International Joint Conference on Neural Networks (IJCNN), Dallas, TX, USA, 4–9 August 2013; pp. 1–5. [CrossRef]
43. Xin, B.; Wang, Y.; Gao, W.; Wipf, D.; Wang, B. Maximal sparsity with deep networks? In *Advances in Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 4340–4348.
44. Donoho, D.L.; Maleki, A.; Montanari, A. Message passing algorithms for compressed sensing. *Proc. Natl. Acad. Sci. USA* **2009**, *106*, 18914–18919. [CrossRef]
45. Rangan, S.; Schniter, P.; Fletche, A.K. Vector approximate message passing. *arXiv* **2016**, arXiv:1610.03082.
46. Tsiligianni, E.; Deligiannis, N. Deep coupled-representation learning for sparse linear inverse problems with side information. *IEEE Signal Process. Lett.* **2019**, *26*, 1768–1772. [CrossRef]
47. Dua, D. and Graff, C. *UCI Machine Learning Repository*; University of California: Oakland, CA, USA, 2017.
48. Georghiades, A.; Belhumeur, P.; Kriegman, D. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intell.* **2001**, *23*, 643–660. [CrossRef]
49. Martinez, A.; Benavente, R. *The AR Face Database*; Tech. Rep; Comput. Vis. Center, Purdue University: West Lafayette, IN, USA, 1998.
50. Huang, G.; Zhou, H.; Ding, X.; Zhang, R. Extreme learning machine for regression and multiclass classification. *IEEE Trans. Syst. Man, Cybern. Part Cybern.* **2012**, *42*, 513–529. [CrossRef]
51. Yu, Y.; Sun, Z. Sparse coding extreme learning machine for classification. *Neurocomputing* **2017**, *261*, 50–56. [CrossRef]
52. Lee, H.; Battle, A.; Raina, R.; Ng, A. Efficient sparse coding algorithm. In *Advances in Neural Information Processing Systems*; Curran Associates Inc.: Vancouver, BC, Canada, 2006; pp. 801–808.
53. Wright, J.; Yang, A.; Ganesh, A.; Sastry, S.; Ma, Y. Robust Face Recognition via Sparse Representation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 210–227. [CrossRef] [PubMed]
54. Zhang, L.; Yang, M.; Feng, X. Sparse representation or collaborative representation: Which helps face recognition? In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 471–478. [CrossRef]