Feature Learning via Partial Differential Equation with Applications to Face Recognition

Cong Fang[a,b], Zhenyu Zhao[c], Pan Zhou[a], Zhouchen Lin[a,b,*]

[a] *Key Lab. of Machine Perception (MOE), School of EECS, Peking University, P. R. China*
[b] *Cooperative Medianet Innovation Center, Shanghai Jiaotong University, P. R. China*
[c] *Department of Mathematics, School of Science, National University of Defense Technology, P. R. China*

Abstract

Feature learning is a critical step in pattern recognition, such as image classification. However, most of the existing methods cannot extract features that are discriminative and at the same time invariant under some transforms. This limits the classification performance, especially in the case of small training sets. To address this issue, in this paper we propose a novel Partial Differential Equation (PDE) based method for feature learning. The feature learned by our PDE is discriminative, also translationally and rotationally invariant, and robust to illumination variation. To our best knowledge, this is the *first* work that applies PDE to feature learning and image recognition tasks. Specifically, we model feature learning as an evolution process governed by a PDE, which is designed to be translationally and rotationally invariant and is learned via minimizing the training error, hence extracts discriminative information from data. After feature extraction, we apply a linear classifier for classification. We also propose an efficient algorithm that optimizes the whole framework. Our method is very effective when the training samples are few. The experimental results of face recognition on the four benchmark face datasets show that the proposed method outperforms the state-of-the-art feature learning methods in the case of small training samples.

Keywords: feature learning, partial differential equation, face recognition.

1. Introduction

Nowadays, many well-known methods for image classification tasks (e.g. face recognition) involve two steps: feature extraction and classification. As the performance of the classifier is heavily dependent on the quality of features (or data representation), much of the effort on image classification goes

*Corresponding author.
Email addresses: fangcong@pku.edu.cn (Cong Fang), dwightzzy@gmail.com (Zhenyu Zhao), pzhou@pku.edu.cn (Pan Zhou), zlin@pku.edu.cn (Zhouchen Lin)

into the design of features and data transformations [1]. The approaches to feature extraction can be split into two categories: manually designing features and automatically learning features.

Manual feature design is a way that incorporates human ingenuity and prior knowledge to represent data. Features extracted by existing popular methods, such as Scale-Invariant Feature Transform (SIFT) [2], Histogram of Oriented Gradients (HOG) [3], and Invariant Scattering Convolution Networks [4], usually satisfy some invariance properties, e.g., translational and rotational invariance, that are beneficial to the image classification tasks. They are intuitive and fit for various image classification tasks relatively well. However, inventing these methods is extremely labor-intensive, and existing methods may not extract discriminative information from the data well. So researchers gradually turn to learn representations of data.

Linear representation based feature learning methods have attracted much attention recently. This is because images of convex and Lambertian objects taken under distant illumination lie near an approximately nine-dimensional linear subspace, known as the harmonic plane [5]. By utilizing this subspace property, Low Rank Representation [6] based methods extract feature to capture the global structure of the whole data and are robust to noise. Chen et al. [7] extract the low rank matrix as feature and then apply Sparse Representation Classification (SRC) [8] for classification. Li et al. [9] propose a semi-supervised framework with class-wide diagonal structure to learn low-rank representations. Zhang et al. [10] expand the low-rank model into a dictionary learning method. Wu et al. [11] also apply a low-rank dictionary model into multi-view tasks. Dictionary learning methods, which learn a set of representation atoms and weighted coefficients (feature) at the same time, have also achieved huge success. Zhang et al. [12] propose a discriminative KSVD method (D-KSVD) which combines the dictionary reconstruction error and classification error and then solve their model by a single KSVD. Mairal et al. [13] model the supervised dictionary learning as a bilevel optimization framework. To build the relationship between dictionary atoms and the class labels, Jiang et al. [14] associate label information with each dictionary item and propose a Label Consistent K-SVD method (LC-KSVD). Liu et al. [15] also propose an oriented-discriminative dictionary to tackle this problem. There are also some works which construct several different dictionaries for classification. Ou et al. [16] use an occlusion dictionary for face recognition with occlusion. Liu et al. [17] apply a bilinear dictionary for face recognition. However, these linear representation based feature learning methods ignore the invariance of the features. For example, in face recognition tasks the changes of illumination or poses can only be regarded as noise. Moreover, since a little misalignment among faces can bring down the performance of classification significantly, much effort is spent on aligning the faces before

classification [18].

Deep neural networks, which are composed of multiple nonlinear transformations, have shown their superiority during the past few years [19, 20, 21]. Their hierarchical structure is effective in extracting discriminative information. Convolutional Neural Networks (CNN) [22] cut down the connections between the successive layers by using shared weights (same filters) and apply pooling strategies to extract local useful features, which have achieved an amazing performance [21] in image classification tasks. However, deep neural networks usually need a huge number of samples for training. Unfortunately, for many problems, such as tasks in bioinformatics and face recognition, each class only has several samples for training.

Recently, Liu et al. [23, 24] have proposed a framework that learns partial differential equations (PDEs) from training image pairs, which has been successfully applied to several computer vision and image processing problems. In [24], they apply learning-based PDEs to object detection, color2gray, and demosaicking. In [25], they model the saliency detection task as learning a boundary condition of a PDE system. Zhao et al. [26] extend this model to text detection.

The incapability of the existing methods in incorporating both discrimination and invariance into features motivates us to find new ways to feature learning, *especially in the case of limited training samples*. Considering that symmetry methods for differential equations can construct invariances rigorously, in this paper we propose a novel PDE model for feature learning. An illustration of the proposed approach is shown in Figure 1. The PDE is formulated as a linear combination of fundamental differential invariants. The evolution process of the PDE works as a mapping from the raw images to the features of the same dimension. Distinguished from traditional PDE methods, our PDE is data-driven, enhancing discriminative information in the learned feature. In addition, its evolution process is strictly translationally and rotationally invariant. Then the feature is fed to a simple linear classifier for classification. We also provide an algorithm that updates the parameters alternately to optimize our discretized model. By utilizing the invariance property well, our method is very efficient when the training samples are few. We summarize the contributions of this paper as follows:

- We propose a novel PDE based method to extract image feature for classification. We model the feature extraction process as an evolutionary PDE. The learned feature is both discriminative and invariant under translation, rotation and gray-level scaling. To our best knowledge, this is the *first* work that applies PDE to feature learning and image recognition.

- We provide a simple yet effective algorithm to optimize our discretized PDE model. The whole

training time in each experiment is less than five minutes[1].

Face recognition is a paradigm where the training samples are few. Our experimental results[2] on the four well-known public face recognition datasets show that our method outperforms the state-of-the-art methods in this case. For example, we obtain a recognition accuracy of 96% on Extended Yale B, with only 10 samples for each person, which is about 9% higher than sparse coding and dictionary learning methods.

The rest of the paper is structured as follows: we will first introduce our PDE model in Section 2. In Section 3, we provide our algorithm to optimize our model. We discuss some other related works in Section 4. In Section 5, we evaluate our PDE model on face recognition tasks and show the superiority of our model. Finally, we will conclude our paper in Section 6.

Table 1: Notations (Nota. stands for notation.)

Nota.	Description	Nota.	Description
u	Evolution of the input image.	$\mathrm{vec}(\cdot)$	Rearrange a matrix to a column vector.
Ω	An open bounded region in $\mathbf{R}^2$.	$\|\|\cdot\|\|_F$	Frobenious norm, $\|\|X\|\|_F = \sqrt{\sum_{i,j} X_{ij}^2}$.
$\partial\Omega$	Boundary of Ω.	I_m, h_m	The m-th training image and its tag vector.
Q	$\Omega \times [0,T]$.	$\{a_i(t)\}_{i=0}^5$	Parameters in the PDE.
Γ	$\partial\Omega \times [0,T]$.	A, W	Parameters in the PDE and classifier.
∇u	Gradient of u.	X^T	Transpose of matrix (or vector).
$\mathbf{H}_u$	Hessian of u.	$\langle\cdot,\cdot\rangle$	Inner product, $\langle C, D\rangle = (\mathrm{vec}(C))^T \mathrm{vec}(D)$.

2. PDE Based Feature Learning Model

In this section, we present our PDE model for discriminative feature learning. We first propose the general framework and then crystallize our model via some invariance properties. To begin with, we provide in Table 1 a brief summary of the notations used throughout the paper. For vector x, x_i presents its i-th component.

[1] We will post our code online once our paper is accepted.

[2] Currently, our method focuses on low-resolution images. To best of our knowledge, all the compared methods which aim at classification with limited training samples also solve images at this scale.

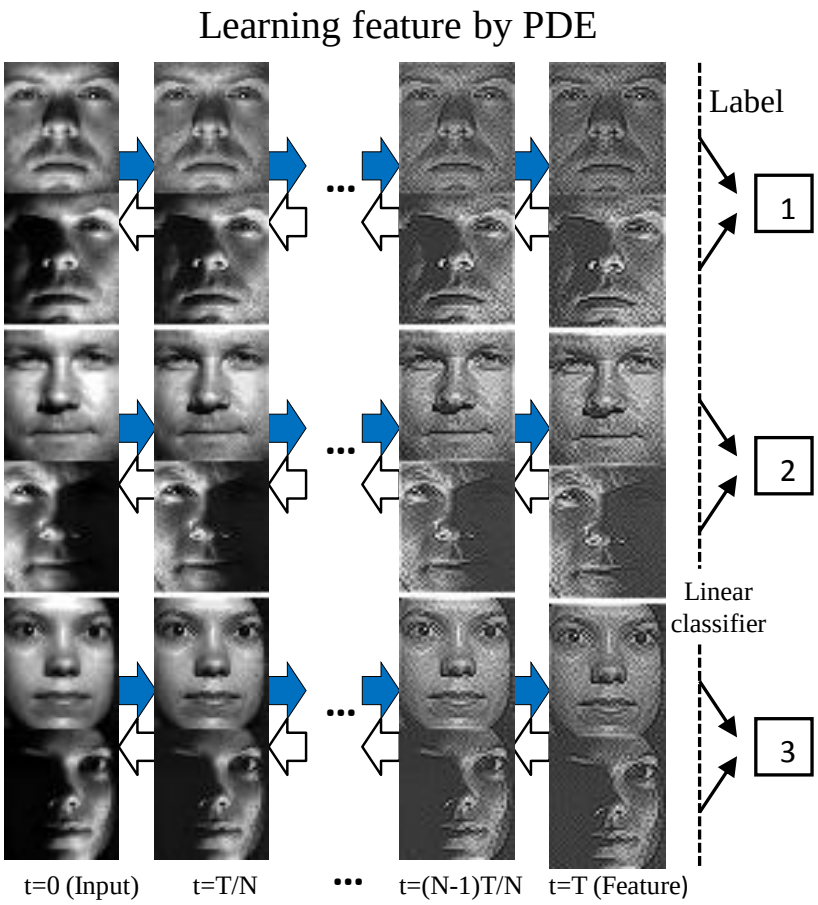

Figure 1: Illustration of the proposed approach. The evolutionary process of our PDE (solid arrow) with respect to the time ($t = 0, T/N, \cdots, T,$) extracts the feature from the image and the gradient descent process (hollow arrow) learns a transform to represent the feature.

2.1. General PDE Model

We first assume that feature extraction is an evolution process which can be described by a certain kind of time-dependent PDE. The input of the PDE (initial condition) is the original image. The output of the PDE is the feature of the image. The time-dependent operations of the evolutionary PDE resemble different steps of information processing. The PDE can be formulated as:

$$\begin{cases} \frac{\partial u}{\partial t} = F(u, x, y, t), & (x, y, t) \in Q, \\ u(x, y, t) = 0, & (x, y, t) \in \Gamma, \\ u|_{t=0}(x, y, t) = I, & (x, y) \in \Omega, \end{cases} \tag{1}$$

Table 2: Rotational invariants up to the second order.

i	$\text{inv}_i(u)$
$0, 1, 2$	$1, u, \|\nabla u\|^2 = u_x^2 + u_y^2,$
3	$tr(\mathbf{H}_u) = u_{xx} + u_{yy},$
4	$(\nabla u)^T \mathbf{H}_u \nabla u = u_x^2 u_{xx} + 2u_x u_y u_{xy} + u_y^2 u_{yy},$
5	$tr(\mathbf{H}_u^2) = u_{xx}^2 + 2u_{xy}^2 + u_{yy}^2.$

where I is the input image, Ω is the rectangular region occupied by the input image I, and T is the time that the PDE finishes feature extraction[3]. The evolution result $u|_{t=T}$ is the learned feature map. The meanings of other notations in Eq. (1) can be found in Table 1. So when the PDE is discretized, which will be discussed in Subsection 3.1, the dimension (size) of the feature map $u|_{t=T}$ will be the same as the input image I.

2.2. Formulate the PDE

The $F(u, x, y, t)$ in (1) is unknown. For most existing evolutionary PDE methods for image processing tasks [27, 28], the PDEs can be written as follows:

$$\frac{\partial u}{\partial t} = F(u, \nabla u, \mathbf{H}_u), \tag{2}$$

where F is a function of $u, \nabla u$, and $\mathbf{H}_u$. Different choices of F result in different PDEs. For some image processing problems, people can use their intuition (e.g. smoothness of edge contour and surface shading) to devise a particular F. But for classification tasks, it is hard to directly write down an F which can describe the feature extraction process. Inspired by [24], we tend to deduce the property of F in order to narrow down its search space instead of directly finding the right form the PDE.

2.2.1. Translational and rotational invariants

For many image classification tasks, the features need to be invariant under some transformations so as to make the classification robust. The most basic transformations are translation and rotation. Some existing manually designed features, such as SIFT and HOG, are roughly invariant under translation and rotation. Inspired by [24], we also require our PDE to be translationally and rotationally invariant over time. According to the differential invariant theory [29], $F(\cdot, \cdot, \cdot, t)$ must be a function of the fundamental differential invariants under the group of translation and rotation. The fundamental

[3]When discretizing the PDE, we pad images with zeros so as to satisfy the Dirichlet boundary conditions $u(x, y, t) = 0$, where $(x, y, t) \in \Gamma$.

differential invariants are invariants under translation and rotation and any other invariant can be written as their function. The fundamental invariants up to the second order[4] that we will use are listed in Table 2, which we refer to as $\text{inv}_i(u)$, $i = 0, \cdots, 5$.

To verify that the $\text{inv}_i(u)$, $i = 0, \cdots, 5$, are invariant under rotation, it is not hard to find that $\nabla u, \mathbf{H}_u$ will change to $\mathbf{R}\nabla u$ and $\mathbf{R}\mathbf{H}_u\mathbf{R}^T$, repectively, when the image is rotated by a matrix $\mathbf{R}$.

2.2.2. Nonlinear mapping

In many image classification tasks, such as face recognition, variation of illumination is a big challenge [8]. To achieve approximate invariance in illumination, we add a nonlinear mapping $g(x) = \frac{x}{1+|x|}$ on each fundamental differential invariant, making it nearly invariant under gray-level scaling. Note that we cannot use $\tilde{g}(x) = \frac{x}{|x|} = \text{sgn}(x)$ because it is not a bijection. So $\{\tilde{g}(\text{inv}_i(u))\}_{i=0}^5$ are not fundamental differential invariants that can be used to represent other differential invariants. In contrast, $\{g(\text{inv}_i(u))\}_{i=0}^5$ are still fundamental differential invariants. In the same spirit, $g(x)$ can be chosen as other commonly used transfer function in neural networks, such as the logistic function [19, 20]. But $g(x)$ here is much simpler. Since $F(\cdot, \cdot, \cdot, t)$ can be written as a function of fundamental differential invariants, in the simplest case we choose F as a linear combination of these transformed fundamental differential invariants, formulated as follows:

$$F(u, x, y, t) = \sum_{i=0}^{5} a_i(x, y, t) g\left(\text{inv}_i\left(u\left(t\right)\right)\right), \tag{3}$$

where $\{a_i(x, y, t)\}_{i=0}^5$ are parameters to be determined.

The nonlinear mapping has another advantage, i.e., making the fundamental differential invariants bounded, reducing the difficulty of optimization and improving numerical stability of the PDE. The experiments show that the mapping can improve face recognition rate by about 4%.

When $F(u, x, y, t)$ in Eq. (1) is chosen as Eq. (3), our PDE is actually a simplified version of the PDE system proposed by Liu et al. [24], who have successfully used this model to handle different image processing problems. Our model adds a nonlinear mapping on each fundamental differential invariants, and drops the indicator function in their model which was introduced for collecting global information. This is because we are considering local features. Omitting the indication function greatly reduces the computational complexity and the training cost. Our PDE also has the following properties:

[4] Like most PDE based methods, we limit our attention to second order PDEs, since higher order PDEs will pose difficulties in numerical stability and theoretical analysis.

Proposition 1. *Suppose the PDE (1) is translationally invariant, then* $\{a_j(x,y,t)\}_{j=0}^5$ *must be independent of* (x,y).

Proposition 2. *When* $F(u,x,y,t)$ *is a function of the fundamental differential invariants,* $u(t)$ *is invariant under the group of translation and rotation through the evolution of the PDE (1).*

The proofs are the same as those in [24], despite the introduction of the nonlinear mapping g. According to Proposition 1, we will use $\{a_j(t)\}_{j=0}^5$ to denote $\{a_j(x,y,t)\}_{j=0}^5$ in the following.

2.3. Classification

When obtaining the feature $u_m|_{t=T}$ from the input image I_m, we need a classifier for classification. In the training phase, we minimize a loss function to determine both the F and the parameters in the classifier. Especially, we first prepare training samples $\{(I_m,h_m)\}_{m=1}^M$, where I_m is the m-th input image, h_m is its corresponding tag vector with 1 at the i-th entry if the m-th input image belongs to class i, and M is the number of samples. For each input image I_m, we obtain a feature map $u_m|_{t=T}$ by PDE (1) for classification. Then the whole learning model can be formulated as finding a certain function $F(u,x,y,t)$ and parameters W of a classifier to minimize a loss function L with a regularization term J:

$$\min_{F,W} E = \frac{1}{M}\sum_{m=1}^{M} L(W; u_m|_{t=T}, h_m) + \lambda J(W), \tag{4}$$

where u_m satisfies the PDE (1) with $u_m|_{t=0} = I_m$ and $\lambda > 0$ is a trade-off parameter. When F is chosen as Eq. (1), we are to determine $a_i(t)$, $i = 0, \cdots, 5$, instead.

For simplicity, we use a linear classifier, such as Multivariate Ridge Regression (MRR), for classification, which is widely used in multi-class classification [12, 14, 10]. We can also adopt the hinge loss as it is advantageous in many cases, such as in face recognition and in dimensionality reduction [30, 31, 32]. The objective in (4) to learn MRR is as follows:

$$E = \frac{1}{M}\|H - W\cdot U|_{t=T}\|_F^2 + \lambda\|W\|_F^2, \tag{5}$$

where $H = [h_1, h_2, \cdots, h_M]$. And as mentioned before, $u_m|_{t=T}$ will be a matrix of the same size as the input image I_m when the PDE is discretized. So for MRR, W will be a matrix with size of $c \times p$, where c is the number of categories and p is the pixel number of the input images I_m[5]. We set $U|_{t=T} = [\mathrm{vec}(u_1|_{t=T}), \mathrm{vec}(u_2|_{t=T}), \cdots, \mathrm{vec}(u_M|_{t=T})]$. When testing, the class label l^* of a testing image I can be obtained as follows:

$$l^* = \arg\max_l \quad \{s_l\}, \tag{6}$$

[5]We assume that all images are in a same size. Otherwise, we will normalize them to a unique size.

Algorithm 1 Training PDEs

Input Training image pairs $\{(I_m, h_m)\}_{m=1}^{M}$, η, λ.

Initialize $\Delta t = 0.5, N = 5, \rho = 0.95, \varepsilon = 10^{-6}, k = 1, k_{\max} = 10$.

Initialize A with each entry uniformly sampled from $[-1, 1]$.

while $k \leq k_{\max}$ **and** $||E^k - E^{k-1}|| > \varepsilon$ **do**

1. For all images, set $u_m^0 = I_m$ and calculate u_m^n by Eq. (10).
2. Solve W by Eq.(11).
3. Update A by one gradient descent step as Eq. (22).
4. Update $\eta = \rho\eta$.
5. Update $k = k + 1$.

end while

where $s = W \cdot \mathrm{vec}(u|_{t=T})$ is the label vector and u satisfies our learned PDE (1) with $u|_{t=0} = I$.

2.4. The Whole PDE Based Feature Learning Model

Integrating feature extraction and classification, our whole PDE model can be formulated as follows:

$$\min_{W,\{a_i(t)\}} E = \frac{1}{M} \|H - W \cdot U|_{t=T}\|_F^2 + \lambda \|W\|_F^2, \tag{7}$$
$$s.t. \begin{cases} \frac{\partial u_m}{\partial t} = \sum_{i=0}^{5} a_i(t) g\left(\mathrm{inv}_i\left(u_m\left(t\right)\right)\right), & (x, y, t) \in Q, \\ u_m(x, y, t) = 0, & (x, y, t) \in \Gamma, \\ u_m|_{t=0}(x, y, t) = I_m, & (x, y) \in \Omega, \end{cases}$$

where $m = 1, 2, \cdots, M$, I_m presents each training image, $H, U|_{t=T}$, W, and λ are the same as those in Eq. (5), and $a_i(t)$ is give in Eq. (3). One can find that our PDE extracts discriminative feature as $\{a(t)_j\}_{j=0}^{5}$ is determined to minimize the loss function of the training data.

3. Algorithm for Solving (7)

In this section, we propose an algorithm to solve our feature learning model (7). The main strategy is to update the parameters A and W alternately, where discretized of a_i is the i-th column of A. We first discretize the PDE and then show details of optimizing A and W. When updating A, we use the gradient descent method. W is given a closed-form solution. The whole algorithm is shown in Algorithm 1, including some fixed hyper-parameters.

3.1. Discretization

We first discretize our PDE. We use central difference to approximate the spatial derivatives as follows:

$$\begin{cases} \frac{\partial f}{\partial x} &= \frac{f(x+1)-f(x-1)}{2}, \\ \frac{\partial^2 f}{\partial x^2} &= f(x+1)-2f(x)+f(x-1), \end{cases} \tag{8}$$

The discrete forms of $\frac{\partial f}{\partial y}, \frac{\partial^2 f}{\partial y}$, and $\frac{\partial^2 f}{\partial x \partial y}$ can be defined similarly through central difference. Then $\mathrm{inv}_i(u), i = 0, \cdots, 5$, can be calculated directly through the discrete form of spatial derivatives, e.g. $\mathrm{inv}_3(u)(p,q) = u(p-1,q)+u(p+1,q)+u(p,q+1)+u(p,q-1)-4u(p,q)$, where (p,q) is the coordinate in the image u.

The temporal derivatives is approximated by forward difference, formulated as:

$$\frac{\partial f}{\partial t} = \frac{f(t+\Delta t)-f(t)}{\Delta t}, \tag{9}$$

where Δt is the step size. We then denote discreted temporal variable t as $t_i = i \cdot \Delta t$, $i = 0, \cdots, N$, where in our experiments $N = 5$. In the sequel, we simply use u_m^n to denote $u_m(x, y, t_n)$ and a_i^n to denote $a_i(t_n)$. So A can be written as a matrix with a_i^n being the (n, i)-th entry. The forward scheme to approximate the evolutionary PDE in Eq. (7) can be written as follows:

$$u_m^{n+1} = u_m^n + \Delta t \sum_{i=0}^{5} a_i^n \cdot g\left(\mathrm{inv}_i\left(u_m^n\right)\right), \tag{10}$$

where $n = 0, 1, \cdots, N-1$.

3.2. Updating W

By fixing A, we calculate $u_m|_{t=T} = u_m^N$ by iterating Eq. (10) with n ranging from 1 to $N-1$. Then W can be solved as:

$$\begin{aligned} W &= \arg\min_W \frac{1}{M} \left\| H - W \cdot U^N \right\|^2 + \lambda \|W\|_F^2 \\ &= H \cdot \left(U^N\right)^T \cdot \left[U^N \cdot \left(U^N\right)^T + \lambda M \mathcal{I}\right]^{-1}, \end{aligned} \tag{11}$$

where $\mathcal{I} \in \mathcal{R}^{p \times p}$ is an identity matrix, p is the pixel number of an image, and $U^N = [\mathrm{vec}(u_1^N), \mathrm{vec}(u_2^N), \cdots, \mathrm{vec}(u_M^N)]$.

3.3. Updating A

When W is fixed, A is updated by the gradient descent method. So we deduce the gradient first. $\frac{\partial E}{\partial a_i^n}$ is obtained by the chain rule or back-propagation [33]:

$$\frac{\partial E}{\partial a_i^n} = \frac{\partial E}{\partial U^{n+1}} \cdot \frac{\partial U^{n+1}}{\partial a_i^n}. \tag{12}$$

where $U^n = [\text{vec}(u_1^n), \text{vec}(u_2^n), \cdots, \text{vec}(u_M^n)]$. According to Eq. (10), $\frac{\partial E}{\partial a_i^n}$ can be rewritten as

$$\frac{\partial E}{\partial a_i^n} = \Delta t \sum_{m=1}^{M} \left\langle \frac{\partial E}{\partial u_m^{n+1}}, g(\text{inv}_i\left(u_m^n\right)) \right\rangle, \tag{13}$$

where $\frac{\partial E}{\partial u_m^n}$ is a matrix with $\frac{\partial E}{\partial u_m^n}(p,q) = \frac{\partial E}{\partial u_m^n(p,q)}$ and $\langle \cdot, \cdot \rangle$ is the matrix inner product. Now we compute $\frac{\partial E}{\partial u_m^n}$. When $n = N$,

$$\frac{\partial E}{\partial U^N} = \frac{1}{M} W^T \cdot \left(W \cdot \text{vec}\left(U^N\right) - H\right). \tag{14}$$

For $n < N$, by the chain rule we have

$$\frac{\partial E}{\partial u_m^n}(p,q) = \frac{\partial E}{\partial u_m^{n+1}}(p,q) + \Delta t \sum_{i=0}^{5} a_i^n \sum_r \sum_s \frac{\partial E}{\partial u_m^{n+1}(r,s)} \frac{\partial g\left(\text{inv}_i\left(u_m^n\right)(r,s)\right)}{\partial u_m^n(p,q)}, \tag{15}$$

where (r,s) is the image coordinate and travels all the pixels over the image. Since the central difference are only linked to the adjacent points on each point, Eq. (15) reduces to:

$$\frac{\partial E}{\partial u_m^n} = \frac{\partial E}{\partial u_m^{n+1}} + \Delta t \sum_{i=0}^{5} a_i^n Z(i,m,n), \tag{16}$$

where $Z(i,m,n)$ is a matrix in a same size of the input image I_m with each element (p,q) being

$$Z(i,m,n)(p,q) = \sum_{r=-1}^{1} \sum_{s=-1}^{1} \frac{\partial E}{\partial u_m^{n+1}(p+r,q+s)} \frac{\partial g\left(\text{inv}_i\left(u_m^n\right)(p+r,q+s)\right)}{\partial u_m^n(p,q)}. \tag{17}$$

In the following, we give details of computing $Z(i,m,n)$. We use $(i=3)$ as an example. The discrete form of $\text{inv}_3(u_m^n)(p,q) = u_m^n(p-1,q) + u_m^n(p+1,q) + u_m^n(p,q+1) + u_m^n(p,q-1) - 4u_m^n(p,q)$. Then we have

$$\begin{aligned}
\frac{\partial g\left(\text{inv}_3\left(u_m^n\right)(p,q)\right)}{\partial u_m^n(p-1,q)} &= g'\left(\text{inv}_3\left(u_m^n(p,q)\right)\right), \\
\frac{\partial g\left(\text{inv}_3\left(u_m^n\right)(p,q)\right)}{\partial u_m^n(p+1,q)} &= g'\left(\text{inv}_3\left(u_m^n(p,q)\right)\right), \\
\frac{\partial g\left(\text{inv}_3\left(u_m^n\right)(p,q)\right)}{\partial u_m^n(p,q-1)} &= g'\left(\text{inv}_3\left(u_m^n(p,q)\right)\right), \\
\frac{\partial g\left(\text{inv}_3\left(u_m^n\right)(p,q)\right)}{\partial u_m^n(p,q+1)} &= g'\left(\text{inv}_3\left(u_m^n(p,q)\right)\right), \\
\frac{\partial g\left(\text{inv}_3\left(u_m^n\right)(p,q)\right)}{\partial u_m^n(p,q)} &= -4g'\left(\text{inv}_3\left(u_m^n(p,q)\right)\right),
\end{aligned} \tag{18}$$

Table 3: Operator $K(g(\text{inv}_i(u_m^n)))$, where $u_A^n = u_x^n u_{xx}^n + u_y^n u_{xy}^n$, $u_B^n = u_y^n u_{yy}^n + u_x^n u_{xy}^n$, and $g' = \frac{1}{(1+|x|)^2}\Big|_{x=\text{inv}_i(u_m^n)}$.

$i=0$	$i=3$
$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & g' & 0 \\ g' & -4g' & g' \\ 0 & g' & 0 \end{pmatrix}$
$i=1$	$i=4$
$\begin{pmatrix} 0 & 0 & 0 \\ 0 & g' & 0 \\ 0 & 0 & 0 \end{pmatrix}$ $\begin{pmatrix} g'u_x^n u_y^n/2 & g'u_B^n + g'(u_y^n)^2 & -g'u_x^n u_y^n/2 \\ g'u_A^n + g'(u_x^n)^2 & -2g'(u_x^n)^2 - 2g'(u_y^n)^2 & -g'u_A^n + g'(u_x^n)^2 \\ -g'u_x^n u_y^n/2 & -g'u_B^n + g'(u_y^n)^2 & g'u_x^n u_y^n/2 \end{pmatrix}$	
$i=2$	$i=5$
$\begin{pmatrix} 0 & g'u_y^n & 0 \\ g'u_x^n & 0 & -g'u_x^n \\ 0 & -g'u_y^n & 0 \end{pmatrix}$ $\begin{pmatrix} g'u_{xy}^n & 2g'u_{yy}^n & -g'u_{xy}^n \\ 2g'u_{xx}^n & g'\cdot(-4u_{xx}^n - 4u_{yy}^n) & 2g'u_{xx}^n \\ -g'u_{xy}^n & 2g'u_{yy}^n & g'u_{xy}^n \end{pmatrix}$	

where $g'(x) = \frac{1}{(1+|x|)^2}$. So we obtain

$$\begin{aligned} Z(3,m,n)(p,q) = & \frac{\partial E}{\partial u_m^{n+1}(p+1,q)} g'\left(\text{inv}_3\left(u_m^n\right)(p+1,q)\right) + \frac{\partial E}{\partial u_m^{n+1}(p-1,q)} g'\left(\text{inv}_3\left(u_m^n\right)(p-1,q)\right) \\ & + \frac{\partial E}{\partial u_m^{n+1}(p,q+1)} g'\left(\text{inv}_3\left(u_m^n\right)(p,q+1)\right) + \frac{\partial E}{\partial u_m^{n+1}(p,q-1)} g'\left(\text{inv}_3\left(u_m^n\right)(p,q-1)\right) \\ & - 4\frac{\partial E}{\partial u_m^{n+1}(p,q)} g'\left(\text{inv}_3\left(u_m^n\right)(p,q)\right). \end{aligned} \tag{19}$$

To make the above expression simple, we define an operator:

$$(C \circ D)(p,q) = \sum_{r=-1}^{1} \sum_{s=-1}^{1} C(p+r, q+s)[D(r+2, s+2, p+r, q+s)],$$

where C is a matrix with the same size of the image and D is a 3×3 operator, with each entry being a function. D actually has 4 parameters. The first two parameters index an entry in the 3×3 matrix and the last two index the coordinate in an image. Then Eq. 19 can be written as

$$Z(3,m,n) = \frac{\partial E}{\partial u_m^{n+1}} \circ K\left(g\left(\text{inv}_3\left(u_m^n\right)\right)\right), \tag{20}$$

where $K(g(\text{inv}_3(u_m^n)))$ is a 3×3 operator and is $\begin{pmatrix} 0 & g' & 0 \\ g' & -4g' & g' \\ 0 & g' & 0 \end{pmatrix}$. For example, when $i = 3, r = 0$, and $s = -1$, $K(g(\text{inv}_3(u_m^n))(r+2, s+2, p+r, q+s)) = g'(\text{inv}_3(u_m^n(p, q-1))$. For other i, similarly we also have

$$Z(i,m,n) = \frac{\partial E}{\partial u_m^{n+1}} \circ K\left(g\left(\text{inv}_i\left(u_m^n\right)\right)\right), \tag{21}$$

where $K(g(\text{inv}_i(u_m^n)))$ is shown in Table 3.

With the gradient of E computed, by gradient descent, in the k-th iteration A is updated as follows:

$$(a_i^n)^{k+1} = (a_i^n)^k - \eta \frac{\partial E^k}{\partial (a_i^n)^k}, \tag{22}$$

where η is the step size and $\frac{\partial E}{\partial a_i^n}$ is obtained through Eq. (13).

Figure 2: The architecture of L-PDE.

3.4. Complexity

Since each point on $\text{inv}_i(u_m^n)$ is linked only to nine adjacent points in u_m^n, the back-propagation process can be calculated in linear time with respect to the pixel number. The whole complexity of our algorithm is $O(Nmp + p^3)$, where N is 5, m is the number of training samples, and p is the pixel number of the input image. The experiments on Section 5 show that our method is much faster than sparse coding and dictionary learning methods.

4. Disscussion

4.1. Distinction with other PDE based methods

There are also some PDE based works which try to devise particular PDEs for classification [34, 35]. In [34], Yin et al. apply the total variation as regularization to decompose the image, and use the decomposed part as feature for classification. In [35], Shan et al. devise a simple PDE to normalize illumination and then use the normalized image as feature for classification. These PDE based works are actually using the PDE *as a pre-processing for classification*. The classification and PDE are still separated. So these methods should belong to image processing. In contrast, our method integrates classification with feature extraction, which uses a PDE as a learning tool to extract discriminative feature.

4.2. Relation with CNN

The CNN models have achieved a huge success in image classification tasks [21] in recent layers. Like CNN, the discrete form of our PDE has a hierarchical architecture. Since the element-wise operations can be available to all modern deep learning training suits, e.g. Caffe [36], Torch [37], our PDE can be implemented as a "special CNN". The architecture of our PDE model is shown in Fig. 2 and Fig. 3, where Fig. 3 illustrates the internal architecture in the processing cell in Fig. 2. A traditional CNN with a similar architecture is shown in Fig. 2 and Fig. 4.

There are still critical distinctions between our L-PDE and CNN. First, from Fig 3, the fundamental differential invariants are non-linear and are calculated through element-wise multiplication, not convolutional operators. Second, for neural networks, the non-linear mapping is after the linear

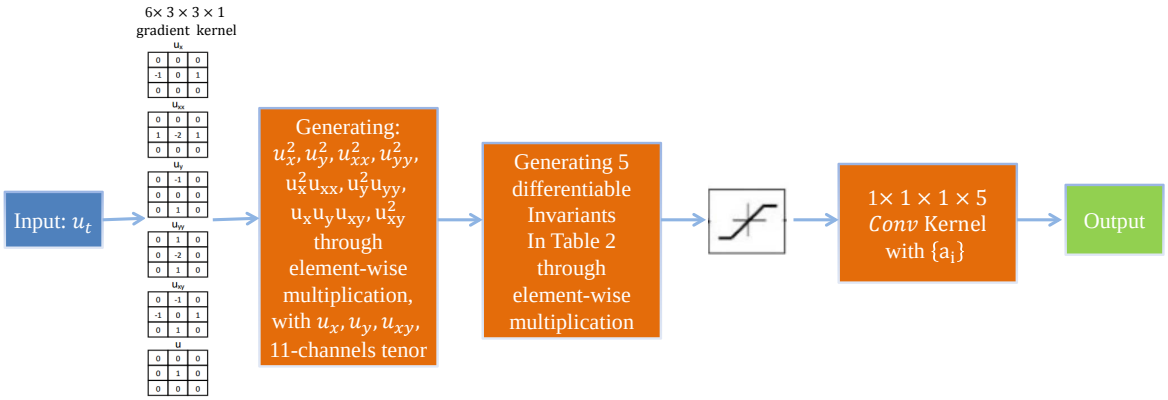

Figure 3: The internal architecture in processing cell of L-PDE.

Figure 4: The internal architecture in processing cell of CNN.

transformation, while our PDE does the non-linear mapping before the linear transformation. $\{a_i\}$ are used to linearly combine the fundamental differential invariants. Third, in practice, most CNN models have a large number of parameters in the convolutional kernels, and they learn the feature through the strength of "big data". Our PDE model develops invariance properties in the evolution process and works with few training samples.

5. Experiments

In this section, we present experiments to validate the proposed method. Classification with few training samples is a big challenge in image classification tasks, which is often encountered in reality and could be as difficult as the case of large training samples. Many sparse coding and dictionary learning methods [14, 10, 7, 8, 12] have aimed at classification in this case and have shown their superiorities. Face recognition is a paradigm which has few training samples but a lot of real applications, such as biometrics, information security, access control, law enforcement, smart cards and surveillance system (see [38] for a review). We focus our experiments on face recognition and *do the same or similar experiments* to compare with those sparse coding and dictionary learning methods. Currently, like all the compared methods, we focus on low-resolution images. We choose four datasets: Extended Yale B [39], PIE [40], AR [41], and FRGC [42], shown in Figure 5 sequentially. The first three datasets have also been used by compared methods [14, 10, 7, 8]. We *use the same or similar training*

samples and image scales on these datasets to compare with them. The three datasets have different difficulties. The faces in Extended Yale B are under different illuminations which are hard to be linearly represented. The PIE dataset is taken under different poses. The main challenge of AR is that it contains different facial expressions and occlusions (sunglasses and scarf). We use the FRGC dataset to test our method when the training samples are few (only five images for each person).

In the above recognition tasks, we compare our method with the existing state-of-the-art sparse coding and dictionary learning feature learning methods: D-KSVD [12], LC-KSVD [14], Task-Driven Dictionary Learning (TDDL) [13], and Low-Rank Representations Classification (LRRC) [10]. All these methods use Ridge Regression for classification. So the differences in recognition performance reflect the effectiveness of feature learning. We do not compare our model with the old PDE based methods [34, 35] which use the PDE as a pre-processing for classification, since we find that their results are inferior to those sparse coding methods, such as SRC [8]. We also compare our method with representative face recognition methods: k-Nearest Neighbors [43], Kernel Support Vector Machine [30], SRC [8], and Low-Rank Structural Incoherence Classification (LRC and LRSIC) [7], since all the experiments are conducted on the face datasets[6]. LRC and LRSIC [7] are regarded as face recognition methods because they use SRC [8] for recognition. We also compare our method with CNN in all experiments. The architecture is shown in Fig. 2 and Fig. 4. The depth and the parameter numbers of the CNN are almost the same as L-PDE's. We train the CNN through two ways. The first way is through the standard Gradient Descent. We set the step size in Gradient Descent using the $t-$inverse schedule $\eta_t = \eta_0(1+\eta' t)^{-1}$, where η_0 and η' are chosen to give the best recognition results. We find CNN is faced with serious overfitting when training by Gradient Descent. So we also compare the CNN which use the same optimization tools as L-PDE's. We alternately update the parameters in the kernels and the linear classifier. It is the same optimization method as L-PDE's. We use CNN-GD to denote the recognition accuracies of CNN trained by Gradient Descent, and use CNN-AD to denote the recognition accuracies of CNN trained by Alternately Descent. Throughout the experiments, our method and CNN work on the raw data, while we normalize the Frobenius norm of each image to 1 when testing other methods. We choose a Gaussian kernel in SVM (K-SVM). For dictionary learning methods, including LC-KSVD [14], TDDL [13], and LRRC [10], we choose the number of atoms to be 5 for each class. For each algorithm, parameters are tuned to the best. And for each experiment, we repeat 10 times and report the average accuracy. The platform is Matlab 2013a under Windows 7 on

[6]The codes for D-KSVD and LC-KSVD are downloaded from the authors' websites. SVM is from libSVM [44]. kNN is a function in Matlab. Other methods are our own implementations.

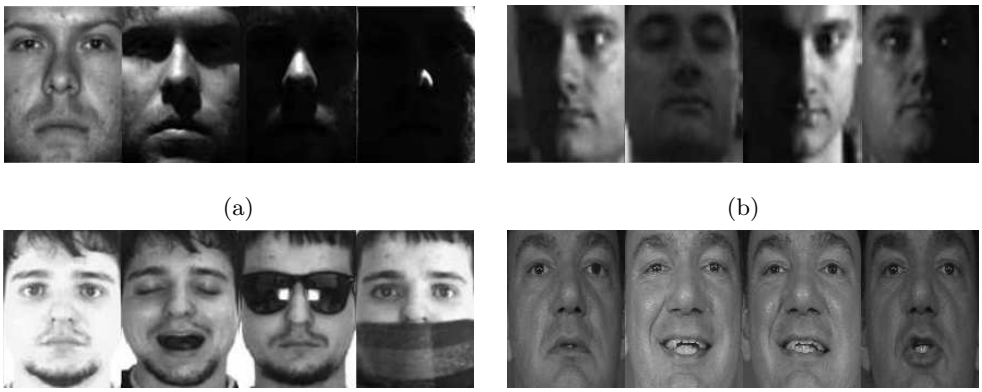

Figure 5: Sample images from (a) YaleB, (b) PIE, (c) AR, and (d) FRGC, respectively.

a PC equipped with a 3.4GHz CPU and 8GB memory.

5.1. Extended Yale B Dataset

Table 4: Recognition accuracies (%) on Extended Yale B, with 10, 15, and 20 training samples.

Type	Method	# training samples		
		10	15	20
Feature Learning + Ridge Regression	**L-PDE (ours)**	**96.3**	**98.1**	**98.8**
	CNN-GD	17.3	21.2	24.2
	CNN-AD	83.7	86.2	88.5
	LC-KSVD1	88.0	91.2	93.2
	LC-KSVD2	89.2	92.4	94.2
	TDDL	84.7	89.5	93.8
	LRRC	84.8	91.6	93.6
Others	kNN	54.8	63.8	69.8
	K-SVM	87.8	93.1	95.1
	SRC	87.9	93.6	96.4
	LRC	87.7	92.3	94.6
	LRSIC	88.2	94.0	95.1

We first test our method on the Extended Yale B dataset [39]. There are $2,414$ frontal-face images of 38 people with a cropped and normalized size of 192×168. The faces are captured under various

laboratory-controlled lighting conditions [45]. Following [10, 7], for each person we randomly select 10, 15, and 20 images for training and the others for testing. As the dimension of the images is high, we down sample each image by 1/4.

We choose $\lambda = 1.5$ and $\eta = 0.5$ in our method. The experimental results are summarized in Table 4. Our approach outperforms all the methods in all cases and the advantages are more when the train samples are fewer. CNN-GD achieves poor recognition results due to serious overfitting. Our method achieves higher recognition accuracies than CNN-AD since our method maintains invariant properties through the evolution process. We also find KSVD methods, including LC-KSVD [14], achieve inferior results than SRC [8]. The same phenomenon is also observed in [12].

Figure 1 shows the evolution process of our learned PDE on three persons. One can see that the lighter faces gradually become darker and the darker faces change to lighter during the evolution of PDE. So the features U_i^N become invariant under different illuminations. This demonstrates that our methods are robust to illumination variation. This phenomenon may be due to two reasons. First, we add a nonlinear mapping $g(x) = \frac{x}{1+|x|}$ on each fundamental differential invariant which is nearly constant when $|x|$ is large. So the fundamental differential invariants are nearly invariant under gray-level scaling. Second, our PDE is learned to obtain good recognition results. The training dataset provides training samples that are under different illuminations. So feature is learned to be invariant under these variants.

5.2. PIE Dataset

The PIE dataset [40] consists of $41,368$ images of 68 individuals. Each individual has 4 different expressions, 13 different poses and 43 different illumination conditions. Like [46], a subset (C05, C07, C09, C27, C29) of PIE contains 5 near frontal poses and all the images under different illuminations and expressions are chosen for experiment. Thus, each subject has about 170 images. Like [46], we also randomly select 10, 15, and 20 images for training and the others for testing. Each image is down sampled to 32×32.

We choose $\lambda = 1.5$ and $\eta = 0.5$ in our method. The experimental results are summarized in Table 5. Our method also obtains the best recognition rates at different numbers of training samples. Since the dataset is relatively hard, some feature learning methods perform poorly. The experiment demonstrates the robustness of our method to different poses.

5.3. AR Dataset

The AR dataset [41] consists of over $4,000$ frontal images of 126 people. For each individual, images are separated into 2 sessions with different difficulties, including illumination, expression, and facial

Table 5: Recognition accuracies (%) on PIE, with 10, 15, and 20 training samples.

Type	Method	# training samples		
		10	15	20
Feature Learning + Ridge Regression	**L-PDE (ours)**	**84.1**	**88.9**	**90.9**
	CNN-GD	18.7	19.6	21.5
	CNN-AD	71.9	80.2	83.1
	LC-KSVD1	35.8	36.8	65.0
	LC-KSVD2	36.2	37.7	65.3
	TDDL	78.4	84.4	87.9
	LRRC	79.8	85.2	89.1
Others	kNN	29.0	29.3	31.1
	K-SVM	73.4	82.9	85.7
	SRC	77.3	87.2	90.5
	LRC	79.1	84.7	88.3
	LRSIC	82.4	87.7	90.6

occlusion/disguise. All images are at the size of 165×120. For each session, there are 3 images obscured by sunglasses, 3 images obscured by scarves, and 7 clean images with expressions and illuminations variations. Following [10, 7, 8], in our experiments we select a subset of the AR dataset consisting of 50 men and 50 women and down sample each image by 1/5. Following [10, 7], the experiments are under the following scenarios:

- **Sunglasses**: We consider the case where images are only occluded by sunglasses. We use 7 clean images and 1 image with sunglasses (randomly chosen) from session 1 for training. The testing images consist of 4 sunglasses images (2 from session 1 and 3 from session 2) and 7 remaining clean images (all from session 2).

- **Mixed**: We consider the case where images are both occluded by sunglasses and scarf. We select all 7 clean images from session 1 and 2 corrupted images (occluded by sunglasses and the scarf, respectively) for training. The rest of 19 images are for testing.

- **Hybrid**: In this case, we choose images from session 1 for training and session 2 for testing. The numbers of training and testing images are all 13 for each person.

Table 6: Recognition accuracies (%) on AR (S.G. is short for Sunglasses).

Type	Method	Scenario		
		S.G.	Mixed	Hybrid
	L-PDE (ours)	**88.9**	**87.1**	**87.2**
Feature	CNN-GD	31.5	29.8	31.4
Learning	CNN-AD	82.5	83.6	82.7
+ Ridge	D-KSVD	76.6	69.5	71.4
Regression	LC-KSVD1	78.0	79.5	79.7
	LC-KSVD2	79.2	80.8	81.3
	TDDL	83.6	82.7	83.5
	LRRC	86.1	82.7	83.4
	kNN	66.9	61.6	61.1
	K-SVM	81.6	79.9	81.2
Others	SRC	88.6	83.9	85.0
	LRC	84.7	81.3	82.6
	LRSIC	87.2	83.5	84.0

We choose $\lambda = 45$ and $\eta = 0.15$ in our method. The experimental results are summarized in Table 6. Our approach obtains the best results in all three scenarios. This shows that the occlusion problem can be relieved by learning discriminative local feature.

5.4. FRGC Dataset

We also conduct our experiment on Experiment 4 in the FRGC 2.0 dataset [42]. Experiment 4 is the most challenge FRGC experiment. In the query set, the dataset consists of $8,014$ single uncontrolled still images of 466 individuals. Like [47, 48], we search all images of each person in this set and take the first 60 images of the first 60 individuals, whose number of facial images is more than 60. Thus, we collect 3,600 facial images for our experiments. We down sample the images to a size of 32×36. For each person, we only randomly choose 5 images for training. The rest 55 images are for testing.

We choose $\lambda = 1.6$ and $\eta = 0.1$ in our method. The experimental results are summarized in Table 7. Our method also gets the best results in the case of few samples.

Table 7: Recognition accuracies (%) on FRGC (Acc. stands for the recognition accuracy).

Type	Method	Acc.	Type	Method	Acc.
Feature Learning + Ridge Regression	**L-PDE (ours)**	**92.3**	Others.	kNN	54.4
	CNN-GD	15.1		K-SVM	85.4
	CNN-AD	83.8		SRC	87.8
	D-KSVD	60.2		LRC	85.6
	LC-KSVD1	63.4		LRSIC	87.6
	LC-KSVD2	88.7			
	TDDL	91.3	F.+R.	LRRC	87.6

Table 8: Average training time (s), normalized by the training samples, on the four database.

Method \ Dataset	Yale B	PIE	AR	FRGC
D-KSVD	1.0368	1.3521	0.9949	0.8757
LC-KSVD1	0.2634	0.4008	0.2445	0.2036
LC-KSVD2	0.2758	0.4079	0.2593	0.2191
L-PDE (ours)	**0.0519**	**0.0549**	**0.0424**	**0.0593**

5.5. Comparison of Computation Time and Hyper-parameter Selection

We compare the average training and testing time of our method with those dictionary learning and sparse coding methods. The average training or testing time is the total training or testing time divided by the number of training or testing samples. Since SRC [8] have no training time, and LRC [7] and LRSIC [7] only use the low rank ingredient as a dictionary, their training times can be ignored. So we only compare the average training time with dictionary learning methods. Tables 8 and 9 show the average training time and testing time for each image, respectively. We can see that our model is fast in both training and testing processes. As a result, the whole training and testing time on each database are no more than 5 minutes. This is due to the low complexity ($O(5mp + p^3)$ for one iteration) of our method. The results show the practicability and efficiency of our PDE method.

Our method has two hyper-parameters, λ and η, to tune. One may notice that we use different parameters in the different datasets. The settings of hyper-parameters that we use in the experiments are tuned to obtain the best recognition performances of our method. We have also tuned the parameters to be best for the compared methods. So the experiments are fair. Our method has two hyper-parameters, λ and η, to tune. Now we give suggestions on how to set the hyper-parameters.

Table 9: Average testing time (s), normalized by the testing samples, on the four database.

Method \ Dataset	Yale B	PIE	AR	FRGC
D-KSVD	0.0151	0.0298	0.0135	0.0088
LC-KSVD1	0.0144	0.0287	0.0123	0.0063
LC-KSVD2	0.0141	0.0264	0.0121	0.0059
SRC	0.3936	0.3499	0.1245	0.0133
LRC	0.3527	0.5479	0.2482	0.0183
LRSIC	0.3617	0.6658	0.2799	0.0182
L-PDE (ours)	**0.0027**	**0.0010**	**0.0014**	**0.0011**

Table 10: Recognition accuracies using the suggested parameters

Extended Yale B	10	15	20
L-PDE	96.1	97.8	98.7
PIE	10	15	20
L-PDE	83.3	88.0	90.0
AR	S.G.	Mixed	Hybrid
L-PDE	88.5	86.6	86.5

λ is a regularization parameter in the linear classifier. Since the training samples are limited, λ is critical in the performance. We suggest λ chosen from $\{1, 5, 10, 50, 100\}$. η is the step size during optimization. We suggest setting η from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. There are 25 selections to choose the pairs of hyper-parameters. Fig. 6 shows the effects of hyper-parameters on recognition accuracy on the Extended Yale B dataset. Table 10 reports the recognition accuracies on Extended Yale B , PIE, and AR using the suggested parameters. We can see that the recognition accuracies drop less than 1% in all experiments.

5.6. Comparison with Pre-trained Neural Networks on Low-Resolution Images

Deep neural networks trained on large dataset are observed to have a great generalization ability to extract discriminative feature for images. It is shown in [49] that the classification accuracy obtained by pre-trained CNN surpasses around 20% than the traditional method which uses SIFT [2] to extract the feature on the Caltech 101 dataset. In this subsection, we compare our method with the pre-trained VGG-Face [50] model to demonstrate the effectiveness of our method on low-resolution images.

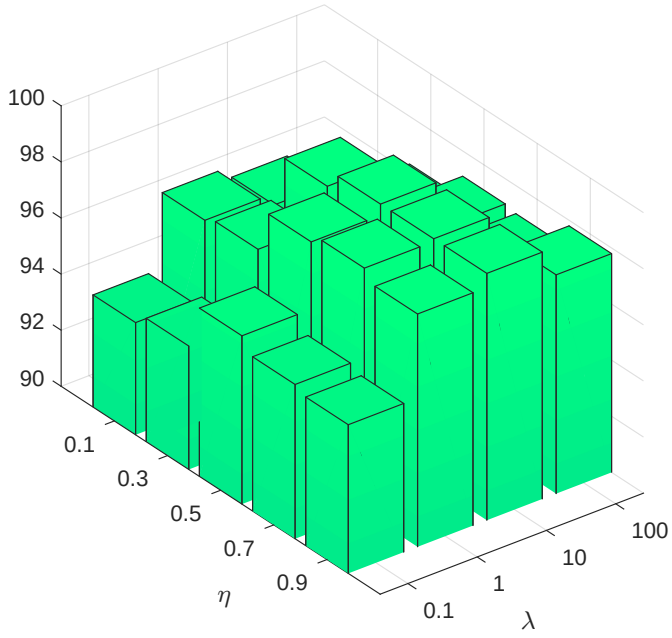

Figure 6: The effects of hyper-parameters on recognition accuracy on the Extend Yale B dataset.

Table 11: Comparison with pre-trained VGG-Face on Extended Yale B, PIE, AR and FRGC.

Method \ Dataset	Extended Yale B			PIE			AR			FRGC
	# training samples			# training samples			Scenario			
	10	15	20	10	15	20	S.G.	Mixed	Hybrid	
L-PDE (ours)	**96.3**	**98.1**	**98.8**	**84.1**	**88.9**	**90.9**	**88.9**	**87.1**	**87.2**	**92.3**
VGG-Face	81.5	84.5	85.0	83.3	86.2	88.8	77.2	79.1	83.5	91.2

VGG-Face is originally trained on the dataset with $2.6M$ images, and has achieved the state-of-art performance for face recognition. In this experiment, the network is used out-of-box in order to produce a discriminant feature of facial images. The facial images are first normalized to the sizes that are used in all other methods (48×42 in Extended Yale B, 32×32 in PIE, 33×24 in AR, and 32×32 in FRGC) and then back to 224×224 in order to match the input size of pre-trained VGG-Face net. The normalized images then go through the pre-trained VGG-Face [7] model. We apply Ridge Regression on the outputs of the last feature layer for classification. We conduct experiments on the four datasets. The recognition results are shown in Table 11. Our PDE model achieves higher recognition accuracies on low-resolution images. The advantages are more on Extended Yale B and AR.

[7] The model is downloaded from the website: http://www.robots.ox.ac.uk/~vgg/software/vgg_face/.

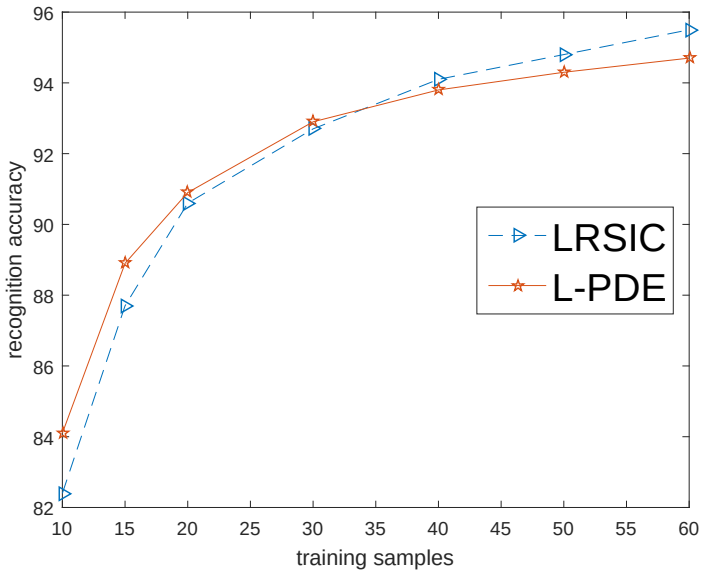

Figure 7: The recognition accuracy against the number of training samples on PIE. The horizontal axis represents the number of training samples for each person. LRSIC achieves the best results among the compared methods when the training samples are 10, 15, and 20. So we only compare with it.

5.7. Training with More Samples

The previous experiments have demonstrated the effectiveness of our method when there are few training samples. It is also interesting to explore the case when there are more training samples. We conduct an experiment on the PIE dataset. Fig. 7 shows the recognition accuracy against the number of training samples on the PIE dataset. We choose the PIE dataset, since we have achieved very high recognition accuracies on Extended Yale B, and the rest datasets (AR and FRGC) do not have enough training samples. Due to the time limit, we only compare our method with Low-Rank Structural Incoherence Classification (LRSIC) [7], because LRSIC achieves the best recognition accuracies among all compared methods when the training samples are 10, 15, and 20. Compared with LRSIC, the improvement of our method through the increase of training samples is not remarkable. This may be due to the limited parameters in our PDE. We are going to extend our PDE to a system of PDEs in the future.

6. Conclusions

In this paper, we propose a novel PDE method for feature learning. We model the feature extraction process as an evolution process governed by a PDE. The PDE is assumed to be a linear combination of

fundamental differential invariants under translation and rotation, which is transformed by a nonlinear mapping to achieve the invariance with respect to gray-level scaling. The experiments with few training samples show that our approach achieves the best performance in various settings. It should be mentioned that our approach could be applied to not only face recognition problems but also general image classification problems. In the future, we will extend our PDE to a system of PDEs and carry out some theoretical analysis.

Acknowledgment

Zhouchen Lin is supported by National Basic Research Program of China (973 Program) (grant no. 2015CB352502), National Natural Science Foundation (NSF) of China (grant no. 61231002), and the Okawa Foundation. Zhenyu Zhao is supported by NSF China (grant nos. 61473302).

References

[1] Yoshua Bengio, Aaron Courville, and Pierre Vincent. Representation learning: A review and new perspectives. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[2] David Lowe. Object recognition from local scale-invariant features. In *Proc. IEEE Int'l. Conf. Computer Vision*, volume 2, pages 1150–1157, 1999.

[3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE Conf. Automatic Face and Gesture Recognition*, pages 886–893, 2005.

[4] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, 2013.

[5] Ronen Basri and David Jacobs. Lambertian reflectance and linear subspaces. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(2):218–233, 2003.

[6] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2013.

[7] Chih-Fan Chen, Chia-Po Wei, and Yu-Chiang Frank Wang. Low-rank matrix recovery with structural incoherence for robust face recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 2618–2625, 2012.

[8] John Wright, Allen Yang, Arvind Ganesh, Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009.

[9] Yong Li, Jing Liu, Zechao Li, Yangmuzi Zhang, Hanqing Lu, and Songde Ma. Learning low-rank representations with classwise block-diagonal structure for robust face recognition. In *Proc. AAAI Conf. on Artificial Intelligence*, 2014.

[10] Yangmuzi Zhang, Zhuolin Jiang, and Larry Davis. Learning structured low-rank representations for image classification. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 676 – 683, 2013.

[11] Fei Wu, Xiao Yuan Jing, Xinge You, Dong Yue, Ruimin Hu, and Jing Yu Yang. Multi-view low-rank dictionary learning for image classification. *Pattern Recognition*, 50:143–154, 2015.

[12] Qiang Zhang and Baoxin Li. Discriminative K-SVD for dictionary learning in face recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 2691–2698. IEEE, 2010.

[13] Julien Mairal, Francis Bach, and Jean Ponce. Task-driven dictionary learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34(4):791–804, 2012.

[14] Zhuolin Jiang, Zhe Lin, and Larry Davis. Label consistent K-SVD: Learning a discriminative dictionary for recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(11):2651–2664, 2013.

[15] Pengju Liu, Hongzhi Zhang, Kai Zhang, Changchun Luo, and Wangmeng Zuo. Class relatedness oriented discriminative dictionary learning. *Pattern Recognition*, 546:335–343, 2015.

[16] Weihua Ou, Xinge You, Dacheng Tao, Pengyue Zhang, Yuanyan Tang, and Ziqi Zhu. Robust face recognition via occlusion dictionary learning. *Pattern Recognition*, 47(4):1559–1572, 2014.

[17] Hui Dong Liu, Ming Yang, Yang Gao, Yilong Yin, and Liang Chen. Bilinear discriminative dictionary learning for face recognition. *Pattern Recognition*, 47(5):1835–1845, 2014.

[18] Andrew Wagner, John Wright, Arvind Ganesh, Zihan Zhou, and Yi Ma. Towards a practical face recognition system: Robust registration and illumination by sparse representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34(2):597–604, 2012.

[19] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[20] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-Rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Conf. Neural Information Processing Systems*, pages 1097–1105, 2012.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] Risheng Liu, Zhouchen Lin, Wei Zhang, and Zhixun Su. Learning PDEs for image restoration via optimal control. In *Proc. European Conf. Computer Vision*, pages 115–128. Springer, 2010.

[24] Risheng Liu, Zhouchen Lin, Wei Zhang, Kewei Tang, and Zhixun Su. Toward designing intelligent PDEs for computer vision: An optimal control approach. *Image and Vision Computing*, 31(1):43–56, 2013.

[25] Risheng Liu, Junjie Cao, Zhouchen Lin, and Shiguang Shan. Adaptive partial differential equation learning for visual saliency detection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 3866–3873, 2014.

[26] Zhenyu Zhao, Cong Fang, Zhouchen Lin, and Yi Wu. A robust hybrid method for text detection in natural scenes by learning-based partial differential equations. *Neurocomputing*, 168:23–34, 2015.

[27] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[28] Joachim Weickert. *Anisotropic diffusion in image processing*. Teubner Stuttgart, 1996.

[29] Peter Olver. *Applications of Lie groups to differential equations*. Springer-Verlarg, 1993.

[30] Bernhard Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Proc. of Annual Acm Workshop on Computational Learning Theory*, volume 5, pages 144–152, 1996.

[31] Kilian Weinberger, John Blitzer, and Lawrence Saul. Distance metric learning for large margin nearest neighbor classification. In *Proc. Conf. Advances in Neural Information Processing Systems*, pages 1473–1480, 2005.

[32] Xinggang Wang, Baoyuan Wang, Xiang Bai, Wenyu Liu, and Zhuowen Tu. Max-margin multiple-instance dictionary learning. In *Proc. Int'l. Conf. on Machine Learning*, pages 846–854, 2013.

[33] Rumelhart Williams and Geoffrey Hinton. Learning representations by back-propagating errors. *Nature*, 323(6088):323–533, 1986.

[34] Wotao Yin, Donald Goldfarb, and Stanley Osher. The total variation regularized L^1 model for multiscale decomposition. *Multiscale Modeling and Simulation*, 6(1):190–211, 2007.

[35] Terrence Chen, Wotao Yin, Xiang Sean Zhou, Dorin Comaniciu, and Thomas Huang. Illumination normalization for face recognition and uneven background correction using total variation based image models. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 532–539, 2005.

[36] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proc. 22nd ACM Int'l. Conf. on Multimedia*, pages 675–678. ACM, 2014.

[37] Collobert Ronan. Torch: a modular machine learning software library. Technical report, 2002.

[38] Wenyi Zhao, Rama Chellappa, Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458, 2003.

[39] Athinodoros Georghiades, Peter Belhumeur, and David Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.

[40] Terence Sim, Simon Baker, and Maan Bsat. The CMU pose, illumination, and expression (PIE) database. In *Proc. IEEE Conf. Automatic Face and Gesture Recognition*, pages 46 – 51, 2002.

[41] Aleix Martinez. The AR face database. *CVC Technical Report*, 24, 1998.

[42] Jonathon Phillips, Patrick Flynn, Todd Scruggs, Kevin Bowyer, Jin Chang, Kevin Hoffman, Joe Marques, Jaesik Min, and William Worek. Overview of the face recognition grand challenge. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 947–954. IEEE, 2005.

[43] Keinosuke Fukunaga and Patrenahalli Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE Trans. on Computers*, C-24(7):750–753, 1975.

[44] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *Acm Trans. on Intelligent Systems and Technology*, 2(3):389–396, 2011.

[45] Kuang-Chih Lee, Jeffrey Ho, and David Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(5):684–698, 2005.

[46] Pan Zhou, Zhouchen Lin, and Chao Zhang. Integrated low-rank-based discriminative feature learning for recognition. *IEEE Trans. on Neural Networks and Learning Systems*, 2015.

[47] Wei Zhang, Zhouchen Lin, and Xiaoou Tang. Learning Semi-Riemannian metrics for semisupervised feature extraction. *IEEE Trans. on Knowledge and Data Engineering*, 23(4):600–611, 2011.

[48] Risheng Liu, Zhouchen Lin, Zhixun Su, and Kewei Tang. Feature extraction by learning Lorentzian metric tensor and its extensions. *Pattern Recognition*, 43(10):3298–3306, 2010.

[49] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conf. on Computer Vision*, pages 818–833, 2014.

[50] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference*, volume 1, page 6, 2015.