An Interpretable Compression and Classification System: Theory and Applications

Tzu-Wei Tseng, Kai-Jiun Yang, C.-C. Jay Kuo and Shang-Ho (Lawrence) Tsai†

arXiv:1907.08952v1 [cs.CV] 21 Jul 2019

***Abstract*—This study proposes a low-complexity interpretable classification system. The proposed system contains three main modules including feature extraction, feature reduction, and classification. All of them are linear. Thanks to the linear property, the extracted and reduced features can be inversed to original data, like a linear transform such as Fourier transform, so that one can quantify and visualize the contribution of individual features towards the original data. Also, the reduced features and reversibility naturally endue the proposed system ability of data compression. This system can significantly compress data with a small percent deviation between the compressed and the original data. At the same time, when the compressed data is used for classification, it still achieves high testing accuracy. Furthermore, we observe that the extracted features of the proposed system can be approximated to uncorrelated Gaussian random variables. Hence, classical theory in estimation and detection can be applied for classification. This motivates us to propose using a MAP (maximum a posteriori) based classification method. As a result, the extracted features and the corresponding performance have statistical meaning and mathematically interpretable. Simulation results show that the proposed classification system not only enjoys significant reduced training and testing time but also high testing accuracy compared to the conventional schemes.**

Index Terms—Linear transform, classification, feature extraction, feature reduction, image recognition, data compression, convolution neural network, machine learning.

I. Introduction

Classification for multimedia has been studied and applied to a variety of applications such as security, entertainment, and forensics for years. The development of intelligent algorithms for classification results in efficiency improvements, innovations, and cost savings in several areas. However, classification based on visual content is a challenging task because there is usually a large amount of intra-class variability, caused by different lighting conditions, misalignment, blur, and occlusion. To overcome the difficulty, numerous feature extraction modules have been studied to extract the most significant features and the developed models can achieve higher accuracy. Selecting a suitable classification module to handle the extracted features according to their properties is also important. A good combination of feature extraction and classification modules is the key to attain high accuracy in image classification problems.

In general, classification models can be divided into two categories: 1) Convolutional neural networks (CNN) structure and 2) non-CNN based structure. The CNN structures are in the mainstream [1]-[6], and they are usually stacked up with several convolution layers, max-pooling layers, and ReLU layers. Hence the depth of the structure can be deep such as ResNet [6]. Due to the deep structure, CNN models can extract the features well and achieve high accuracies. However, CNN models usually share several disadvantages, including mathematically intractable, irreversible, and time-consuming. Most of the CNN models use handcrafted-based feature extraction or backpropagation to train the convolution layers which makes the models mathematically inexplicable and time-consuming. In addition, the max pooling and ReLU layers are nonlinear, hence the structures become irreversible. The above disadvantages increase the difficulty in designing and adjusting suitable structures for various types of data sources. Therefore, research has been conducted to design a model which can be easily and fully interpretable mathematically like the second category introduced below.

Classification models using non-CNN based structures are mainly built with two parts, feature extraction modules, and classification modules. To design a model which is mathematically interpretable, several machine learning techniques for feature extraction or classification have been developed including various versions of principal component analysis (PCA) [7]-[17], linear discriminant analysis (LDA) [18]-[25], support vector machine (SVM), nearest neighbor (NN), etc. For the feature extraction modules, in [7]-[17] the authors used PCA; while in [18]-[25], the authors used LDA to extract features from images. For the classification modules, [8], [9], [13], [17] and [21] applied the nearest neighbor classifier. On the other hand, [7] and [20] utilized the SVM classifier. The combinations of feature extraction and classification modules are generally nonlinear. Thus to inverse or recover the extracted features to original data is difficult.

Inversing the features to original data reveals important information for classification. For instance, if certain significant features are extracted and they result in high classification accuracy, one would be interested in visualizing or quantifying these features in the original multimedia if this is feasible. The reversibility is like Fourier series. One can inverse individual Fourier series, quantify how they contribute to the original signals and understand the importance of individual series. Due to the difficulty of data reversibility in most existing solutions, methods to compress data efficiently dedicated for

[1]S.-H. Tsai and T.-W. Tseng are with the department of Electrical Engineering, National Chiao Tung University, Hsinchu, Taiwan. E-mails: shanghot@alumni.usc.edu and ryan8314brad@gmail.com. This research was supported by the Ministry of Science and Technology (MOST), Taiwan under Grant MOST 107-2221-E-009-065.

[2]K.-J. Yang is with the Industrial Technology Research Institute, Zhudong 31057, Taiwan. E-mail: kaijiuny.ece98g@g2.nctu.edu.tw.

[3]C.-C. J. Kuo is with the department of Electrical Engineering, Signal and Image Processing Institute, University of Southern California, Los Angeles, USA. E-mail: cckuo@sipi.usc.edu.

classification purposes have not been well addressed yet.

Data compression for classification purposes lead to not only reduced storage size but also decreased computational complexity. For instance, in a classification problem, if the extracted features can be reduced while the reduced features can achieve satisfactory testing accuracy and the inversed image from the reduced features still keep important characteristics of the original image as well, one may store the reduced features instead of all features. Also, the reduced features can be used for classification to decrease computational complexity. Furthermore, such data compression is interpretable because it keeps the most significant features for classifications.

In this study, we propose an interpretable classification system that can achieve high testing accuracy with low complexity. The proposed system is linear and inversible. Hence data compression via the system is possible. The proposed system can be divided into three parts including feature extraction, feature reduction and the classification for the reduced features. Thanks to the linear property of the proposed system, the extracted features can be inversed to original data to see insight into individual features. Thus, it is like a linear transform, such as Fourier transform, where both forward and backward directions are feasible. The proposed system can also significantly reduce the extracted features while it still maintains high classification accuracy. As a result, data compression for classification purposes is realizable via the proposed system. Moreover, we find that the extracted features in the proposed system can be approximated to uncorrelated Gaussian random variables. It is worth pointing out that the Gaussian approximation result was also observed in [7]. The Gaussian approximation result endues the proposed system and the extracted features statistical meaning. Hence classical estimation and detection theory can be applied to the proposed system for classifying the data [26]. This motivates us to use the concept of maximum a posteriori (MAP) to detect the class of input images. Therefore for every input image, there is a probability that this image belongs to each class. The detected class is the one that has the maximum probability. Since every class has a probability for the input data, one can also determine top candidate classes for the data and develop more sophisticated algorithms to refine the classification results. Consequently the proposal is an interpretable compression and classification system.

To verify the classification ability of the proposed system, we conduct experiments in face recognition who is this person?. We use the dataset in [27], Labeled Faces in the Wild (LFW). LFW is widely used by face recognition models like [1]-[4] and [8]. Experiment results show that the proposed scheme outperforms conventional systems in terms of both testing accuracy and computational complexity. Moreover, the training and testing time of the proposed system is much faster than conventional schemes. In a standard PC platform, two datasets with 2804 and 6592 images only take 11 and 16 seconds for training respectively. Also, less than 1 msec to recognize the class of one image. The accuracy reaches 97.61% for a 19-person dataset and 84.71% for a 158-person dataset. Furthermore, thanks to the linear property, the proposed system can inverse the reduced features to the original image and the compression ratio is significant. In our experiments, the proposed system can reduce the number of features from 12288 to 270, a compression ratio up to 45.51:1; at the same time, the average deviation between the original and compressed images is only 9.14%, and more importantly the testing accuracy is 97.61% for the 19-person dataset and 84.71% for the 158-person dataset.

The outline of the remaining parts is organized as follow: In Section II, we present the linear recognition model and corresponding algorithms. In Sections III-V, individual modules of the proposed system are explained in details, where Section III explains the linear feature extraction module, Section IV describes the linear feature reduction module, and Section V introduces the linear classification module. In Section VI, we provide experimental results to show the advantages of the proposed system. Conclusions and future works are given in Section VII.

II. Proposed Recognition Model and Methods

A block diagram of the proposed recognition model is shown in Fig. 1, which consists of preprocessing, feature extraction, feature reduction, and classification modules. The preprocessing module handles the original defects of the dataset such as noise, contrast, and brightness of the images. The preprocessing module contains operations including object detection, image processing, and data augmentation. It is an adjustable module designed according to the dataset. The feature extraction module extracts the significant features out from the dataset and has multiple stages. Each of the stages is a linear transformation. Hence the transformation is invertible, and the images can be reconstructed from the extracted features. This point will become more clear later. After the feature extraction module, features are reduced for data compression purpose and avoiding overfitting. Finally, the reduced features are passed through the classification module, where we propose to use linear discriminant classifier because we find that a Gaussian statistical model can be assumed after the proposed linear transformation. Because the proposed system is linear and based on statistics, many results can be explained and improved mathematically. Let us explain individual modules more detailed in the following sections.

III. Feature extraction with variable cuboid size

In this section, we introduce the feature extraction module, which is linear and hence both forward direction (data to extracted features) and inverse direction (features to original data) are feasible. The extracted features can be assumed to be uncorrelated Gaussian random variables. Those points are explained in the following subsections.

A. Forward Direction

The proposed feature extraction module is a fast linear data-driven feedforward transformation with low space complexity. Referring to the block diagram in Fig. 1, let $\mathbf{Q}(n) \in R^{I\times J}$ be the nth preprocessed image of a size of $I \times J$, where $n = 1, 2, \cdots, N$. Hence N is the total number of the input images. In the testing phase, the value of N can be one; while in the

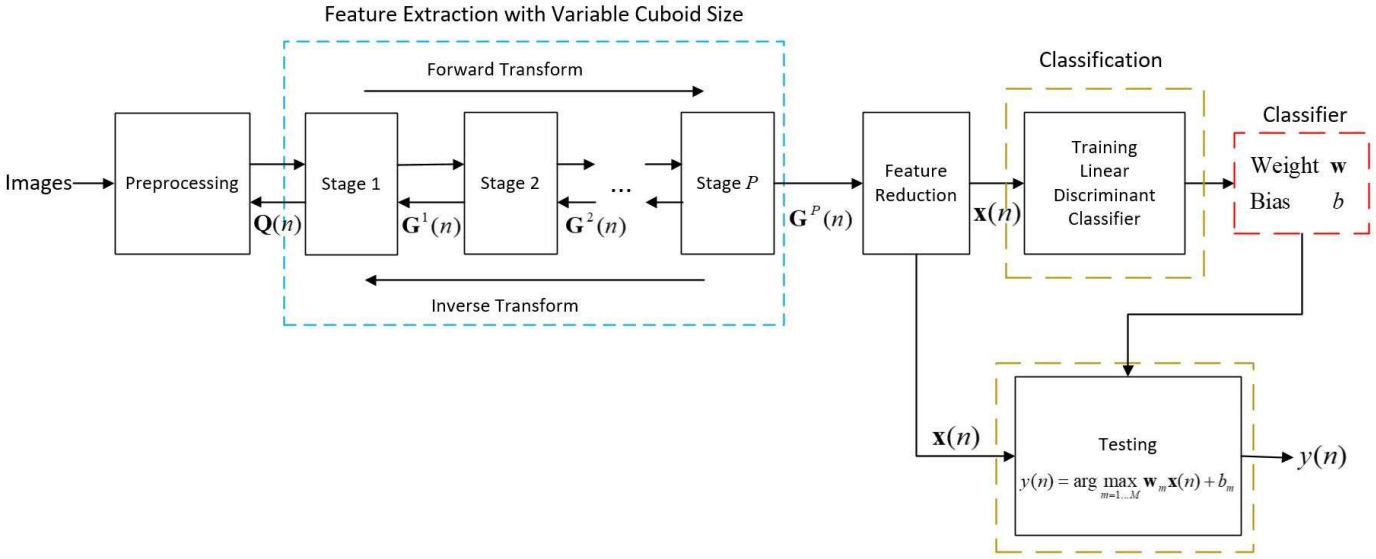

Fig. 1. The proposed linear classification system consisting of the preprocessing, feature extraction, feature reduction, and classification modules.

training phase, it has to be more than one to find the transform kernels. Then, we feed all of the preprocessed images into the multi-stage proposed scheme and let the initial global cuboid $\mathbf{G}^0(n) \in R^{I^0 \times J^0 \times K^0}$ be $\mathbf{Q}(n)$, where the superscript is the stage index and $I^0 = I$, $J^0 = J$ and $K^0 = 1$.

Assume that there are P stages. At each of stage, we reshape the global cuboid into multiple non-overlapping local cuboids, perform principal component analysis on individual cuboids, collect the results, and reshape them into another global cuboid for the next stage. According to the size of input images, one can adjust the side lengths of the local cuboids in the vertical and horizontal direction at each stage. Let the side lengths of the local cuboids in stage p be $l_i^p \times l_j^p \times l_k^p$, where l_i^p, l_j^p are adjustable. The values of l_i^p, l_j^p and l_k^p satisfy

$$\prod_{p=1}^{P} l_i^p = I, \quad I^{p-1}/l_i^p \in N, \tag{1}$$

$$\prod_{p=1}^{P} l_j^p = J, \quad J^{p-1}/l_j^p \in N, \tag{2}$$

$$l_k^p = K^{p-1}. \tag{3}$$

When the initial input and the side lengths l_i^p, l_j^p are given, the input can be processed by the multi-stage scheme. The dataflow and the dimension conversion at Stage p are shown in Fig. 2 and also explained in Steps 1-3 below: The stage index p is with increasing order, *i.e.,* $p = 1, 2, \cdots P$.

Step 1. Global cuboid to several local cuboids. Let the global cuboid of the nth image at Stage $p-1$ be $\mathbf{G}^{p-1}(n)$ with size $I^{p-1} \times J^{p-1} \times K^{p-1}$. The global cuboid is cut into several local cuboids $\mathbf{G}_{i,j}^{p-1}(n)$ with size $l_i^p \times l_j^p \times l_k^p$ at stage p. With the constraints of the local cuboid side lengths, one can perfectly cut the global cuboid into $I^p \times J^p$ non-overlapping

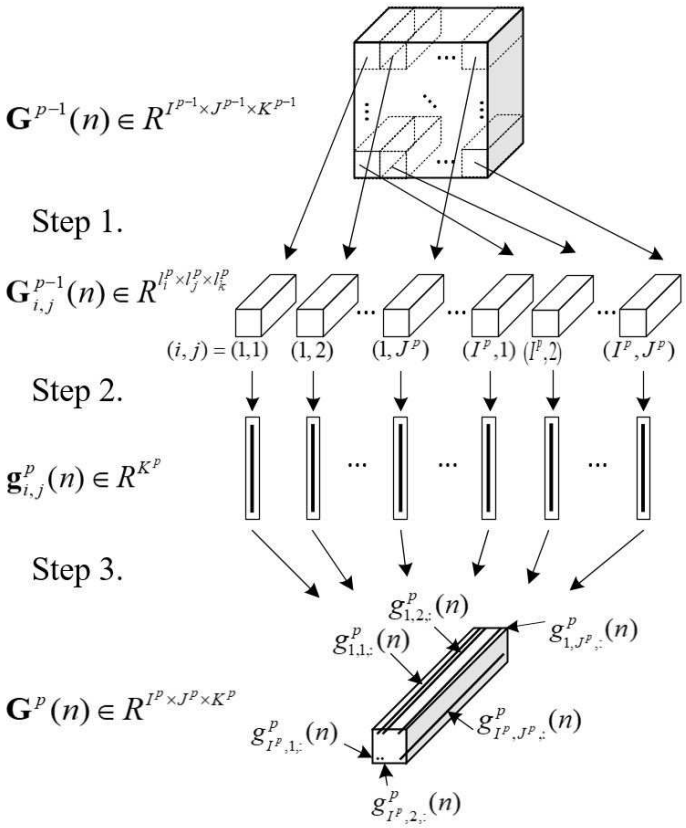

Fig. 2. The dataflow, reshape operations and dimensions of cuboids between Steps $p-1$ and p.

local cuboids below

$$\mathbf{G}^{p-1}(n) = \begin{bmatrix} \mathbf{G}_{1,1}^{p-1}(n) & \mathbf{G}_{1,2}^{p-1}(n) & \cdots & \mathbf{G}_{1,J^p}^{p-1}(n) \\ \mathbf{G}_{2,1}^{p-1}(n) & \mathbf{G}_{2,2,:}^{p-1}(n) & \cdots & \mathbf{G}_{2,J^p}^{p-1}(n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{G}_{I^p,1}^{p-1}(n) & \mathbf{G}_{I^p,2}^{p-1}(n) & \cdots & \mathbf{G}_{I^p,J^p}^{p-1}(n) \end{bmatrix}, \tag{4}$$

where $I^p = I^{p-1}/l_i^p$ and $J^p = J^{p-1}/l_j^p$. For instance, letting the side lengths be 4. If the image at Stage $p-1$ is with size 64×64, the size becomes 16×16 at Stage p.

Step 2. Principal component analysis to local cuboids. The principal components are used as the transform kernels. Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of possibly correlated observations into linearly uncorrelated sets. When data vectors are projected onto the principal components, the PCA coefficients have larger variances which can help in separating data into various classes.

First, we reshape the local cuboids $\mathbf{G}_{i,j}^{p-1}(n)$ to vectors $\mathbf{f}_{i,j}^p(n) \in R^{K^p}$, where K^p is equal to the volume of the local cuboid defined as

$$K^p = l_i^p \times l_j^p \times l_k^p, \tag{5}$$

and the reshape operation is defined as

$$\mathbf{f}_{i,j}^p(n) = \text{reshape}(\mathbf{G}_{i,j}^{p-1}(n), K^p, 1) \in R^{K^p}. \tag{6}$$

Second, the principal components of the vectors $\mathbf{f}_{i,j}^p(n)$ are calculated. The principal components are the eigenvectors of the covariance matrix of the data vectors. Let the mean for all data vectors be $\bar{\mathbf{f}}_{i,j}^p$ given by

$$\bar{\mathbf{f}}_{i,j}^p = \frac{1}{N}\sum_{n=1}^{N} \mathbf{f}_{i,j}^p(n). \tag{7}$$

The covariance matrix $\mathbf{R}_{i,j}^p$ is then calculated using

$$\mathbf{R}_{i,j}^p = \frac{1}{N}\sum_{n=1}^{N} (\mathbf{f}_{i,j}^p(n) - \bar{\mathbf{f}}_{i,j}^p)(\mathbf{f}_{i,j}^p(n) - \bar{\mathbf{f}}_{i,j}^p)^T. \tag{8}$$

Then we find the K^p eigenvectors $\mathbf{B}_{i,j}^p$ of $\mathbf{R}_{i,j}^p$, where each $[\mathbf{B}_{i,j}^p]_k$ is the kth eigenvector of $\mathbf{R}_{i,j}^p$, and $k = 1, 2, \cdots, K^p$. Note that the notation $[\mathbf{A}]_k$ is the kth column of matrix $\mathbf{A}$. Hence $[\mathbf{B}_{i,j}^p]_k$ are the principal components of $\mathbf{f}_{i,j}^p(n)$. Third, we project the vectors $\mathbf{f}_{i,j}^p(n)$ onto the transform kernels (principal components) and obtain the PCA coefficients $\mathbf{g}_{i,j}^p(n)$ defined as

$$\mathbf{g}_{i,j}^p(n) = \mathbf{B}_{i,j}^{p\,T} \mathbf{f}_{i,j}^p(n). \tag{9}$$

Step 3. Reshape PCA coefficients and form another global cuboid $\mathbf{G}^p(n)$. We then reshape all the PCA coefficients obtained in Step 2 to place the coefficient vectors in the spectral direction using the following procedure:

$$g_{i,j,:}^p(n) = \text{reshape}(\mathbf{g}_{i,j}^p(n), 1, 1, K^p). \tag{10}$$

Combining all $g_{i,j,:}^p(n)$, where $i = 1, 2, \cdots, I^p$ and $j = 1, 2, \cdots, J^p$, global cuboid $\mathbf{G}^p(n)$ for next stage can be formed given by

$$\mathbf{G}^p(n) = \begin{bmatrix} g_{1,1,:}^p(n) & g_{1,2,:}^p(n) & \cdots & g_{1,J^p,:}^p(n) \\ g_{2,1,:}^p(n) & g_{2,2,:}^p(n) & \cdots & g_{2,J^p,:}^p(n) \\ \vdots & \vdots & \ddots & \vdots \\ g_{I^p,1,:}^p(n) & g_{I^p,2,:}^p(n) & \cdots & g_{I^p,J^p,:}^p(n) \end{bmatrix}, \tag{11}$$

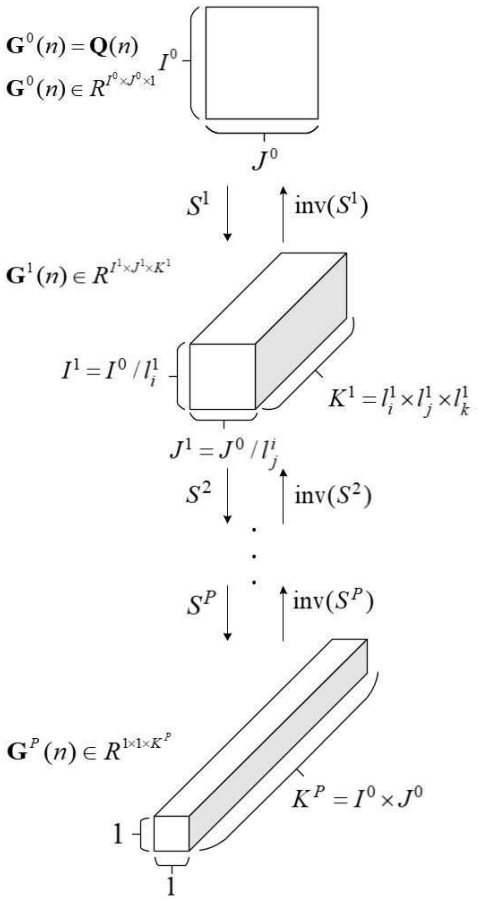

Fig. 3. The dimension conversion of the global cuboid through stages. S^p and $\text{inv}(S^p)$ are the forward and inverse of the proposed scheme between stage p and $(p-1)$, respectively.

where $\mathbf{G}^p(n) \in R^{I^p \times J^p \times K^p}$. The dimension conversion of global cuboids from Stage $p-1$ to Stage p can be written as

$$I^p = I^{p-1}/l_i^p, \tag{12}$$

$$J^p = J^{p-1}/l_j^p, \tag{13}$$

$$K^p = l_i^p \times l_j^p \times l_k^p. \tag{14}$$

Stop criterion. The whole process is stopped when the last stage, Stage P, is approached, and the global cuboids $\mathbf{G}^P(n) \in R^{1 \times 1 \times K^P}$ is obtained.

At the final stage, one can obtain the final PCA coefficients given by

$$\mathbf{G}^P(n) = g_{1,1,:}^P(n) \in R^{1 \times 1 \times K^P}, \tag{15}$$

where K^P is with the constraints of l_i^p, l_j^p and l_k^p that $K^P = I \times J$. As a result, there is no growth in space complexity and the space complexity is $O(1)$ throughout the whole process. Note that the space complexity doubles at each stage for the system in [7]. Then the global cuboid at the final stage can be reshaped to the PCA coefficient vector $\mathbf{g}^P(n)$ at the final stage:

$$\mathbf{g}^P(n) = \text{reshape}(\mathbf{G}^P(n), K^P, 1). \tag{16}$$

The PCA coefficients at the final stage are also called features that are used to determine the class of the input images.

B. Gaussian Approximation of Features

As mentioned in Step 2 of the previous subsection, the PCA is a procedure that can convert a set of possibly correlated coefficients into uncorrelated ones. Hence the coefficients at the final stage should be almost uncorrelated. To see this, the $(i,j)-th$ element of the correlation coefficient matrix $\rho_{\mathbf{g}^P,\mathbf{g}^P}$ of the PCA coefficients at the final stage is defined as

$$[\rho_{\mathbf{g}^P,\mathbf{g}^P}]_{i,j} = \frac{\sum_{n=1}^{N}([\mathbf{g}^P(n)]_i - [\bar{\mathbf{g}}^P]_i)([\mathbf{g}^P(n)]_j - [\bar{\mathbf{g}}^P]_j)}{\sqrt{\sum_{n=1}^{N}([\mathbf{g}^P(n)]_i - [\bar{\mathbf{g}}^P]_i)^2 \sum_{n=1}^{N}([\mathbf{g}^P(n)]_j - [\bar{\mathbf{g}}^P]_j)^2}}, \tag{17}$$

where $\bar{\mathbf{g}}^P$ is the mean of $\mathbf{g}^P(n)$ defined as

$$\bar{\mathbf{g}}^P = \frac{1}{N}\sum_{n=1}^{N}\mathbf{g}^P(n). \tag{18}$$

For example, let $\mathbf{g}^P$ be the feature vector that will be detailed in Experiment 1 in Sec. VI, where the dimension is 90. Then, the obtained correlation coefficient matrix is shown in Fig. 5. From the figure, we see that the PCA coefficients at the final stage are almost uncorrelated. In addition to the uncorrelated relationship among features, we also find that the statistics of individual features can be approximated by Gaussian distribution. The histograms of some randomly picked samples are shown in Fig. 4, in which Gaussian approximation well matches the histograms. In addition to this dataset, we have also verified various datasets and observed similar Gaussian approximation results of the proposed system.

Knowing that the features are nearly uncorrelated Gaussian distribution, we can treat the features as the Gaussian mixture model (GMM). The GMM phenomenon was also reported in [7]. Approximating the features using the GMM is useful and motivates us to use the linear discriminant classifier, a MAP-based detector, that will be introduced in Sec. V.

C. Inverse Direction

Since the proposed scheme is a linear transformation, both forward and inverse transformations can be conducted. The inverse transformation is to reconstruct the PCA coefficients $\mathbf{g}^P(n)$ at the final stage to the preprocessed image $\mathbf{Q}(n)$. We elaborate on the inverse transformation in the following steps: Now the stage index p is with decreasing order, *i.e.,* $p = P, P-1, \cdots, 1$.

Step 1. Reshape the PCA coefficients back to vectors $\mathbf{g}_{i,j}^p(n)$. This is a simple inverse procedure of (10) given by

$$\mathbf{g}_{i,j}^p(n) = \text{reshape}(g_{i,j,:}^p(n), K^p, 1). \tag{19}$$

Step 2. Project the the vectors onto the inverse transform kernel. Referring to (9), the inverse kernel of $\mathbf{B}_{i,j}^{p\,T}$ is simply its inverse matrix. Hence the backward result of this step can be easily obtained by

$$\mathbf{f}_{i,j}^p(n) = (\mathbf{B}_{i,j}^{p\,T})^{-1}\mathbf{g}_{i,j}^p(n), \tag{20}$$

where $(\mathbf{B}_{i,j}^{p\,T})^{-1}$ is the inverse matrix of $\mathbf{B}_{i,j}^{p\,T}$.

Step 3. Reshape $\mathbf{f}_{i,j}^p(n)$ back to the local cuboids. Referring to (6), this step is given by

$$\mathbf{G}_{i,j}^{p-1}(n) = \text{reshape}(\mathbf{f}_{i,j}^p(n), l_i^p, l_j^p, l_k^p). \tag{21}$$

Step 4. Form global cuboid and take PCA coefficients. The method to collect the local cuboids to form global cuboid is the inverse of (4), and the method for taking the PCA coefficients for the previous stage is the inverse of (11).

Conduct Steps 1-4 for all P stages, the preprocessed images $\mathbf{Q}(n)$ can be recovered losslessly.

IV. Feature Reduction

The invertibility is an important property of the proposed system. One can reduce the features and thus keep significant elements of $\mathbf{g}^P(n)$. By inversing $\mathbf{g}^P(n)$ with reduced features, the original image can still be reconstructed with important features reserved. One may treat this as a lossy data compression dedicated for classification purposes or feature filtering. Next, let us introduce how to reduce the features and how to recover data from the reduced features.

A. Proposed Methods for Feature Reduction

To reserve the features that have the largest discriminations, the PCA can again be applied for this purpose. The property is that when data projects onto its significant principal components, the principal coefficients have larger variances. As a result, the principal coefficients with larger variances can be considered more significant and have larger discriminant power than those with smaller ones. Hence the problem reduces to keeping the features that have the largest variances. The variance vector of the features are defined as

$$\boldsymbol{\sigma}_{\mathbf{g}^P} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{g}^P(n) - \bar{\mathbf{g}}^P)^2, \tag{22}$$

where $\bar{\mathbf{g}}^P$ is the mean defined in (18). When $\boldsymbol{\sigma}_{\mathbf{g}^P}$ is obtained, one can find the set of indices $\mathcal{I}$ which corresponds to the D largest variances defined as

$$\mathcal{I} = \{i \,|\, [\boldsymbol{\sigma}_{\mathbf{g}^P}]_i \,\text{are the}\, D \,\text{largest variances}\} \in R^D. \tag{23}$$

Once the set $\mathcal{I}$ is found, the features can be reduced accordingly. The feature vector after the feature reduction $\mathbf{x}(n)$ can be obtained by

$$\mathbf{x}(n) = [\mathbf{g}^P(n)]_{\mathcal{I}} \in R^D. \tag{24}$$

The feature vector has a size reduction from K^P to D. Also, $\mathbf{x}(n)$ contains the top few significant features of the nth image and represents this image in training or testing phase.

Determining a suitable value of D is important. This value can be determined in the training phase. More specifically, given a set of training data, one can set D so that the classier has the best performance and this will be shown in the simulation results later.

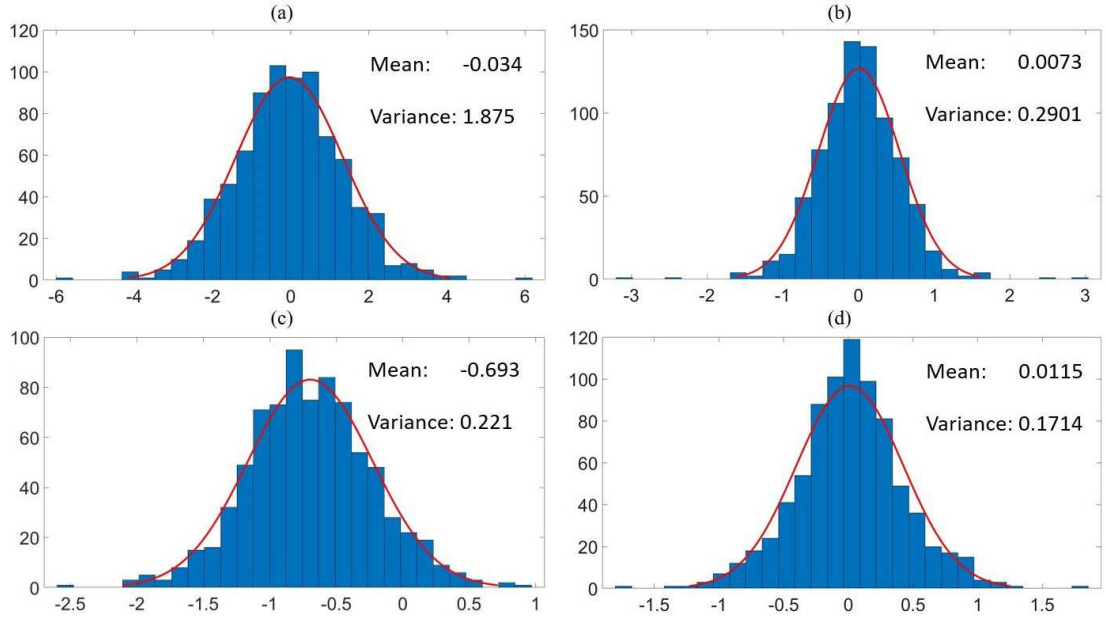

Fig. 4. Gaussian approximation of the extracted features in the proposed system. (a) Feature No. 12, (b) Feature No. 55, (c) Feature No. 72, (d) Feature No. 90 from inputs which belong to class 5.

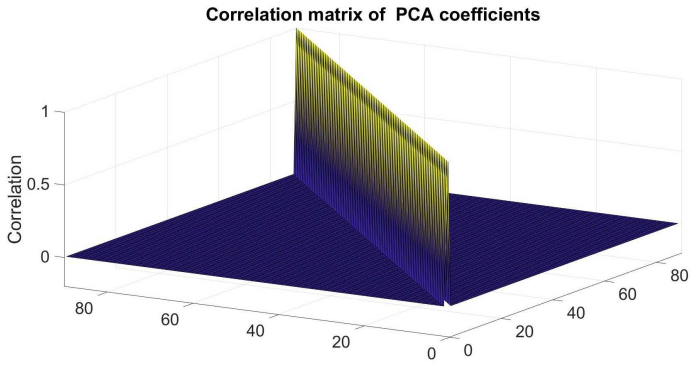

Fig. 5. The correlation matrix of the extracted features, which is nearly diagonal with diagonal elements being 1 and the off-diagonal elements being smaller than 1.0e-05.

B. Recover Images from Reduced Features

As mentioned in Sec. (III-C), the features before feature reduction $\mathbf{g}^P$ can be used to reconstruct the preprocessed images losslessly. This is also true for the features after feature reduction $\mathbf{x}(n)$ with lossy image reconstruction. The inverse transform is different only at Stage P, which is elaborated below. For Stage P:

Step 1. Form transform kernel at Stage P with feature reduction. The transform kernel $\hat{\mathbf{B}}_{i,j}^P$ can be obtained by extracting the columns indexed by set $\mathcal{I}$ given by

$$\hat{\mathbf{B}}_{i,j}^P = \left[[\mathbf{B}_{i,j}^P]_{\mathcal{I}}\right] \in R^{K^p \times D}. \tag{25}$$

Step 2. Project features after feature reduction. The feature vector $\mathbf{x}(n)$ is projected onto the Moore-Penrose pseudo-inverse matrix of transform kernel, *i.e.,* $(\hat{\mathbf{B}}_{i,j}^{P\,T})^{\text{pinv}} \in R^{K^P \times D}$ and yield

$$\hat{\mathbf{f}}_{i,j}^P(n) = (\hat{\mathbf{B}}_{i,j}^{P\,T})^{\text{pinv}}\mathbf{x}(n) \in R^{K^P \times 1}, \tag{26}$$

where $(\hat{\mathbf{B}}_{i,j}^{P\,T})^{\text{pinv}}$ is the pseudo-inverse matrix of $\hat{\mathbf{B}}_{i,j}^{P\,T}$ in (25).

Step 3. Reshape to local cuboids. The vector $\hat{\mathbf{f}}_{i,j}^P(n)$ is shaped to the local cuboids,

$$\hat{\mathbf{G}}_{i,j}^{P-1}(n) = \text{reshape}(\hat{\mathbf{f}}_{i,j}^P(n), l_i^P, l_j^P, l_k^P). \tag{27}$$

After Steps 1-3 for Stage P, the same procedure for Steps 1-4 introduced in Sec.III-C shall be conducted for other stages from $P-1$ to 1. After P-stage of inverse transformation, the initial global cuboid $\hat{\mathbf{G}}^0(n)$ is reconstructed. Due to the energy loss from the eliminated features, image processing is needed for reducing the reconstruction loss. First, we compensate the energy loss using a brightness gap $h(n)$ defined as

$$h(n) = \frac{1}{I \cdot J}\sum_{i=1}^{I}\sum_{j=1}^{J}([\mathbf{Q}(n)]_{i,j} - [\hat{\mathbf{G}}^0(n)]_{i,j}). \tag{28}$$

Second, we apply histogram equalization to enhance the contrast and fix the range span as well. The histogram equalization was widely used, see *e.g.,* [28] and [29]. Here, we use the function histeq provided by the Matlab. Finally, the recovered image can be obtained with a slight loss $\hat{\mathbf{Q}}(n)$ defined as

$$\hat{\mathbf{Q}}(n) = \text{histeq}(\hat{\mathbf{G}}^0(n) + h(n)). \tag{29}$$

In Fig. 6, we provide an example using an image from the testing dataset that will be introduced in Sec. VI later. The original features of the image in the red layer is 4096, and we recover the image using only 90 features. In row (a), the red layer of the original image and its histogram are shown. In row (b), we show the recovered image without any image processing $\hat{\mathbf{G}}^0(n)$ and we can see the image is darker than the original one and the contrast is poor, though one can still recognize the image. In row (c), the brightness has been compensated by adding $h(n)$ defined in (28). Hence the mean

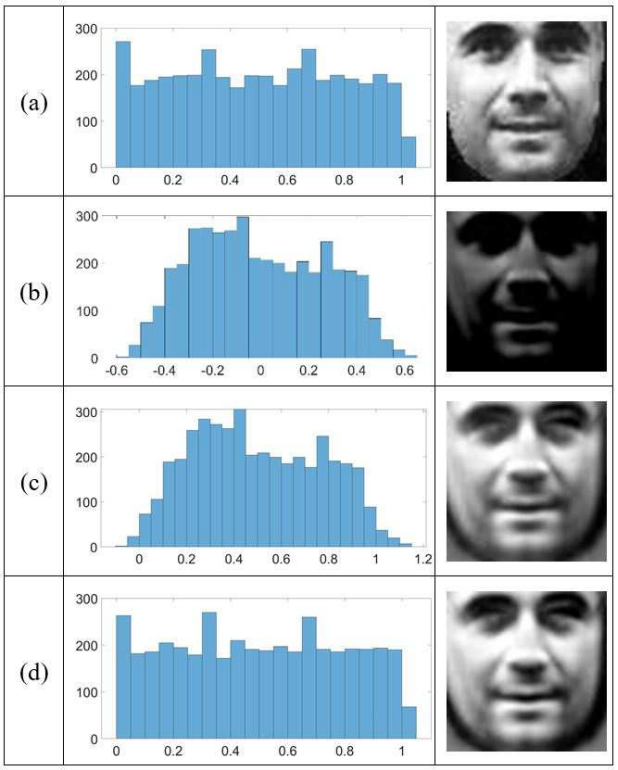

Fig. 6. (a): Original image and its histogram. (b): Recovered image without any image processing. (c): Recovered image after adding brightness. (d): Recovered image after adding brightness and histogram equalization.

of the histogram shifts. In row (d), we show the recovered image $\hat{\mathbf{Q}}(n)$ using (29). One can see that no matter from the image or its histogram, they are the closest to those in row (a). Thus the image processing in (28) and (29) really helps when recovering images from reduced few features.

When $D < K^P$, reconstructing images from the reduced features can be regarded as a lossy data decompression. Such data compression and decompression are meaningful for data classification. More specifically, if only D most significant features are needed to achieve a target classification performance, one can simply store the data set with D features instead of K^P features because both have comparable classification performance. Namely, we only store the top D significant features $\mathbf{x}(n) \in R^D$, instead of the original data $\mathbf{Q}(n) \in R^{I \times J}$. Such a concept may be used to reduce the data size and computational complexity in classification algorithms. The compression ratio r with feature reduction can then be written as

$$r = \frac{I \times J}{D} = \frac{K^P}{D}. \tag{30}$$

The examples in Fig. 6 (a) and (d) thus have a compression ratio of $4096 : 90 = 45.51 : 1$.

V. Linear Discriminant Classifier

After the feature reduction, the reduced features can be fed into the classifier, which is introduced here. As discussed in Sec. III-B, the features can be approximated to be uncorrelated Gaussian random variables. Hence classical theory in estimation and detection can be used and the resulting performances have theoretical explanations. Now we have the reduced feature vector $\mathbf{x}(n)$ and would like to determine which class it belongs to. To treat such a problem, classical maximum a posteriori (MAP) estimator provides a good reference. In the area of machine learning and pattern recognition, this concept is usually called linear discriminant analysis (LDA), which finds a linear combination of features that can separate two or more classes. Here we combine the two concepts, LDA and MAP estimator to design the linear discriminant classifier introduced as follows:

First, let $y(n)$ be the output of the classifier, which is the class that $\mathbf{x}(n)$ belongs to, and C_m be Class m. The MAP estimator is given by

$$y(n) = \arg \max_{m=1 \cdots M} P(C_m|\mathbf{x}(n)), \tag{31}$$

where M is the number of classes. The posterior probability is defined as

$$P(C_m|\mathbf{x}(n)) = \frac{P(\mathbf{x}(n)|C_m)P(C_m)}{P(\mathbf{x}(n))}, \tag{32}$$

where $P(\mathbf{x}(n))$ is written as

$$P(\mathbf{x}(n)) = \sum_{m=1}^{M} P(\mathbf{x}(n)|C_m)P(C_m). \tag{33}$$

Since $P(\mathbf{x}(n))$ is a constant for all classes, one can ignore the denominator in (32). As for the numerator in (32), since we consider the input features are multivariate Gaussian distributed, the priori probability can be written as

$$P(C_m) = \frac{n_m}{N}, \tag{34}$$

where n_m is the number of training images which are with class m, and $N = \sum_{m=1}^{M} n_m$ is the number of all training images. The mean of $\mathbf{x}_m(n)$ is defined as

$$\mu_m = \frac{1}{n_m} \sum_{n=1}^{n_m} \mathbf{x}_m(n), \tag{35}$$

and the covariance matrix $\mathbf{\Sigma}_m$ is defined as

$$\mathbf{\Sigma}_m = \frac{1}{n_m} \sum_{n=1}^{n_m} (\mathbf{x}_m(n) - \boldsymbol{\mu}_m)^T (\mathbf{x}_m(n) - \boldsymbol{\mu}_m), \tag{36}$$

where $\mathbf{x}_m(n)$, $n = 1, 2, \cdots, n_m$, are the feature vectors belong to class m.

In the linear discriminant classifier, we need a common covariance matrix so as to keep the likelihood function linear. In this work, we propose to use the pooled covariance matrix $\mathbf{\Sigma}$ as the common matrix. The pooled covariance matrix is defined as

$$\mathbf{\Sigma} = \sum_{m=1}^{M} \frac{n_m}{N} \mathbf{\Sigma}_m. \tag{37}$$

The likelihood function $P(\mathbf{x}(n)|C_m)$ for multivariate Gaussian random variables is then given by

$$P(\mathbf{x}(n)|C_m) = \tag{38}$$

$$\frac{1}{\sqrt{(2\pi)^D |\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2} (\mathbf{x}(n) - \boldsymbol{\mu}_m)^T \mathbf{\Sigma}^{-1} (\mathbf{x}(n) - \boldsymbol{\mu}_m) \right), \tag{39}$$

where recall that D is the number of reduced features.

To simplify the calculation, referring to (32), we take the natural logarithm, and ignore the denominator of the posterior probability function $P(C_m|\mathbf{x}(n))$, *i.e.,*

$$y(n) = \arg \max_{m=1\cdots M} \ln P(\mathbf{x}(n)|C_m)P(C_m), \tag{40}$$

The term $\ln P(\mathbf{x}(n)|C_m)P(C_m)$ can be rewritten as

$$\begin{aligned} &\ln P(\mathbf{x}(n)|C_m)P(C_m) \\ &= \ln(\frac{1}{\sqrt{(2\pi)^D|\mathbf{\Sigma}|}}) \\ &- \frac{1}{2}(\mathbf{x}(n)^T\mathbf{\Sigma}^{-1}\mathbf{x}(n) - 2\boldsymbol{\mu}_m^T\mathbf{\Sigma}^{-1}\mathbf{x}(n) + \boldsymbol{\mu}_m^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_m) \\ &+ \ln(\frac{n_m}{N}). \end{aligned} \tag{41}$$

From (37), since the covariance is common, $\ln(\frac{1}{\sqrt{(2\pi)^D|\mathbf{\Sigma}|}})$ and $-\frac{1}{2}\mathbf{x}(n)^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{x}(n)$ can be ignored. Then the remaining term can be written in a linear form consisting of a weight $\mathbf{w}_m$ and a bias term b_m for an individual class given by

$$\boldsymbol{\mu}_m^T\mathbf{\Sigma}^{-1}\mathbf{x}(n) - \frac{1}{2}\boldsymbol{\mu}_m^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_m + \ln(\frac{n_m}{N}) = \mathbf{w}_m\mathbf{x}(n) + b_m, \tag{42}$$

where the term $\mathbf{w}_m = \boldsymbol{\mu}_m^T\mathbf{\Sigma}^{-1}$ is the weight and the term $b_m = -\frac{1}{2}\boldsymbol{\mu}_m^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_m + \ln(\frac{n_m}{N})$ is the bias. Finally, the classifier can be written in a linear equation form as follows:

$$y(n) = \arg \max_{m=1\cdots M} \mathbf{w}_m\mathbf{x}(n) + b_m \ . \tag{43}$$

We summarize the proposed scheme for training and testing including the linear transformation of feature extraction, feature reduction and linear discriminant classifier in respectively in Algorithms 1 and 2.

VI. Experimental Results

In this section, the experiment results are provided to show the performance of the proposed classification system.

Dataset and Hardware. Labeled Faces in the Wild (LFW) is a well known academic test set for face verification [1]. There are two datasets used in the experiments. For the first data set, 158 classes are selected from the whole dataset which has more than 6 images for training and 2 images to testing per class. For the second data set, 19 classes are selected out of the 158 classes in the first data set, which has more than 30 images for training and 10 images for testing. Table I details the size of the two datasets. Intel(R) Core(TM) i7-8700CPU and 16GB RAM is used to conduct the experiments.

Image Preprocess. Firstly, the face is detected and the background of the image is blacked via an open-source python file [30]. Then all the images are resized to 64×64. An overview of the images after the face detection operation and resizing is shown in Fig. 7. Secondly, the training images are augmented. The reason is that in the first dataset, there are some classes that have only 6 images for training, which is insufficient to train a good model. Also, some faces do not face the same directions as shown in Fig. 7. Therefore, we augment the training images by flipping the images horizontally and make the number of images doubles. Thirdly, we separate the three primary colors of images into R, G, B 3 layers and then equalize the 3 layers of histogram individually. This step enhances the contrast of images and enable the separation of the three primary color layers.

After preprocessing, the training data are passed through the proposed system, and Algorithm 1 is used to process the data.

Algorithm 1: Training Process

Input: N preprocessed training images $\mathbf{Q}(n) \in R^{I\times J}$. Local cuboid spatial sizes $l_i^p \times l_j^p$ in stage p for $p = 1, 2, \cdots, P$.

Output: Transform kernels $\mathbf{B}_{i,j}^p$. Weight $\mathbf{w}_m$ and bias b_m. Remaining indices $\mathcal{I}$.

Initialization: $\mathbf{G}^0(n) = \mathbf{Q}(n)$. Let global cuboid size be $I^0 \times J^0 \times K^0$, where $I^0 = I$, $J^0 = J$ and $K^0 = 1$.

- - - - - - - - - - - - **Feature Extraction** - - - - - - - - - - -

for $p = 1 : P$ **do**

  Form local cuboids $\mathbf{G}_{i,j}^{p-1}(n)$ using (4) and reshape them to vectors $\mathbf{f}_{i,j}^p(n)$ using (6).

  $I^p = I^{p-1}/l_i^p$ and $J^p = J^{p-1}/l_j^p$

  for $i = 1 : I^p$ **do**

    for $j = 1 : J^p$ **do**

      Find the transform kernel (PCs) $\mathbf{B}_{i,j}^p$ using eigenvector-based method.

      Project $\mathbf{f}_{i,j}^p(n)$ onto $\mathbf{B}_{i,j}^p$ to obtain PCA coefficients $\mathbf{g}_{i,j}^p(n)$ using (9).

      Reshape $\mathbf{g}_{i,j}^p(n)$ to $g_{i,j,:}^p(n)$ to form global cuboid $\mathbf{G}^p(n)$ using (11).

    end for

  end for

end for

Obtain final PCA coefficients $g_{i,j,:}^P(n)$ at the final stage and reshape them to vector $\mathbf{g}^P(n)$.

- - - - - - - - - - - - **Feature Reduction** - - - - - - - - - - -

Reduce $\mathbf{g}^P(n)$ to obtain the feature vector $\mathbf{x}(n)$ using Sec. IV, and store the indices $\mathcal{I}$ of remaining features.

For colored images, repeat 1-15 for each primary color layer, and cascade all features to a vector $\mathbf{x}(n)$.

- - - - - - - **Linear Discriminant Classifier** - - - - - - -

Feed $\mathbf{x}(n)$ and labels into the Linear Discriminant Classifier, and obtain weight $\mathbf{w}_m$ and bias b_m.

Fig. 7. An overview of images after face detection and resizing.

Algorithm 2: Testing Process

Input: The N preprocessed testing image(s) $\mathbf{Q}(n) \in R^{I\times I}$. Local cuboid spatial sizes $l_i^p \times l_j^p$ same as Algo. 1. Transform kernels $\mathbf{B}_{i,j}^p$. Weight $\mathbf{w}_m$ and bias b_m. Remaining indices $\mathcal{I}$.

Output: The classification accuracy.

Initialization: $\mathbf{G}^0(n) = \mathbf{Q}(n)$. Let global cuboid size be $I^0 \times J^0 \times K^0$, where $I^0 = I$, $J^0 = J$ and $K^0 = 1$.

- - - - - - - - - - - - **Feature Extraction** - - - - - - - - - - -

for $p = 1 : P$ **do**
  Repeat Steps 3-4 in Algo. 1.
  for $i = 1 : I^p$ **do**
    for $j = 1 : J^p$ **do**
      Read the transform kernel $\mathbf{B}_{i,j}^p$.
      Repeat Steps 8-9 in Algo. 1.
    end for
  end for
end for
Repeat Step 13 in Algo. 1.

- - - - - - - - - - - - **Feature Reduction** - - - - - - - - - - -

Reduce features according to remaining indices $\mathcal{I}$ to obtain feature vector $\mathbf{x}(n)$.
Repeat Step 15 in Algo. 1.

- - - - - - - **Linear Discriminant Classifier** - - - - - - -

Use the classifier in (43) to estimate which class the nth image belongs to.
Use label(s) to check the classification accuracy.

TABLE I
DATASET DETAILS

# classes	Training data per class	Testing data per class	Total training data	Total testing data
19	≥ 30	≥ 10	1402	460
158	≥ 6	≥ 2	3296	1014

Experiment 1: Testing accuracy as functions of various numbers of features. As discussed in Sec. IV, the number of features id reduced from K^P to D to avoid overfitting in training classifier, where $K^P = 4096$ for one primary color layer in the experiments. For designing a face recognition model, the main goal is to achieve high testing accuracy, and thus D is decided mainly according to the testing accuracy. Let the local cuboid sizes be set as $(l_i^1, l_j^1, l_k^1) = (8, 8, 1)$, $(l_i^2, l_j^2, l_k^2) = (4, 4, 64)$, $(l_i^3, l_j^3, l_k^3) = (2, 2, 1024)$, which is a 3-stage scheme. Table II shows the testing accuracy corresponding to 8 different numbers of features. Several observations are summarized: First, more features do not necessarily lead to better testing accuracy. This is because more features may result in overfitting problem and hence feature reduction is needed. Second, for both datasets, the best testing accuracy occurs when $D = 90$ for one primary color layer and a total of 270 features for three layers. Consequently, the compression ratio is $1 : 45.5$, which is a significant reduction of data size.

Experiment 2: Testing accuracy and computations as functions of various local cuboid sizes. In this experiment, three sets of variable local cuboid sizes are used to figure out efficient local cuboid sizes. Efficiency considers the computational speed and testing accuracy. The computational speed depends on the number of stages. When the local cuboid sizes are large, the proposed multi-stage scheme has fewer stages and hence fewer computations, but the feature extraction might lose the precision of extracting the most significant features, and it leads to bad accuracy in the end. Thus a good local cuboid size should be determined for achieving high testing accuracy with low computations. We consider the following three settings of local cuboid sizes: **Setting 1:** There is one stage with $(l_i^1, l_j^1, l_k^1) = (64, 64, 1)$. **Setting 2:** There are 3 stages with $(l_i^1, l_j^1, l_k^1) = (8, 8, 1)$, $(l_i^2, l_j^2, l_k^2) = (4, 4, 64)$, $(l_i^3, l_j^3, l_k^3) = (2, 2, 1024)$. **Setting 3:** There are 6 stages with $(l_i^1, l_j^1, l_k^1) = (2, 2, 1)$, $(l_i^2, l_j^2, l_k^2) = (2, 2, 4)$, $(l_i^3, l_j^3, l_k^3) = (2, 2, 16)$, $(l_i^4, l_j^4, l_k^4) = (2, 2, 64)$, $(l_i^5, l_j^5, l_k^5) = (2, 2, 256)$, $(l_i^6, l_j^6, l_k^6) = (2, 2, 1024)$. Here, the number of extracted features is fixed to 270, which leads the best testing accuracy according to Experiment 1. The results are shown in Table III. Observed from the table that Settings 2 and 3 both achieve the best accuracy. However, Setting 2 has a lower computational cost than that in Setting 3. Hence Setting 2 is an efficient setting of local cuboid sizes for this experiment.

TABLE II
TESTING ACCURACY AS FUNCTIONS OF VARIOUS NUMBERS OF FEATURES

# features	# classes		# features	# classes	
	19	158		19	158
12288	12.61%	0.10%	300	97.17%	83.53%
9000	5.43%	0.10%	**270**	**97.61%**	**84.71%**
6000	7.39%	23.08%	240	97.17%	84.12%
3000	6.09%	61.24%	90	93.91%	78.60%
900	96.09%	81.66%	45	87.61%	63.51%

Experiment 3: Performance with multiple candidates. Following Experiments 1 and 2, we show all the images that are incorrectly recognized from the 19-classes testing dataset in Fig. 8. We see that there are some problematic images which are marked with red and blue rectangles. In the images with red rectangles, the faces are covered by hands; while in the image with a blue rectangle, the error actually cames from image preprocessing (face detection operation). When we eliminate those problematic images or select a better face detection operation, the testing accuracy can be improved from 97.6% to 98.5%. Similarly, in Fig. 9, we displayed the incorrectly recognized problematic images from the 158-classes testing dataset. Some of them are seriously affected by sunglasses, hands, and even other's shoulder. When we exclude these errors, the accuracy improves from 84.71% to 86.3%.

Additionally, because the proposed scheme classifies the images using a MAP-like estimator, for each image the classifier outputs the individual probabilities for individual classes that this image belongs to. Hence it is possible to determine several candidates with the highest probabilities that one image belongs to. Refined and improved algorithms can be developed to determine the best decision from the candidates with the highest probabilities. It is worth mentioning that some classifiers cut space into regions and do not have the cluster

TABLE III
TESTING ACCURACY AND COMPUTATIONS AS FUNCTIONS OF VARIOUS LOCAL CUBOID SIZES.

	19-classes			158-classes		
Setting	Time(Train)	Time(Test)	Accuracy	Time(Train)	Time(Test)	Accuracy
1	9s	0.1s	5.0%	10s	0.3s	0.3%
2	**11s**	**0.3s**	**97.61%**	**16s**	**1.1s**	**84.71%**
3	25s	1.2s	97.61%	61s	3.2s	84.71%

Fig. 8. All the 11 incorrectly recognized images from the 19-classes dataset. There are some problematic images boxed in red and blue rectangles. If they are eliminated, the accuracy can reach 98.5%.

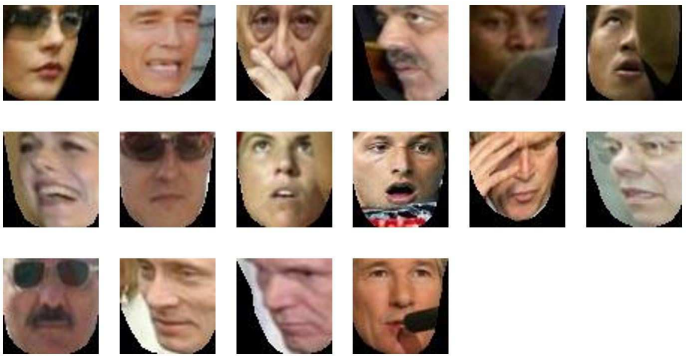

Fig. 9. All the incorrectly recognized problematic images from the 158-classes dataset. If these unavoidable errors are eliminated, the accuracy can reach 86.3%.

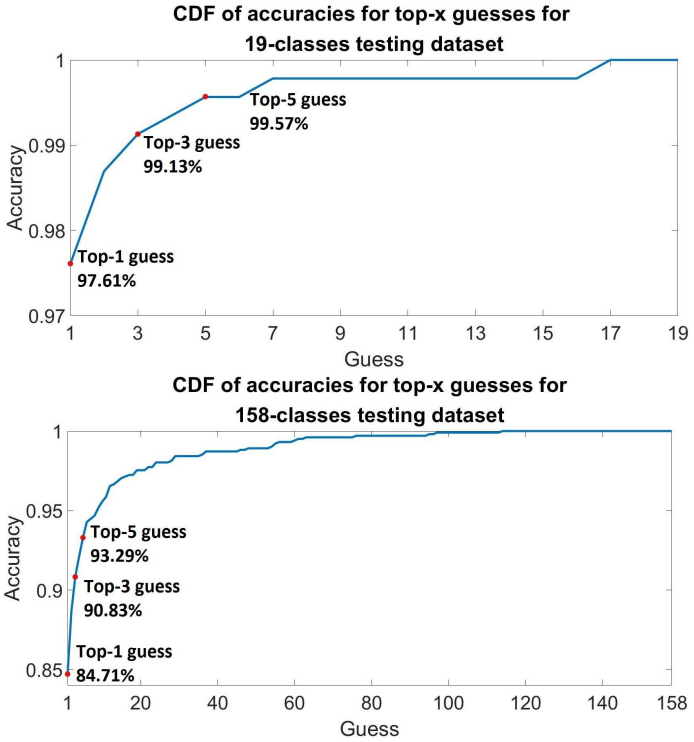

Fig. 10. The CDF of top-X guesses for both testing datasets. The accuracies of both datasets reach above 90% when having top-3 guesses, 99.1% for the 19-class dataset and 93.1% for the 158-class dataset.

center in advance, such as SVM. Such schemes need to pay more effort if more candidates are to be selected. Here we show the testing accuracy of top-3 and top-5 candidates. The accuracy for 158-class dataset reaches 93.1% when having top-3 guesses, and 95.7% when having top-5 guesses. Moreover, the accuracy for 19-class dataset reaches 99.1% when having top-3 guesses, and 99.6% when having top-5 guesses.

The CDFs of top guesses are shown in Fig. 10.

Experiment 4: Image reconstruction using various numbers of features. In this experiment, we reconstruct images from various numbers of features. The reasons are two folded. The first one is to see how many features are sufficient to reconstruct the original images from a point of view of data compression. The second reason is to see what features and why they are eliminated in the original images. In Experiments 1 and 2, we see using full features may lead to overfitting problem. By reversing the images from the reduced features, the corresponding results can also be used to explain what may lead to overfitting problem and what are the important features of the original images that can be used to mostly distinguish images from different classes. Here, we recover images from 12288 (full features), 9000, 6000, 3000, 270 features which respectively have 4096, 3000, 2000, 1000, 90 features for recovering one primary color layer. To compare the recovery result, we use the percent deviation to calculate the loss defined as

$$l = \frac{\sqrt{\sum_{i=1}^{I^0}\sum_{j=1}^{J^0}\sum_{k=1}^{K^0}\left([\hat{\mathbf{Q}}(n)]_{i,j,k} - [\mathbf{Q}(n)]_{i,j,k}\right)^2}}{I^0 \times J^0 \times K^0}. \tag{44}$$

Table IV shows the corresponding percent deviations and compression ratio. It is observed that using more features does not necessarily lead to a small percent deviation due to overfitting issue. In addition, reconstructing images from 270 features has a very high compression ratio and a satisfactory low percent deviation. Note that the deviation for 12288 is very small but nonzero due to the accumulated computational error of multiple stages in the platform.

Let us see how the recovered images look like. Fig. 11, shows some reconstructed sample images. Row (a) are the samples recovered from full 12288 features and the average percent deviation is 1.7226e-04%, which proves that we can

TABLE IV
COMPRESSION RATIOS

Features	Percent deviation	Compression ratios
12288	1.7e-04%	1:1
9000	5.48%	1.37:1
6000	12.41%	2.05:1
3000	15.51%	4.10:1
270	**9.14%**	**45.51:1**
240	9.31%	51.2:1
90	11.45%	136.5:1

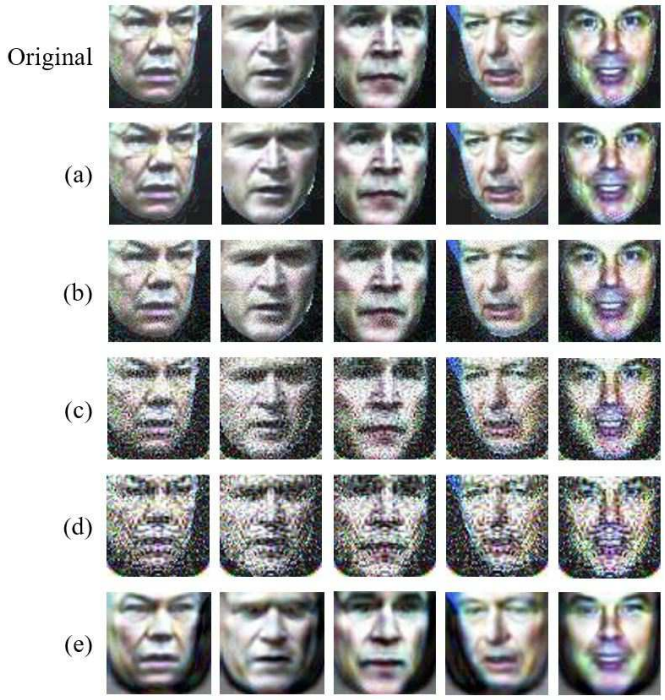

Fig. 11. Some image recovery samples. (a) are the samples of recovery images recovered from 12288 features, the percent deviation is $1.7226e-04\%$. (b) to (e) are recovered from reduced features, and the numbers of features are 9000, 6000, 3000, 270, the percent deviations are $5.48\%, 12.41\%, 15.51\%, 9.14\%$.

recover the image losslessly from full features. Row (b) are the samples recovered from 9000 features and the average percent deviation is 5.48%. Rows (c) and (d) are the samples recovered from 6000 and 3000 features and the average percent deviations are 12.41% and 15.51% respectively. Observing that as the reduced features increase in this level, the deviations grow higher. Also in rows (c) and row (d), the reconstructed images contain certain insignificant details which look like noises. As a result, the reconstruction quality and the testing accuracy are poor due to overfitting as we can see in the previous example in Table II. When the number of features reduces to 270, however, in row (e), we see that the insignificant details disappear and the effect just like passing the images through a smoothing process. Consequently, the deviation is only 9.14% which is even lower than row (c) and row (d). This result also reflects that 270 features are suitable for training the classifier since they grasp most of the significant information of an image and avoid overfitting occurred as using 6000 and 3000 features.

Moreover from the top and bottom samples of the first

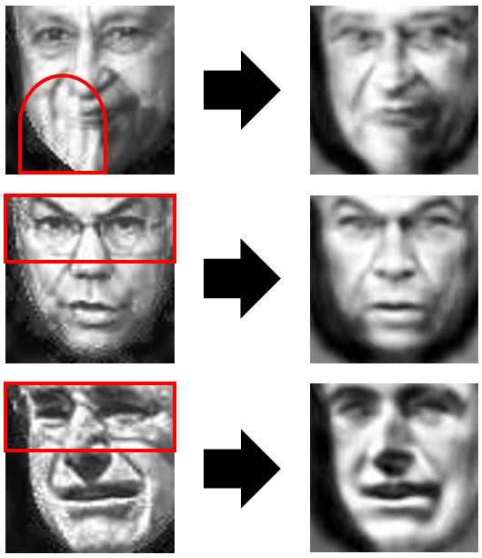

Fig. 12. The image recovery method eliminates disturbing objects. The top image originally has fingers in front of the man's chin and disappears after recovered. The middle and the bottom images both have glasses at first but disappear after recovered.

column in Fig. 11, the glasses disappear in the reconstructed image from the reduced 270 features. This implies that the proposed scheme can remove redundancy that is irrelevant or unimportant to classification. To see this point more clearly, Fig. 12 shows more reconstructed images from the 270 features. Observed from the figure that glasses and figures disappear. That is, significant features extracted by the proposed scheme would be the information about face structures, not those disturbing objects like glasses and fingers. Therefore, when we reconstruct the images from a suitable amount of features, the disturbing items should disappear. From a viewpoint of data compression, the compressed data is dedicated to achieving better classification performance. This effect may be treated as "feature filtering" of the proposed system in classification.

Experiment 5: Comparisons between conventional and proposed schemes. In this experiment, we compare the proposed system with the AlexNet and the Saak transform in [7]. The same preprocessed training and testing images are used. We apply the AlexNet provided by Matlab. The CNN model is trained using exactly the same layers like those in AlexNet. Moreover, considering the hardware, we allow that the CNN provided by Matlab uses GPU to speed up, and the proposed scheme only uses CPU. The GPU that CNN uses is NVIDIA GeForce GTX 1050 Ti. For the Saak transform, 600 features are used, which can achieve its best accuracy. The result is shown in Table V. We see that the proposed model achieves a better accuracy while the whole computational time including training and testing is only one twenty-fifth of that with CNN.

VII. CONCLUSION AND FUTURE WORK

We have proposed a linear classification system that can inverse the extracted and reduced features to original data,

TABLE V
ALEXNET VS. SAAK TRANSFORM VS. PROPOSED RECOGNITION MODEL

	AlexNet			Saak Transform (600 features)			Proposed Scheme (270 features)		
# of classes	Time (Train)	Time (Test)	Accuracy	Time (Train)	Time (Test)	Accuracy	Time (Train)	Time (Test)	Accuracy
19	550s	4.1s	$\approx 86\%$	218s	14.3s	87.83%	11s	0.3s	$\mathbf{97.61}\%$
158	1260s	12.6s	$\approx 63\%$	2032s	49.3s	61.93%	16s	1.1s	$\mathbf{84.71}\%$

and achieve data compression for classification purposes as well. Experimental results show that the proposed system outperforms the conventional classification schemes in terms of not only computational complexity but also testing accuracy. From the viewpoint of data compression, the proposed system compresses the data in a way beneficial for classification purposes. That is, when the images are recovered from the reduced features via the proposed system, several unimportant feature redundancies for classifications such as glasses and covered hand on the faces are naturally filtered out. We call this effect feature filtering. When the compressed data is used for classification, the testing accuracy is high while the recovered images can still achieve a small percent deviation as well. The nice properties including linearity, reversibility, achieving data compression and feature filtering have made the proposed system worth further investigation. The solutions to develop sophisticated algorithms to refine the detection results among top guesses, and to efficiently utilize the advantages of data compression for classification are still open.

REFERENCES

[1] F. Schroff, D. Kalenichenko and J. Philbin, FaceNet: a unified embedding for face recognition and clustering, *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 815823, Jun. 2015.

[2] H. Wang, J. Hu and W. Deng, Compressing fisher vector for robust face recognition, *IEEE Access,* vol. 5, pp. 2315723165, Sep. 2017.

[3] J. Lu, V. E. Liong, X. Zhou, J. Zhou, Learning compact binary face descriptor for face recognition, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, pp. 20412056, Mar. 2015.

[4] C. Ding and D. Tao, Robust face recognition via multimodal deep face representation, *IEEE Trans. Multimedia*, vol. 17, pp. 20492058, Nov. 2015.

[5] B.-C. Chen, C.-S. Chen and W. H. Hsu, Face recognition and retrieval using cross-age reference coding with cross-age celebrity dataset, *IEEE Trans. Multimedia*, vol. 17, pp. 804815, Apr. 2015.

[6] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, *Computer Vision and Pattern Recognition (CVPR),* pp. 770778, Dec. 2016.

[7] C.-C. J. Kuo and Y. Chen, On data-driven saak transform, *arXiv preprint arXiv:1710.04176,* Oct. 2017.

[8] T. H. Chan, K. Jia, S. Gao, et al.: Pcanet: a simple deep learning baseline for image classification?, *IEEE Trans. Image Process.*, vol. 24, pp. 50175032, Dec. 2015.

[9] L. Zhao and Z. Guo, Face recognition method based on adaptively weighted block-two dimensional principal component analysis, *Proc. of IEEE Int. Conf. on Computational Intelligence, Communication Systems and Networks*, pp. 22-25, Jul. 2011.

[10] I. Dagher and R. Nachar, Face recognition using IPCA-ICA algorithm, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, pp. 9961000, Jun. 2006.

[11] S. Rajendran, A. Kaul, R. Nath, A. Arora and S. Chauhan, Comparison of pca and 2d-pca on indian faces, *Signal Propagation and Computer Technology (ICSPCT), 2014 International Conference on. IEEE.*, pp. 561566, Jul. 2014.

[12] R. He, B.-G. Hu, W.-S. Zheng and X.-W. Kong, Robust principal component analysis based on maximum correntropy criterion, *IEEE Trans. Image Process.*, vol. 20, pp. 14851494, Jun. 2011.

[13] H. M. Ebied, Feature extraction using pca and kernel-pca for face recognition, *2012 8th International Conference on Informatics and Systems (INFOS).*, pp. MM72MM77, May 2012.

[14] X. Xiao and Y. Zhou, Two-dimensional quaternion PCA and sparse PCA, *IEEE Trans. on Neural Networks and Learning Systems.*, vol. 30, pp. 115, Jul. 2018.

[15] Y. Pei, Linear principal component discriminate analysis, *2015 IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC2015).*, vol. 28, pp. 21082113, Feb. 2015.

[16] J. Xu, M. Li, X. Zhao and Z. Chang, Self-learning super-resolution using convolutional principal component analysis and random matching, *IEEE Trans. Multimedia.*, vol. 21, pp. 1108-1121, Sep. 2018.

[17] Y. Choi, T. Tokumoto, M. Lee and S. Ozawa, Incremental two-dimensional two-directional principal component analysis (l(2D)^{2}PCA) for face recognition, *IEEE International Conference on Acoustics, Speech and Signal Processing.*, pp. 1493-1496, May 2011.

[18] X. Jiang, Linear subspace learning-based dimensionality reduction,*IEEE Signal Process. Mag.*, vol. 28, pp. 16-26, Mar. 2011.

[19] C. Liu and H. Wechsler, Gabor feature based classification using the enhanced Fisher linear discriminant model for face recognition,*IEEE Trans. Image Process.*, vol. 11, pp. 467476, Apr. 2002.

[20] F. Ye, Z. Shi and Z. Shi, A comparative study of PCA, LDA and kernel LDA for image classification, *International Symposium on ubiquitous virtual reality, IEEE.*, pp. 51-54, Jul. 2009.

[21] Z. Zeng and P. Huang, Palmprint recognition using Gabor feature-based two-directional two-dimensional linear discriminant analysis, *IEEE International Conference on EMEIT.*, vol. 4, pp. 1917-1921, Aug. 2011.

[22] C. Xiang and D. Huang, Feature extraction using recursive cluster-based linear discriminant with application to face recognition, *IEEE Transactions on Image Processing.*, vol. 15, pp. 38243832, Nov. 2006.

[23] T.-K. Kim and J. Kittler, Locally linear discriminant analysis for multimodally distributed classes for face recognition with a single model image, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, pp. 318327, Mar. 2005.

[24] S. W. Ji and J. P. Ye, Generalized linear discriminant analysis: a unified framework and efficient model selection, *IEEE Transactions on Neural Networks.*, vol. 19, pp. 1768-1782, Sep. 2008.

[25] M. H. Siddiqi, R. Ali, A. M. Khan, Y. T. Park, S. Lee, et al.: Human facial expression recognition using stepwise linear discriminant analysis and hidden conditional random fields, *IEEE Trans. Image Process.*, vol. 24, pp. 13861398, Feb. 2015.

[26] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, 1993.

[27] University of Massachusetts Amherst, *Labeled Faces in the Wild.*, [Online]. Available: http://vis-www.cs.umass.edu/lfw/#information

[28] N. Senthilkumaran and J. Thimmiaraja, Histogram equalization for image enhancement using MRI brain images, in *2014 World Congress on Computing and Communication Technologies.*, pp. 8083, Mar. 2014.

[29] J. H. Han, S. Yang and B. U. Lee, A novel 3-D color histogram equalization method with uniform 1-D gray scale histogram, *IEEE Trans.on Image Processing.*, vol. 20, pp. 506-512, Feb. 2011.

[30] Dlib, *shape predictor 68 face landmarks.*, [Online]. Available: http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2