End-to-End Feature-Aware Label Space Encoding for Multilabel Classification With Many Classes

Zijia Lin, *Student Member, IEEE*, Guiguang Ding, *Member, IEEE*, Jungong Han, Ling Shao, *Senior Member, IEEE*

***Abstract*— To make the problem of multilabel classification with many classes more tractable, in recent years, academia has seen efforts devoted to performing label space dimension reduction (LSDR). Specifically, LSDR encodes high-dimensional label vectors into low-dimensional code vectors lying in a latent space, so as to train predictive models at much lower costs. With respect to the prediction, it performs classification for any unseen instance by recovering a label vector from its predicted code vector via a decoding process. In this paper, we propose a novel method, namely End-to-End Feature-aware label space Encoding (E^2FE), to perform LSDR. Instead of requiring an encoding function like most previous works, E^2FE directly learns a code matrix formed by code vectors of the training instances in an end-to-end manner. Another distinct property of E^2FE is its feature awareness attributable to the fact that the code matrix is learned by jointly maximizing the *recoverability* of the label space and the *predictability* of the latent space. Based on the learned code matrix, E^2FE further trains predictive models to map instance features into code vectors, and also learns a linear decoding matrix for efficiently recovering the label vector of any unseen instance from its predicted code vector. Theoretical analyses show that both the code matrix and the linear decoding matrix in E^2FE can be efficiently learned. Moreover, similar to previous works, E^2FE can be specified to learn an encoding function. And it can also be extended with kernel tricks to handle nonlinear correlations between the feature space and the latent space. Comprehensive experiments conducted on diverse benchmark data sets with many classes show consistent performance gains of E^2FE over the state-of-the-art methods.**

***Index Terms*— End-to-end feature-aware label space encoding, label space dimension reduction (LSDR), multilabel classification.**

I. Introduction

As a generalized version of multiclass classification [1]–[5], where each instance is restricted to having only one class label, multilabel classification [6]–[23] allows an instance to be associated with several class labels to describe its semantic content or attributes more clearly. Multilabel classification methods are increasingly demanded by modern applications, such as multilabel text classification [6], music emotion categorization [7], and semantic image annotation [21]–[23]. In addition, many studies on neural networks and other learning approaches are also dedicated to the field, such as the tree-structure based method ML-TREE [20], the multiview vector-valued manifold regularization method MV^{3}MR [21], and the label inference method LI-MLC [24].

Recently, due to the emergence of Web-based applications, multilabel classification problems tend to be large scale, with new challenges of numerous instances and large label sets (i.e., high-dimensional label spaces) coming up. For instance, in the picture sharing community Flickr, there are billions of images and each can be annotated with textual labels selected from millions of candidates. In the community of neural networks and related learning systems, to handle the challenges, some works such as [25]–[29] focus on feature dimension reduction or model simplification, while others such as LI-MLC [24] focus on shrinking the label space. Here, we follow the latter one.

As advocated in [8], large label sets cause many existing effective multilabel classification methods [9]–[18] to be infeasible, since generally they need to learn a predictive model for each label independently or with interlabel correlations, and then combine them in a certain manner for prediction. Specifically, for a multilabel classification problem with many classes (i.e., a large label set or a high-dimensional label space), the number of needed predictive models would generally be large, thus making the training costs, if not unaffordable, extremely high. To tackle this issue, researchers have recently proposed to perform label space dimension reduction (LSDR) [8], [30]–[35], which aims to reduce the training costs while maintaining acceptable classification performance. Specifically, for LSDR, as shown in Fig. 1, the high-dimensional label vector of any training instance is encoded into a low-dimensional code vector in a latent space.

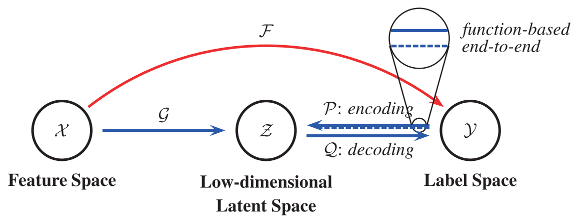

Fig. 1. Illustration of the principles behind traditional multilabel classification methods (red) and those with LSDR (blue).

Manuscript received February 29, 2016; revised November 22, 2016 and March 26, 2017; accepted March 26, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61571269 and in part by the Royal Society Newton Mobility under Grant IE150997. *(Corresponding author: Guiguang Ding.)*

Z. Lin is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: linzijia07@tsinghua.org.cn).

G. Ding is with the School of Software, Tsinghua University, Beijing 100084, China (e-mail: dinggg@tsinghua.edu.cn).

J. Han is with the Department of Computer Science and Digital Technologies, Northumbria University, Newcastle NE2 1UY, U.K. (e-mail: jungong.han@northumbria.ac.uk).

L. Shao is with the School of Computing Sciences, University of East Anglia, NR4 7TJ Norwich, U.K. (e-mail: ling.shao@ieee.org).

This paper contains supplemental material available online at http://ieeexplore.ieee.org (File size: 1 MB).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2017.2691545

Afterward, predictive models are trained to map instance features into low-dimensional code vectors, whose quantity is much smaller and thus can significantly reduce the training costs. As for performing prediction for any unseen instance, a low-dimensional code vector is first obtained with the learned predictive models from its features, and then decoded for recovering its label vector. Generally speaking, if the learned predictive models and the decoding process are effective and efficient enough, LSDR usually yields acceptable classification performance with much lower costs, making the multilabel classification problem with many classes more tractable.

Prior methods dedicated to LSDR mostly require an encoding function (function-based), e.g., a linear one, to map label vectors of training instances into code vectors lying in the latent space. However, due to the following observations, we argue that learning the code vectors of training instances in an end-to-end manner, i.e., directly learning them without any encoding functions, can be feasible and even preferable.

1) From Fig. 1, it can be seen that, to perform prediction, the encoding process is totally redundant, and thus, any encoding function is useless during prediction. Moreover, even for training, it is the encoding result (i.e., code vectors of training instances) that will affect the learning of predictive models, no matter whether an encoding function is required or not.
2) Defining an encoding function may limit the searching space of the to-be-learned code vectors of training instances. For example, given the tagging matrix $\mathbf{Y}$ of training instances, using a linear encoding function $\mathbf{P}$ can limit the to-be-learned code vectors in the space $\mathbf{YP}$, thereby preventing them from being searched in the whole real space that could potentially minimize the loss of classification performance.
3) In some cases, code vectors of training instances are required to have specific properties, such as the orthonormality between code dimensions in this paper. Although those property requirements can somehow be transferred to the encoding function, it will inevitably make the objective function much more complex for optimization.

In fact, compared with a function-based encoding, an end-to-end encoding requires no encoding function, and thus can search the whole real space for the optimal to-be-learned code vectors. Moreover, for an end-to-end encoding, it would be direct to add property requirements for the to-be-learned code vectors, making the objective function less complex for optimization. To the best of our knowledge, MLC-BMaD [34] is the only previous research that pioneered end-to-end label space encoding via Boolean matrix decomposition. However, as will be shown later, its training is not efficient enough, and as a result, it may not fully accomplish the goal of LSDR. Moreover, MLC-BMaD learns the code vectors of training instances in a *feature-unaware* manner, meaning that the correlations between the latent space and the feature space are not considered. That, as advocated in [33], can probably make the learned latent space less predictable and thus degrade the final classification performance. Therefore, further studies on end-to-end label space encoding are highly expected.

In this paper, we propose a novel method termed E^{2}FE to perform LSDR via **E**nd-to-**E**nd **F**eature-aware label space **E**ncoding. Specifically, E^{2}FE directly learns a code matrix formed by code vectors of training instances via jointly maximizing the *recoverability* of the label space and the *predictability* of the latent space, with the latter considering the correlations between the latent space and the feature space. And thus E^{2}FE is *feature-aware*. Based on the learned code matrix, predictive models are trained as other LSDR methods, to predict code vectors from instance features. Meanwhile, E^{2}FE further learns a linear decoding matrix that can recover the predicted label vector of any unseen instance from its code vector generated by the trained predictive models.

Since the predictive models in the proposed E^{2}FE are open for any effective ones, including neural networks, E^{2}FE can actually be applied with existing predictive models or feature dimension reduction approaches in the community to better tackle the large-scale multilabel classification problem. Particularly for LSDR, below are three highlighted properties of E^{2}FE, which are in line with our contributions.

1) We propose an effective LSDR method termed E^{2}FE for tackling (large-scale) multilabel classification problems with many classes. To the best of our knowledge, it is the first to make LSDR both end-to-end and feature-aware.
2) We jointly maximize the *recoverability* of the label space and the *predictability* of the latent space for performing LSDR in E^{2}FE. The objective function with respect to the to-be-learned code matrix can be transformed to an eigenvalue problem, and is sufficiently flexible in the sense that different optimization strategies can be used depending on the applications for efficient optimization.
3) We show that E^{2}FE is a generic approach that covers previous LSDR studies, and it can also be specified to learn an encoding function. Moreover, it can be extended with kernel tricks to handle nonlinear correlations between the feature space and the latent space.

This paper is based on our previous work presented in [36], which was termed FaIE, but it substantially extends that work by enhancing the proposed method to be more efficient and effective. The summarized extensions are shown in the following list.

1) We propose a more efficient optimization method for the proposed method to learn the code matrix in cases where $n \gg d_x + d_y$, with n, d_y, and d_x, respectively, denoting the number of training instances, the dimensionality of the label space, and that of the feature space. This is helpful for practical applications, as such cases are quite common. Specifically, the newly proposed optimization method transforms the size of the eigenvalue problem with respect to the objective function of E^{2}FE from $\mathbb{R}^{n\times n}$ to $\mathbb{R}^{(d_x+d_y)\times(d_x+d_y)}$, which can be solved more efficiently and can substantially reduce space costs.
2) We further propose πE^{2}FE, πLinearE^{2}FE, and *kernel*-πE^{2}FE, to consider a *priori* knowledge provided by the eigenvalue problem with respect to the to-be-learned code matrix for learning an enhanced decoding matrix. Experiments comparing πE^{2}FE, πLinearE^{2}FE, and *kernel*-πE^{2}FE with their corresponding counterparts,

i.e., E^2FE, LinearE^2FE, and *kernel*-E^2FE, show that enhancing the decoding matrix with such *a priori knowledge* can help to gain significant performance improvements (on average 48.1% for *label-based macroF1* and 33.9% for *example-based accuracy*).

3) In this paper, we provide a thorough discussion and experimental validation for that the orthonormality assumption for columns of the to-be-learned code matrix in E^2FE is reasonable. We also make error analyses for the proposed E^2FE, and derive its error bound. Additionally, more theoretical analyses, such as those regarding time complexity and parameter settings, are also presented here.

4) To better validate the effectiveness of E^2FE, we utilize more widely used benchmark data sets for experiments. We also conduct the experiments on the full data sets instead of the sampled ones in [36], so as to demonstrate the applicability of E^2FE for handling larger data sets. More experimental results are also reported, such as the significance tests for the improvements gained by E^2FE over compared baselines, and the comparison of computational costs between the newly proposed optimization method here and that presented in [36].

The remainder of this paper is organized as follows. Section II gives an overview of related works. Section III elaborates on the proposed E^2FE. Section IV shows the proposed optimization methods and its corresponding theoretical analyses. Section V describes the details about enhancing the linear decoding matrix with *a priori* knowledge. Then, Section VI presents the extensions of E^2FE, and analyzes its relations to previous works. Experimental settings, results, and analyses are given in Section VII. Finally, we present discussions regarding E^2FE in Section VIII and conclude this paper in Section IX.

II. Related Work

With the explosion of label spaces in real-world applications, many remarkable effective multilabel classification methods tend to be infeasible due to the high training costs. To tackle such multilabel classification problems with many classes, a lot of effective methods were proposed, such as constructing a hierarchy of multilabel classifiers [37], refining the output of heuristic efficient classifiers [38], performing label selection to recover the vocabulary with only a subset [39], or using label inference method based on the use of association rules to discover label dependences [24]. Recently, LSDR was also proposed and is attracting more and more attention.

To the best of our knowledge, Hsu *et al.* [30] are the first to propose LSDR. Specifically, Hsu *et al.* [30] exploited the sparsity of the label space, and proposed to linearly encode it to a low-dimensional latent space as compressed sensing (CS) and then train linear regression models with respect to the derived codes. As for performing classification for an unseen instance, a code vector is first obtained with the learned regression models from its features and then decoded with standard recovery algorithms, such as CoSaMP [40] to derive the predicted label vector. Kapoor *et al.* [8] further considered both label space compression and predictive model learning

TABLE I
Categorization of Existing LSDR Methods and E^2FE

	feature-unaware	feature-aware
function-based	CS [30], PLST [31], CL [32], ML-CSSP [35]	BML-CS [8], CPLST [33]
end-to-end	MLC-BMaD [34]	E^2FE

in a single probabilistic model, and derived a Bayesian framework termed BML-CS for multilabel classification via jointly optimizing over both.

Apart from CS-based methods, Tai and Lin [31] proposed to perform principle label space transformation (PLST) for seeking important correlations between labels, which is essentially Principal Component Analysis (PCA) [41] for the label space. Chen and Lin [33] further enhanced it by proposing feature-aware conditional principal label space transformation (CPLST), which actually integrates orthogonally constrained canonical correlation analysis into the framework of PLST for considering the *predictability* of the latent space. Both PLST and CPLST performed LSDR via linear encoding and linear decoding. Zhou *et al.* [32] proposed another method termed "compressed labeling," which takes the signs of the linear Gaussian random projection results on the original label vectors as the derived code vectors and utilizes a series of Kullback–Leibler divergence-based hypothesis tests for decoding. Alternatively, Wicker *et al.* [34] proposed MLC-BMaD for LSDR via Boolean matrix decomposition on the binary tagging matrix, factorizing it as the product of a binary code matrix and a binary linear decoding matrix. Bi and Kwok [35] presented an efficient randomized sampling procedure termed ML-CSSP for selecting a column subset of the tagging matrix that can well span it, which is a special case of linear encoding.

Actually, the majority of existing methods perform LSDR in a function-based manner and require an encoding function. Such approaches, as analyzed in Section I, carry several drawbacks. To avoid those, performing LSDR in an end-to-end manner with no need for any encoding function is highly desired. MLC-BMaD seems to be the only existing LSDR method that supports end-to-end label space encoding via boolean matrix factorization. However, MLC-BMaD is feature-unaware, and thus, the learned latent space could be less predictable, which can result in performance deterioration. Therefore, in this paper, we propose E^2FE, which performs LSDR in an end-to-end manner and is also feature-aware.

To sum up, Table I categorizes the remarkable existing LSDR methods and the proposed E^2FE into different combinations of {function-based, end-to-end} and {feature-unaware, feature-aware}, which well highlights the distinctness of E^2FE.

III. Proposed Approach

A. Preliminaries

Generally, in the case of multilabel classification, the features of an instance are represented as a d_x-dimensional feature vector $\mathbf{x}$ in the feature space $\mathcal{X}$, i.e., $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{d_x}$, and its labels are represented as a d_y-dimensional binary label vector $\mathbf{y}$ in the label space $\mathcal{Y}$, i.e., $\mathbf{y} \in \mathcal{Y} \subset \{0, 1\}^{d_y}$. Here, the ith entry of the label vector $\mathbf{y}$ is set as 1 if the instance is associated with the ith label and 0 otherwise. Suppose that we are given n labeled instances for training, denoted as

$\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{n}$, with $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$ being the feature vector and the label vector of the ith training instance. Multilabel classification will utilize them to learn the mapping $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ from the feature space $\mathcal{X}$ to the label space $\mathcal{Y}$, as shown in Fig. 1, and then utilize $\mathcal{F}$ for predicting the label vector of any unseen instance based on its feature vector.

As mentioned earlier, to derive the mapping $\mathcal{F}$, many existing effective multilabel classification methods will learn a predictive model for each label independently or with interlabel correlations, and then combine them in a certain manner for prediction. In that case, the number of the to-be-learned predictive models will be at least d_y, and even much larger for methods using label powerset [42]. Then, for a multilabel classification problem with many classes, d_y will become quite large and the training costs of the to-be-learned predictive models will be extremely high and even unaffordable. To tackle such a challenge, LSDR was recently proposed and is attracting more and more attention. With LSDR, the training process to learn $\mathcal{F}$ is transformed into a two-step learning process. That is, first the label vectors of training instances are encoded into low-dimensional code vectors in a latent space $\mathcal{Z} \subset \mathbb{R}^{d_z}$ with an encoding process $\mathcal{P} : \mathcal{Y} \mapsto \mathcal{Z}$, and then, a mapping $\mathcal{G} : \mathcal{X} \mapsto \mathcal{Z}$ with respect to the code vectors is learned. Here, d_z is the dimensionality of the latent space $\mathcal{Z}$, and generally, $d_z \ll d_y$. Moreover, as shown in Fig. 1, $\mathcal{P}$ can be performed in a function-based manner (e.g., linear encoding function) or an end-to-end manner (e.g., matrix decomposition). Similar to learning $\mathcal{F}$, learning $\mathcal{G}$ can be based on training d_z predictive models, one for a dimension of $\mathcal{Z}$. As for predicting the labels of any unseen instance, a d_z-dimensional code vector in $\mathcal{Z}$ will first be derived using the learned $\mathcal{G}$ with its feature vector, and then, a d_y-dimensional predicted label vector will be recovered through a decoding process $\mathcal{Q} : \mathcal{Z} \mapsto \mathcal{Y}$. For LSDR methods, with $d_z \ll d_y$, the number of the to-be-learned predictive models is generally much smaller, and thus, the training costs are substantially lowered, making the multilabel classification problem with many classes more tractable. Meanwhile, if the mapping $\mathcal{G}$ and the decoding process $\mathcal{Q}$ are effective enough, the classification performance using LSDR is expected to be acceptable.

It should be noticed that for LSDR, the latent space $\mathcal{Z}$ is supposed to be derived from the label space $\mathcal{Y}$ rather than the feature space $\mathcal{X}$, even though $\mathcal{X}$ can sometimes be considered for increasing the *predictability* of $\mathcal{Z}$. And thus the dimensionality of the latent space (i.e., d_z) can either be higher or lower than that of the feature space (i.e., d_x), but will always be lower than that of the label space (i.e., d_y). Moreover, for LSDR methods, the mapping $\mathcal{G}$ from $\mathcal{X}$ to $\mathcal{Z}$ is open for any effective mapping algorithm after $\mathcal{Z}$ is derived. Meanwhile, the decoding process $\mathcal{Q}$ generally needs to be specified before deriving $\mathcal{Z}$, which, from the perspective of efficiency in prediction, is preferred to be linear, such as those in PLST, CPLST, MLC-BMad, and ML-CSSP.

B. End-to-End Feature-Aware Label Space Encoding

Before elaborating on the proposed E^{2}FE, to make it more clear, Table II summarizes the important symbols in this paper.

TABLE II
IMPORTANT SYMBOLS IN THE PROPOSED E^{2}FE

n	the number of training instances
d_x	the dimensionality of the feature space $\mathcal{X}$
d_y	the dimensionality of the label space $\mathcal{Y}$
d_z	the dimensionality of the latent space $\mathcal{Z}$, $d_z \ll d_y$
$\mathbf{X}$	the feature matrix of training instances, $\mathbf{X} \in \mathbb{R}^{n\times d_x}$
$\mathbf{Y}$	the tagging matrix of training instances, $\mathbf{Y} \in \{0,1\}^{n\times d_y}$
$\mathbf{Z}$	the code matrix of training instances, $\mathbf{Z} \in \mathbb{R}^{n\times d_z}$
$\mathbf{Q}$	the linear decoding matrix, $\mathbf{Q} \in \mathbb{R}^{d_z\times d_y}$
$\mathbf{H}$	notation for $\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$, $\mathbf{H} \in \mathbb{R}^{n\times n}$
$\mathbf{M}_{\cdot,i}$	the ith column of a matrix $\mathbf{M}$

As mentioned previously, E^{2}FE performs LSDR in an end-to-end manner and directly learns a code matrix $\mathbf{Z} \in \mathbb{R}^{n\times d_z}$ formed by code vectors of training instances. Generally, the classification performance of LSDR methods depends on both the predictive mapping $\mathcal{G}$ and the decoding process $\mathcal{Q}$. Therefore, it is crucial for code vectors to be predictable, having a strong correlation with instance features, as revealed in [43]. Meanwhile, the label vectors should also be highly recoverable via decoding the corresponding code vectors. Therefore, to learn $\mathbf{Z}$, E^{2}FE jointly maximizes the *recoverability* of the label space and the *predictability* of the latent space. The former is denoted as $\Psi_1(\mathbf{Y}, \mathbf{Z})$ and the latter as $\Psi_2(\mathbf{X}, \mathbf{Z})$, where $\mathbf{Y} \in \{0,1\}^{n\times d_y}$ is the tagging matrix of training instances formed by their label vectors row by row and $\mathbf{X} \in \mathbb{R}^{n\times d_x}$ is the feature matrix formed by their feature vectors in the same way. Then, the objective function with respect to $\mathbf{Z}$ is as follows:

$$\Psi = \max_{\mathbf{Z}} \ \Psi_1(\mathbf{Y}, \mathbf{Z}) + \alpha \Psi_2(\mathbf{X}, \mathbf{Z}) \tag{1}$$

where $\alpha \geq 0$ is a parameter for balancing *recoverability* and *predictability*. When $\alpha = 0$, $\mathbf{Z}$ will be derived via merely maximizing *recoverability*, implying that $\mathbf{Z}$ is just dependent on $\mathbf{Y}$. On the contrary, when $\alpha > 0$, correlations between instance features and code vectors will be further considered for making $\mathbf{Z}$ feature-aware and more predictable.

1) Recoverability of Label Space: To improve the *recoverability* of the label space, the difference between the tagging matrix $\mathbf{Y}$ and the recovered one, which is based on the to-be-learned code matrix $\mathbf{Z}$, is expected to be minimized. Here, we denote the difference as $\mathcal{L}$. As mentioned previously, for efficient decoding, the proposed E^{2}FE learns a linear decoding matrix $\mathbf{Q} \in \mathbb{R}^{d_z\times d_y}$ to recover label vectors from code vectors, following PLST, CPLST, MLC-BMad, and ML-CSSP. Then, $\mathcal{L}$ is formulated as follows:

$$\mathcal{L} = \min \|\mathbf{Y} - \mathbf{Z}\mathbf{Q}\|_{\text{fro}}^2 \tag{2}$$

where $\|\cdot\|_{\text{fro}}$ is the *Frobenius* norm of a matrix. Given $\mathbf{Z}$, the optimal $\mathbf{Q}$ to minimize $\mathcal{L}$ can be derived as the following closed-form expression by solving $(\partial\mathcal{L}/\partial\mathbf{Q}) = \mathbf{0}$:

$$\mathbf{Q} = (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{Y}. \tag{3}$$

To mitigate redundant information in the latent space and then encode the label space more compactly, we assume that the dimensions of the latent space are uncorrelated, and thus, the columns of $\mathbf{Z}$ are orthonormal, as shown in formula

$$\mathbf{Z}^T\mathbf{Z} = \mathbf{I} \tag{4}$$

where $\mathbf{I} \in \mathbb{R}^{d_z \times d_z}$ is an identity matrix. Actually, as analyzed later, although such an orthonormality assumption may seem to be strong, it is still reasonable and important for E^{2}FE. With formula (4), the optimal $\mathbf{Q}$ can be simplified as $\mathbf{Q} = \mathbf{Z}^T\mathbf{Y}$, and then, formula (2) can be reformulated as follows:

$$\mathcal{L} = \mathbf{Tr}[\mathbf{Y}^T\mathbf{Y} - \mathbf{Y}^T\mathbf{Z}\mathbf{Z}^T\mathbf{Y}] \tag{5}$$

where $\mathbf{Tr}[\cdot]$ refers to the *trace* of a matrix. With $\mathbf{Tr}[\mathbf{Y}^T\mathbf{Y}]$ being a constant, minimizing $\mathcal{L}$ is identical to maximizing $\mathbf{Tr}[\mathbf{Y}^T\mathbf{Z}\mathbf{Z}^T\mathbf{Y}]$, which can be seen as an expression of the *recoverability* of the label space, i.e., $\Psi_1(\mathbf{Y}, \mathbf{Z})$. We can thus derive the following formula:

$$\begin{aligned}\Psi_1(\mathbf{Y}, \mathbf{Z}) &= \mathbf{Tr}[\mathbf{Y}^T\mathbf{Z}\mathbf{Z}^T\mathbf{Y}] = \mathbf{Tr}[\mathbf{Z}^T\mathbf{Y}\mathbf{Y}^T\mathbf{Z}]\\ &\text{s.t. } \mathbf{Z}^T\mathbf{Z} = \mathbf{I}.\end{aligned} \tag{6}$$

2) Predictability of Latent Space: As advocated in [43], to improve the *predictability* of the latent space, the code matrix $\mathbf{Z}$ is supposed to be strongly correlated with the instance features. Here, we first consider linear correlations, and will later handle nonlinear ones with kernel tricks. Considering a linear projection $\mathbf{w}$ for the feature space and a dimension $\mathbf{z}$ of the latent space, i.e., a column of $\mathbf{Z}$, the correlation between features and $\mathbf{z}$, denoted as $r(\mathbf{X}, \mathbf{z})$, can be defined as follows:

$$r(\mathbf{X}, \mathbf{z}) = \frac{(\mathbf{X}\mathbf{w})^T\mathbf{z}}{\sqrt{(\mathbf{X}\mathbf{w})^T(\mathbf{X}\mathbf{w})}\sqrt{\mathbf{z}^T\mathbf{z}}}. \tag{7}$$

Due to the orthonormality assumption for $\mathbf{Z}$, i.e., formula (4), $\mathbf{z}^T\mathbf{z} = 1$ will hold for any column of $\mathbf{Z}$. Moreover, linearly rescaling $\mathbf{w}$ by a nonzero multiplier will not change $r(\mathbf{X}, \mathbf{z})$. Then, maximizing $r(\mathbf{X}, \mathbf{z})$ equals the following formula:

$$\max\ (\mathbf{X}\mathbf{w})^T\mathbf{z} \quad \text{s.t. } (\mathbf{X}\mathbf{w})^T\mathbf{X}\mathbf{w} = 1. \tag{8}$$

Given a dimension $\mathbf{z}$ of the latent space, the maximal $r(\mathbf{X}, \mathbf{z})$ reflects its potential maximal correlation with the feature space, and thus, the maximal $r(\mathbf{X}, \mathbf{z})$ can be seen as an expression of the *predictability* of $\mathbf{z}$. Specifically, with $\mathbf{z}$ fixed, the optimal $\mathbf{w}$ for formula (8), denoted as $\mathbf{w}^*$, can be derived as follows with the method of Lagrange multipliers:

$$\mathbf{w}^* = \frac{(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{z}}{\sqrt{\mathbf{z}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{z}}}. \tag{9}$$

Note that following CPLST, here we assume $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ to be invertible. Actually, this assumption usually holds when $n > d_x$, but it will fail in cases with $n < d_x$, as $\mathbf{A}$ will not be full-rank then. To handle the latter cases, we propose to ensure $\mathbf{A}$ to be invertible via: 1) performing dimension reduction for the feature space via PCA or alternative methods to make d_x small enough for obtaining a full-rank $\mathbf{X}^T\mathbf{X}$ or 2) adding a tiny value to the entries on the diagonal of $\mathbf{X}^T\mathbf{X}$, i.e., $\mathbf{A} = \mathbf{X}^T\mathbf{X} + \varepsilon\mathbf{I}_1$ with $\mathbf{I}_1 \in \mathbb{R}^{d_x \times d_x}$ being an identity matrix and ε being a tiny value, e.g., 10^{-6}.

By substituting $\mathbf{w}^*$ into formula (7), the *predictability* of $\mathbf{z}$, denoted as $\psi_2(\mathbf{X}, \mathbf{z})$, can be derived as follows:

$$\psi_2(\mathbf{X}, \mathbf{z}) = \frac{(\mathbf{X}\mathbf{w}^*)^T\mathbf{z}}{\sqrt{(\mathbf{X}\mathbf{w}^*)^T(\mathbf{X}\mathbf{w}^*)}\sqrt{\mathbf{z}^T\mathbf{z}}} = (\mathbf{X}\mathbf{w}^*)^T\mathbf{z} = \sqrt{\mathbf{z}^T\mathbf{H}\mathbf{z}} \tag{10}$$

Algorithm 1 Overview of E^{2}FE

```
Input: Feature matrix $\mathbf{X}_{tr}$ and tagging matrix $\mathbf{Y}_{tr}$ of the
    training instances, feature matrix $\mathbf{X}_{ts}$ of the test instances,
    predefined model parameter $\alpha$, and latent space dimensionality $d_z$
Output: Predicted binary tagging matrix $\mathbf{Y}_{ts}$ of the test
    instances
    Training Process:
1: derive code matrix $\mathbf{Z}_{tr}$ via optimizing formula (12)
2: learn predictive models: $\mathcal{G}(\mathbf{X}_{tr}) \rightarrow \mathbf{Z}_{tr}$
3: derive linear decoding matrix: $\mathbf{Q} = \mathbf{Z}_{tr}^T\mathbf{Y}_{tr}$
    Predicting Process:
4: predict code vectors of test instances: $\mathbf{Z}_{ts} = \mathcal{G}(\mathbf{X}_{ts})$
5: recover the predicted tagging matrix: $\mathbf{Y}_{ts} = round(\mathbf{Z}_{ts}\mathbf{Q})$
```

where $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T \in \mathbb{R}^{n \times n}$. To improve the *predictability* of the latent space, each column $\mathbf{z}$ of the code matrix $\mathbf{Z}$ is supposed to maximize $\psi_2(\mathbf{X}, \mathbf{z})$. As maximizing $\psi_2(\mathbf{X}, \mathbf{z})$ can be guaranteed by maximizing $\mathbf{z}^T\mathbf{H}\mathbf{z}$, the overall *predictability* of $\mathbf{Z}$ can be formulated as follows:

$$\begin{aligned}\Psi_2(\mathbf{X}, \mathbf{Z}) &= \sum_{i=1}^{d_z}\mathbf{Z}_{\cdot,i}^T\mathbf{H}\mathbf{Z}_{\cdot,i} = \mathbf{Tr}[\mathbf{Z}^T\mathbf{H}\mathbf{Z}]\\ &\text{s.t. } \mathbf{Z}^T\mathbf{Z} = \mathbf{I}\end{aligned} \tag{11}$$

where $\mathbf{Z}_{\cdot,i}\ (i \in \{1, 2, \ldots, d_z\})$ denotes the ith column of $\mathbf{Z}$.

3) Detailed Objective Function: With $\Psi_1(\mathbf{Z}, \mathbf{Y})$ and $\Psi_2(\mathbf{X}, \mathbf{Z})$ derived, the objective function with respect to the to-be-learned code matrix $\mathbf{Z}$, i.e., formula (1), can be detailed as follows:

$$\begin{aligned}\Psi &= \max_{\mathbf{Z}}\ \mathbf{Tr}[\mathbf{Z}^T\mathbf{Y}\mathbf{Y}^T\mathbf{Z}] + \alpha\mathbf{Tr}[\mathbf{Z}^T\mathbf{H}\mathbf{Z}]\\ &= \max_{\mathbf{Z}}\ \mathbf{Tr}[\mathbf{Z}^T(\mathbf{Y}\mathbf{Y}^T + \alpha\mathbf{H})\mathbf{Z}]\\ &\quad \text{s.t. } \mathbf{Z}^T\mathbf{Z} = \mathbf{I}\end{aligned} \tag{12}$$

where $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$. As analysed in section IV, Ψ can be transformed to an eigenvalue problem with respect to $\mathbf{Y}\mathbf{Y}^T + \alpha\mathbf{H}$, and $\mathbf{Z}$ is derived by concatenating the normalized eigenvectors corresponding to the top d_z largest eigenvalues column by column. With the code matrix $\mathbf{Z}$ derived, predictive models can be trained for mapping instance features into code vectors.

4) Deriving Linear Decoding Matrix: According to formula (2) and (3), given $\mathbf{Z}$ with $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}$, the optimal linear decoding matrix $\mathbf{Q}$ can be derived as follows:

$$\mathbf{Q} = (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{Y} = \mathbf{Z}^T\mathbf{Y}. \tag{13}$$

And its computational complexity is $\mathcal{O}(nd_yd_z)$.

An overview of E^{2}FE is given in Algorithm 1.

C. Error Analysis

As shown in Algorithm 1, following PLST, CPLST, and ML-CSSP, the proposed E^{2}FE *rounds* each entry of the decoding results into its nearest 0 and 1, so as to derive binary label vectors. Considering that, we proceed to analyze the root mean square error (RMSE) of E^{2}FE on the training instances.

Specifically, RMSE is defined as follows:

$$\text{RMSE} = \frac{1}{\sqrt{n}} \|\text{round}(\mathcal{G}(\mathbf{X})\mathbf{Q}) - \mathbf{Y}\|_{\text{fro}} \tag{14}$$

where $\mathcal{G}$ denotes the learned predictive models for mapping instance features into code vectors, and $round(\mathcal{G}(\mathbf{X})\mathbf{Q})$ denotes the recovered binary tagging matrix. Then, we can derive the following lemma regarding the error bound of E^{2}FE.

Lemma 1: For E^{2}FE, its RMSE is bounded by

$$\text{RMSE} \leq \frac{2}{\sqrt{n}}(\sqrt{d_z}\|\mathbf{Y}\|_{\text{fro}}\|\mathbf{Z} - \mathcal{G}(\mathbf{X})\|_{\text{fro}} + \|\mathbf{Y} - \mathbf{ZQ}\|_{\text{fro}}).$$

For a detailed proof, one can refer to the supplementary material. Actually, the error bound for E^{2}FE is similar to those of PLST and ML-CSSP. Namely, it also consists of two parts. The first part, i.e., $\sqrt{d_z}\|\mathbf{Y}\|_{\text{fro}}\|\mathbf{Z}-\mathcal{G}(\mathbf{X})\|_{\text{fro}}$ denotes the weighted training error of predictive models, and the second part, i.e., $\|\mathbf{Y} - \mathbf{ZQ}\|_{\text{fro}}$, denotes the loss of encoding label vectors into low-dimensional code vectors.

IV. Optimization Methods

For optimizing the objective function Ψ with respect to the to-be-learned code matrix $\mathbf{Z}$, any column $\mathbf{Z}_{\cdot,i} (i \in \{1, 2, \ldots, d_z\})$ can be derived with the following optimization subproblem:

$$\Psi^{(i)} = \max_{\mathbf{Z}_{\cdot,i}} \mathbf{Z}_{\cdot,i}^T(\mathbf{YY}^T + \alpha\mathbf{H})\mathbf{Z}_{\cdot,i}$$
$$\text{s.t. } \mathbf{Z}_{\cdot,i}^T\mathbf{Z}_{\cdot,i} = 1,\ \mathbf{Z}_{\cdot,j}^T\mathbf{Z}_{\cdot,i} = 0 \quad (\forall j < i). \tag{15}$$

With the method of Lagrange multipliers, the optimal $\mathbf{Z}_{\cdot,i}$ should satisfy the following optimality condition:

$$(\mathbf{YY}^T + \alpha\mathbf{H})\mathbf{Z}_{\cdot,i} = \lambda_i\mathbf{Z}_{\cdot,i} \tag{16}$$

where λ_i is the introduced Lagrange multiplier and will also be the optimal value of the subproblem. It can be seen that the optimization for $\mathbf{Z}$ can be transformed to an eigenvalue problem. Then, by normalizing the eigenvectors of $\mathbf{U} = \mathbf{YY}^T + \alpha\mathbf{H}$ that correspond to the top d_z largest eigenvalues, we can derive the optimal code matrix $\mathbf{Z}$ formed of these eigenvectors column by column, which satisfies $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}$.

As described in our previous work [36], we can directly calculate $\mathbf{U}$ and then utilize effective methods to derive its eigenvectors. However, considering that $\mathbf{U} \in \mathbb{R}^{n\times n}$, for cases with $n \gg d_x+d_y$, which are common in practical applications, calculating $\mathbf{U}$ will result in high space costs. To avoid that, we derive the following lemma and further propose a more efficient optimization method for such cases.

Lemma 2: Given $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$, the matrix $\mathbf{U} = \mathbf{YY}^T + \alpha\mathbf{H}$ can be decomposed as $\mathbf{U} = \mathbf{VV}^T$ with $\mathbf{V} \in \mathbb{R}^{n\times(d_y+d_x)}$. Also, the eigenvectors of $\mathbf{U}$ can be derived from those of $\mathbf{V}^T\mathbf{V}$, meaning that the size of the eigenvalue problem with respect to $\mathbf{U}$ can be transformed from $\mathbb{R}^{n\times n}$ to $\mathbb{R}^{(d_x+d_y)\times(d_x+d_y)}$.

Proof: Suppose $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ is invertible. Since $\mathbf{A} \in \mathbb{R}^{d_x\times d_x}$ is a real symmetric and positive-semidefinite matrix, $\mathbf{A}^{-1}$ will be real symmetric and positive semi-definite, and thus, $\mathbf{A}^{-1}$ is diagonalizable by orthogonal matrices [44]. Namely, $\mathbf{A}^{-1} = \mathbf{B}\Lambda\mathbf{B}^T$, with Λ being a diagonal matrix having nonnegative diagonal entries and $\mathbf{B}$ being an orthonormal matrix. Then, $\mathbf{A}^{-1} = \mathbf{B}\Lambda^{(1/2)}\Lambda^{(1/2)}\mathbf{B}^T = (\mathbf{B}\Lambda^{(1/2)})(\mathbf{B}\Lambda^{(1/2)})^T$, where $\Lambda^{(1/2)}$ is a diagonal matrix with each diagonal entry being the square root of the corresponding diagonal entry in Λ. Furthermore, with $\mathbf{G} = \mathbf{XB}\Lambda^{(1/2)} \in \mathbb{R}^{n\times d_x}$, $\mathbf{H} = \mathbf{GG}^T$. Finally, $\mathbf{U} = \mathbf{YY}^T + \alpha\mathbf{H} = \mathbf{YY}^T + (\sqrt{\alpha}\mathbf{G})(\sqrt{\alpha}\mathbf{G})^T = [\mathbf{Y}, \sqrt{\alpha}\mathbf{G}][\mathbf{Y}, \sqrt{\alpha}\mathbf{G}]^T = \mathbf{VV}^T$, with $\mathbf{V} = [\mathbf{Y}, \sqrt{\alpha}\mathbf{G}] \in \mathbb{R}^{n\times(d_y+d_x)}$.

Suppose $\{\lambda, \mathbf{p}\}$ and $\{\sigma, \mathbf{q}\}$ are, respectively, the paired eigenvalue/eigenvector of $\mathbf{VV}^T$ and $\mathbf{V}^T\mathbf{V}$. According to: 1) $\mathbf{VV}^T\mathbf{p} = \lambda\mathbf{p} \to (\mathbf{V}^T\mathbf{V})\mathbf{V}^T\mathbf{p} = \mathbf{V}^T(\mathbf{VV}^T\mathbf{p}) = \lambda\mathbf{V}^T\mathbf{p}$ and 2) $\mathbf{V}^T\mathbf{Vq} = \sigma\mathbf{q} \to (\mathbf{VV}^T)\mathbf{Vq} = \mathbf{V}(\mathbf{V}^T\mathbf{Vq}) = \sigma\mathbf{Vq}$, we can see that $\mathbf{VV}^T$ and $\mathbf{V}^T\mathbf{V}$ share identical eigenvalues, and the eigenvectors of $\mathbf{U} = \mathbf{VV}^T$ can be derived from those of $\mathbf{V}^T\mathbf{V}$ based on the second derivation above. Considering $\mathbf{V}^T\mathbf{V} \in \mathbb{R}^{(d_x+d_y)\times(d_x+d_y)}$, the size of the eigenvalue problem with respect to $\mathbf{U}$ can be transformed from $\mathbb{R}^{n\times n}$ to $\mathbb{R}^{(d_x+d_y)\times(d_x+d_y)}$. □

```
Algorithm 2 Optimization for E$^2$FE
Input: Feature matrix $\mathbf{X}_{tr} \in \mathbb{R}^{n\times d_x}$ and tagging matrix $\mathbf{Y}_{tr} \in \mathbb{R}^{n\times d_y}$ of training instances, predefined model parameter $\alpha$, latent space dimensionality $d_z$
Output: Learned code matrix $\mathbf{Z}_{tr}$ of training instances
 1: if $\mathbf{X}_{tr}^T\mathbf{X}_{tr}$ is NOT invertible then
 2:     Option 1: {dimension reduction for feature space}
 3:         $\mathbf{X}_{tr} = DimReduce(\mathbf{X}_{tr})$
 4:     Option 2: {adding a tiny value to diagonal entries}
 5:         $\mathbf{X}_{tr}^T\mathbf{X}_{tr} = \mathbf{X}_{tr}^T\mathbf{X}_{tr} + \varepsilon\mathbf{I}_1$
 6: end if
 7: if $n \gg d_y + d_x$ then
 8:     $\mathbf{A} = \mathbf{X}_{tr}^T\mathbf{X}_{tr}$
 9:     $[\mathbf{B}, \Lambda] = diagonalize(\mathbf{A}^{-1})$ {$\mathbf{A}^{-1} = \mathbf{B}\Lambda\mathbf{B}^T$}
10:     $\mathbf{G} = \mathbf{X}_{tr}\mathbf{B}\Lambda^{\frac{1}{2}}$
11:     $\mathbf{V} = [\mathbf{Y}_{tr}, \sqrt{\alpha}\mathbf{G}]$
12:     $\mathbf{E} = eigenvector(\mathbf{V}^T\mathbf{V}, d_z)$ {eigenvectors of $\mathbf{V}^T\mathbf{V}$ corresponding to the top $d_z$ largest eigenvalues}
13:     $\mathbf{Z}_{tr} = normalize(\mathbf{VE})$ {normalizing each column of $\mathbf{VE}$ into a unit vector}
14: else
15:     $\mathbf{H} = \mathbf{X}_{tr}(\mathbf{X}_{tr}^T\mathbf{X}_{tr})^{-1}\mathbf{X}_{tr}^T$
16:     $\mathbf{U} = \mathbf{Y}_{tr}\mathbf{Y}_{tr}^T + \alpha\mathbf{H}$
17:     $\tilde{\mathbf{E}} = eigenvector(\mathbf{U}, d_z)$ {eigenvectors of $\mathbf{U}$ corresponding to the top $d_z$ largest eigenvalues}
18:     $\mathbf{Z}_{tr} = normalize(\tilde{\mathbf{E}})$ {normalizing each column of $\tilde{\mathbf{E}}$ into a unit vector}
19: end if
```

With Lemma 2, in different cases, we can utilize different optimization methods to obtain the eigenvectors of $\mathbf{U} = \mathbf{YY}^T + \alpha\mathbf{H}$ and then derive the code matrix $\mathbf{Z}$, as summarized in the following and illustrated in Algorithm 2.

1) If $n \gg d_y + d_x$, it is preferable to firstly derive the matrix $\mathbf{V}$ satisfying $\mathbf{U} = \mathbf{VV}^T$, then calculate the eigenvectors of $\mathbf{V}^T\mathbf{V}$ corresponding to the top d_z

largest eigenvalues, and finally utilize them to derive the eigenvectors of $\mathbf{U}$. Since $d_z \ll d_y$ and $\mathbf{V}^T\mathbf{V}$ is a real symmetric matrix, the eigenvalue problem with respect to $\mathbf{V}^T\mathbf{V}$ can be solved efficiently using iterative methods such as Arnoldi iteration [45], which can achieve an optimal computational complexity of $\mathcal{O}(d_x {d_z}^2 + d_y {d_z}^2)$. Here, the computational complexity of deriving $\mathbf{V}$ is $\mathcal{O}(n d_x^2)$, while that of calculating $\mathbf{V}^T\mathbf{V}$ and deriving the eigenvectors of $\mathbf{U}$ from those of $\mathbf{V}^T\mathbf{V}$ is $\mathcal{O}(n(d_x + d_y)^2)$.

2) Otherwise, it is preferable to directly calculate $\mathbf{U}$ and then perform an eigenvalue decomposition on it. The computational complexity for calculating $\mathbf{U}$ is at most $\mathcal{O}(\min\{n^2 d_x, n d_x^2\}) + \mathcal{O}(n^2 d_x + n^2 d_y)$. Considering that generally $d_z \ll n$ and $\mathbf{U}$ is a real symmetric matrix, the eigenvalue problem with respect to $\mathbf{U}$ can also be solved efficiently using Arnoldi iteration with an optimal computational complexity of $\mathcal{O}(n {d_z}^2)$.

V. πE^{2}FE: Enhancing Linear Decoding Matrix With *a Priori* Knowledge

As analyzed in formula (16), each column of the code matrix $\mathbf{Z}$ corresponds to an eigenvalue of $\mathbf{U} = \mathbf{Y}\mathbf{Y}^T + \alpha\mathbf{H}$, which is also the optimal value for its corresponding optimization subproblem (i.e., formula (15)). Knowing that each column denotes one dimension of the latent space, for each column, the eigenvalue with respect to it actually reflects: 1) how predictable its corresponding dimension of the latent space is and 2) from the dimension how recoverable the label space is. Specifically, a higher eigenvalue with respect to a column of $\mathbf{Z}$ means that its corresponding dimension of the latent space is more predictable and the label space is more recoverable from the dimension.

Here, we propose to consider such *a priori* knowledge to derive an enhanced linear decoding matrix for E^{2}FE. We denote it as πE^{2}FE. Essentially, for a linear decoding matrix $\mathbf{Q}$, its ith column $\mathbf{Q}_{\cdot,i} (i \in \{1, 2, \ldots, d_y\})$ acts as a weighting vector to linearly combine dimensions of the latent space for recovering the ith dimension of the label space. Then, for dimensions of the latent space that are more predictable and make the label space more recoverable, i.e., with higher corresponding eigenvalues, they are expected to be assigned with higher weights in the decoding process. Therefore, we derive the objective function for $\mathbf{Q}_{\cdot,i}$ as follows:

$$\tilde{\mathcal{L}}^{(i)} = \min_{\mathbf{Q}_{\cdot,i}} \|\mathbf{Y}_{\cdot,i} - \mathbf{Z}\mathbf{Q}_{\cdot,i}\|_{\text{fro}}^2 - \eta \sum_{j=1}^{d_z} \lambda_j \mathbf{Q}_{j,i}^2 \tag{17}$$

where $\mathbf{Y}_{\cdot,i}$ is the ith column of the tagging matrix $\mathbf{Y}$, λ_j is the eigenvalue corresponding to the jth column of $\mathbf{Z}$, and η is a nonnegative weighting factor. It can be seen that, by considering the a priori knowledge as a regularizer in $\tilde{\mathcal{L}}^{(i)}$, a larger λ_j can help to lead $\mathbf{Q}_{j,i}^2$ to be larger, meaning that as expected the jth dimension of the latent space is assigned with a higher weight for decoding. For model simplicity, here η is shared by all $\mathbf{Q}_{\cdot,i} (i \in \{1, 2, \ldots, d_y\})$. Then, the objective function for deriving the linear decoding matrix $\mathbf{Q}$ of πE^{2}FE can be formulated as follows with matrix notations:

$$\tilde{\mathcal{L}} = \min_{\mathbf{Q}} \|\mathbf{Y} - \mathbf{Z}\mathbf{Q}\|_{\text{fro}}^2 - \eta \mathbf{Tr}[\mathbf{Q}^T \tilde{\mathbf{\Lambda}} \mathbf{Q}] \tag{18}$$

where $\tilde{\mathbf{\Lambda}}$ is a diagonal matrix with $\tilde{\mathbf{\Lambda}}_{j,j} = \lambda_j$. If η is properly set to make $\tilde{\mathcal{L}}$ nontrivial, as discussed later, the optimal decoding matrix for πE^{2}FE can be derived as follows:

$$\mathbf{Q} = (\mathbf{I} - \eta\tilde{\mathbf{\Lambda}})^{-1} \mathbf{Z}^T \mathbf{Y} \tag{19}$$

where $(\mathbf{I} - \eta\tilde{\mathbf{\Lambda}})$ is a diagonal matrix, and thus, its inverse can be efficiently calculated. Actually, as $d_z \ll d_y$, the computational complexity of deriving $\mathbf{Q}$ in πE^{2}FE is also $\mathcal{O}(n d_y d_z)$.

Note that in formula (18), a large η value can lead $\tilde{\mathcal{L}}$ to become trivial and achieve an optimum of negative infinity. To cope with that, we derive the following lemma for properly setting η, where $\vec{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_{d_z}]$.

Lemma 3: For any $\eta \in [0, (1/\max(\vec{\lambda}))]$ with $\max(\vec{\lambda})$ being the maximal value of $\vec{\lambda}$, $\tilde{\mathcal{L}}$ will be nontrivial for optimization. For a detailed proof, one can refer to the supplementary material.

VI. Extensions and Analyses

A. Function-Based Encoding: A Linear Encoding Case

Though the proposed E^{2}FE requires no encoding function, it can still be specified to learn an encoding function as most previous works, given that the encoding function can be optimized, e.g., a linear one, as described in the following.

Following PLST and CPLST, we use an encoding matrix $\mathbf{P} \in \mathbb{R}^{d_y \times d_z}$ to denote the linear encoding function. Then, the code matrix $\mathbf{Z}$ can be expressed as $\mathbf{Z} = \mathbf{Y}\mathbf{P}$. Substituting $\mathbf{Z}$ with $\mathbf{Y}\mathbf{P}$ in the objective function of E^{2}FE, i.e., formula (12), we can derive the following objective function for $\mathbf{P}$:

$$\begin{aligned}\Psi = \max_{\mathbf{P}} \ & \mathbf{Tr}[\mathbf{P}^T (\mathbf{Y}^T\mathbf{Y}\mathbf{Y}^T\mathbf{Y} + \alpha\mathbf{Y}^T\mathbf{H}\mathbf{Y})\mathbf{P}] \\ \text{s.t. } & \mathbf{P}^T\mathbf{Y}^T\mathbf{Y}\mathbf{P} = \mathbf{I}.\end{aligned} \tag{20}$$

Similarly, we use the method of Lagrange multipliers and decompose Ψ into d_z optimization subproblems with respect to each column $\mathbf{P}_{\cdot,i}$ of the to-be-learned $\mathbf{P}$. Then, we derive that $\mathbf{P}_{\cdot,i}$ should satisfy the following optimality condition:

$$(\mathbf{Y}^T\mathbf{Y}\mathbf{Y}^T\mathbf{Y} + \alpha\mathbf{Y}^T\mathbf{H}\mathbf{Y})\mathbf{P}_{\cdot,i} = \lambda_i(\mathbf{Y}^T\mathbf{Y})\mathbf{P}_{\cdot,i} \tag{21}$$

where λ_i is a Lagrange multiplier and will be the optimal value of the optimization subproblem with respect to $\mathbf{P}_{\cdot,i}$. It can be seen that the optimization of $\mathbf{P}$ is essentially a general eigenvalue problem. And the normalized eigenvectors corresponding to the top d_z largest eigenvalues will form the optimal $\mathbf{P}$.

Denoting this case of linear function-based encoding as LinearE^{2}FE, the linear decoding matrix $\mathbf{Q}$ without considering a priori knowledge is $\mathbf{Q} = (\mathbf{Y}\mathbf{P})^T\mathbf{Y}$. Meanwhile, for the case of utilizing the eigenvalues with respect to $\mathbf{P}$ as *a priori* knowledge, $\mathbf{Q} = (\mathbf{I} - \eta\tilde{\mathbf{\Lambda}})^{-1}(\mathbf{Y}\mathbf{P})^T\mathbf{Y}$ with $\tilde{\mathbf{\Lambda}}$ being a diagonal matrix consisting of the eigenvalues, which is termed πLinearE^{2}FE.

B. Kernel Version

The proposed E^{2}FE, thanks to kernel tricks, can be extended to deal with nonlinear correlations between the feature space and the latent space, which is termed *kernel*-E^{2}FE.

In *kernel*-E^{2}FE, each feature vector $\mathbf{x}^{(i)}$ is mapped to the reproducing kernel Hilbert space (RKHS) as $\phi(\mathbf{x}^{(i)})$. In RKHS, the inner product between $\phi(\mathbf{x}^{(i)})$ and $\phi(\mathbf{x}^{(j)})$ is equal to $\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$, where $\kappa(\cdot,\cdot)$ is the introduced kernel function. Using a nonlinear $\kappa(\cdot,\cdot)$, the linear correlations between the RKHS and the latent space actually reflect the nonlinear correlations between the original feature space and the latent space. Similar to formula (7), we measure the correlation $r(\mathbf{\Phi}, \mathbf{z})$ by considering a linear projection $\mathbf{w}_1$ for kernel features in RKHS and a column $\mathbf{z}$ of the code matrix. Following [46], here $\mathbf{w}_1$ is assumed to be in the span of sampled kernel feature vectors, i.e., $\mathbf{w}_1 = \mathbf{\Phi}_*^T \tilde{\mathbf{w}}$, where $\mathbf{\Phi}_*$ is a matrix built by the sampled kernel feature vectors row by row and $\tilde{\mathbf{w}}$ is an s-dimensional weighting vector with s being the sampling size. Then, $r(\mathbf{\Phi}, \mathbf{z})$ can be measured as follows:

$$r(\mathbf{\Phi}, \mathbf{z}) = \frac{\left(\mathbf{\Phi}\mathbf{\Phi}_*^T \tilde{\mathbf{w}}\right)^T \mathbf{z}}{\sqrt{\left(\mathbf{\Phi}\mathbf{\Phi}_*^T \tilde{\mathbf{w}}\right)^T \left(\mathbf{\Phi}\mathbf{\Phi}_*^T \tilde{\mathbf{w}}\right)}\sqrt{\mathbf{z}^T\mathbf{z}}} = \frac{(\mathbf{K}\tilde{\mathbf{w}})^T \mathbf{z}}{\sqrt{(\mathbf{K}\tilde{\mathbf{w}})^T(\mathbf{K}\tilde{\mathbf{w}})}\sqrt{\mathbf{z}^T\mathbf{z}}} \tag{22}$$

where $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}_*^T \in \mathbb{R}^{n\times s}$ is a kernel matrix and can be efficiently derived using the kernel function with the original feature vectors. Similar to Section III-B.2, the *predictability* of $\mathbf{z}$ based on nonlinear correlations can be derived from the maximal $r(\mathbf{\Phi}, \mathbf{z})$ and measured as $\psi_2(\mathbf{\Phi}, \mathbf{z}) = \sqrt{\mathbf{z}^T\tilde{\mathbf{H}}\mathbf{z}}$ with $\tilde{\mathbf{H}} = \mathbf{K}(\mathbf{K}^T\mathbf{K})^{-1}\mathbf{K}^T$. Then, the objective function of *kernel*-E^{2}FE is as follows, which can also be transformed to an eigenvalue problem:

$$\Psi = \max_{\mathbf{Z}} \ \mathbf{Tr}[\mathbf{Z}^T(\mathbf{Y}\mathbf{Y}^T + \alpha\tilde{\mathbf{H}})\mathbf{Z}] \quad \text{s.t. } \mathbf{Z}^T\mathbf{Z} = \mathbf{I}. \tag{23}$$

Like E^{2}FE, the linear decoding matrix $\mathbf{Q}$ for *kernel*-E^{2}FE is $\mathbf{Q} = \mathbf{Z}^T\mathbf{Y}$. Meanwhile, when eigenvalues with respect to $\mathbf{Z}$ are considered as *a priori* knowledge, $\mathbf{Q} = \left(\mathbf{I} - \eta\tilde{\mathbf{\Lambda}}\right)^{-1}\mathbf{Z}^T\mathbf{Y}$, where $\tilde{\mathbf{\Lambda}}$ is a diagonal matrix consisting of eigenvalues. Here, we denote this case as *kernel*-πE^{2}FE.

C. Relations to Previous Works

If the mean values of the label vectors are shifted as zeros, the proposed E^{2}FE will degenerate to PLST [31] when only the *recoverability* of the label space is considered (i.e., $\alpha = 0$ in formula (12)). Here, we denote this case as R-E^{2}FE and its corresponding objective function is given as follows:

$$\Psi = \max_{\mathbf{Z}} \ \mathbf{Tr}[\mathbf{Z}^T\mathbf{Y}\mathbf{Y}^T\mathbf{Z}], \quad \text{s.t. } \mathbf{Z}^T\mathbf{Z} = \mathbf{I}. \tag{24}$$

The code matrix $\mathbf{Z}$ consists of the normalized eigenvectors of $\mathbf{Y}\mathbf{Y}^T$ corresponding to the top d_z largest eigenvalues, and the linear decoding matrix without considering *a priori* knowledge is $\mathbf{Z}^T\mathbf{Y}$. Meanwhile, the linear encoding matrix $\mathbf{P}$ of PLST is formed with the normalized eigenvectors of $\mathbf{Y}^T\mathbf{Y}$ corresponding to the top d_z largest eigenvalues, with the derived code matrix being $\mathbf{YP}$ and the linear decoding matrix being $\mathbf{P}^T$. As in the proof of Lemma 2, we can derive that $\mathbf{Y}\mathbf{Y}^T$ and $\mathbf{Y}^T\mathbf{Y}$ are positive semidefinite and share the same positive eigenvalues. Specifically, provided that λ_i is the ith largest eigenvalue, we can derive that: 1) $\mathbf{Y}^T\mathbf{Y}\mathbf{P}_{\cdot,i} = \lambda_i\mathbf{P}_{\cdot,i}$; 2) $\mathbf{Y}\mathbf{Y}^T\mathbf{Z}_{\cdot,i} = \lambda_i\mathbf{Z}_{\cdot,i}$; 3) $(\mathbf{Y}\mathbf{Y}^T)[\mathbf{YP}]_{\cdot,i} = \mathbf{Y}(\mathbf{Y}^T\mathbf{Y}\mathbf{P}_{\cdot,i}) = \lambda_i[\mathbf{YP}]_{\cdot,i}$; and 4) $(\mathbf{Y}^T\mathbf{Y})[\mathbf{Y}^T\mathbf{Z}]_{\cdot,i} = \mathbf{Y}^T(\mathbf{Y}\mathbf{Y}^T\mathbf{Z}_{\cdot,i}) = \lambda_i[\mathbf{Y}^T\mathbf{Z}]_{\cdot,i}$. Then, for R-E^{2}FE and PLST, we can find one-to-one correspondences between the ith columns of their encoding results (i.e., $\mathbf{Z}_{\cdot,i} = ([\mathbf{YP}]_{\cdot,i}/\sqrt{\lambda_i})$), and between the ith rows of their linear decoding matrices (i.e., $[\mathbf{Z}^T\mathbf{Y}]_{i,\cdot} = \sqrt{\lambda_i}[\mathbf{P}^T]_{i,\cdot}$). Therefore, R-E^{2}FE is equivalent to PLST, with $\mathbf{Z}(\mathbf{Z}^T\mathbf{Y}) = (\mathbf{YP})\mathbf{P}^T$. However, when $\alpha > 0$, the code matrix $\mathbf{Z}$ in E^{2}FE will be associated with instance features and will then differ from PLST.

When coping with linear function-based encoding, i.e., formula (20), given the mean values of label vectors and those of feature vectors shifted as zeros, E^{2}FE is closely connected to CPLST [33] if only the *predictability* of the latent space is considered. We denote this case as P-LinearE^{2}FE, with its corresponding objective function defined as follows:

$$\Psi = \max_{\mathbf{P}} \ \mathbf{Tr}[\mathbf{P}^T\mathbf{Y}^T\mathbf{H}\mathbf{Y}\mathbf{P}], \quad \text{s.t. } \mathbf{P}^T\mathbf{Y}^T\mathbf{Y}\mathbf{P} = \mathbf{I}. \tag{25}$$

Meanwhile, the objective function of CPLST is as follows:

$$\tilde{\Psi} = \max_{\mathbf{P}} \ \mathbf{Tr}[\mathbf{P}^T\mathbf{Y}^T\mathbf{H}\mathbf{Y}\mathbf{P}], \quad \text{s.t. } \mathbf{P}^T\mathbf{P} = \mathbf{I}. \tag{26}$$

It can be seen that P-LinearE^{2}FE and CPLST share an identical objective function but with different constraints. Namely, the former requires the dimensions of the code matrix (i.e., $\mathbf{YP}$) to be orthonormal, while the latter requires the dimensions of the linear encoding matrix (i.e., $\mathbf{P}$) to be orthonormal.

Another useful observation with respect to E^{2}FE is that E^{2}FE actually performs dimension reduction for both the label space and the feature space when the *predictability* of the latent space is overemphasized with an assumption that the code matrix can be directly expressed by the feature matrix, i.e., $\mathbf{Z} = \mathbf{XW}$, where $\mathbf{W} \in \mathbb{R}^{d_x\times d_z}$ is a regression matrix. This case is termed OP-E^{2}FE. As the *predictability* of the latent space is constant in OP-E^{2}FE, its objective function is formulated as follows:

$$\Psi = \max_{\mathbf{W}} \ \mathbf{Tr}[\mathbf{W}^T\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X}\mathbf{W}] \quad \text{s.t. } \mathbf{W}^T\mathbf{X}^T\mathbf{X}\mathbf{W} = \mathbf{I}. \tag{27}$$

The optimization for $\mathbf{W}$ can again be interpreted as a general eigenvalue problem, i.e., $(\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X})\mathbf{W}_{\cdot,i} = \lambda_i(\mathbf{X}^T\mathbf{X})\mathbf{W}_{\cdot,i}$, but it requires $d_z \le d_x$. Here, $\mathbf{Z}$ can be seen as the dimension reduction result learned from the label space in an end-to-end manner, or the linear dimension reduction result from the feature space with $\mathbf{W}$. However, for OP-E^{2}FE, we can observe the following weak points: 1) the dimensionality of the to-be-learned latent space cannot be larger than the dimensionality of the feature space, which can sometimes be too small to keep enough information of the label space, especially when $d_x \ll d_y$ and 2) the predictive models from the feature space to the latent space are limited to be linear regression, whereas for LSDR, they are expected to be open for any effective model.

TABLE III
STATISTICS OF DATA SETS

	domain	instances	labels	features
delicious	text	16,105	983	500
CAL500	music	502	174	68
mediamill	video	43,907	101	120
ESPGame	image	65,065	1,718	516
bibtex	text	7,395	159	1,836
bookmarks	text	87,856	208	2,150

VII. EXPERIMENTS

A. Experimental Settings

To validate the proposed E^{2}FE, we use in our experiments five widely used benchmark data sets with relatively large vocabularies from Mulan [47], i.e., *delicious*, *CAL500*, *mediamill*, *bibtex*, and *bookmarks*. They belong to one of the following domains: text, music, and video. Moreover, following CS [30], we also conduct experiments on the image data set *ESPGame* [48], and take those tags appearing at least 20 times in the data set to form a large vocabulary, which almost doubles the size of that used in the experiments of CS. Each instance in *ESPGame* is represented by a 516-D feature vector[1] extracted with Lire [49], and it is removed if no tags are associated. The original statistics of the data sets are given in Table III.

For performance comparison, we select binary relevance (BR) [50], CS [30], PLST [31], CPLST and *kernel*-CPLST [33], MLC-BMaD [34], and ML-CSSP [35] as baselines, where BR is a widely used multilabel classification method that trains a separate BR model for each label. In our experiments, we use both linear SVM (L-SVM) [51] and linear ridge regression (L-RR) for BR. And for the latter, we use 0.5 as a threshold to decide the binary (0 or 1) classification results. To reduce the computational costs of L-SVM on *bibtex* and *bookmarks*, we perform feature dimension reduction for both data sets via PCA. We also follow the reported preprocessing steps of baselines, such as shifting the mean values of feature vectors to be zeros. Note that BR in fact does not perform LSDR and thus its performance is a reference for other algorithms. BML-CS [8] is not included, since it is sophisticated with numerous parameters to tune.

For E^{2}FE, we evaluate the following variants.

1) *R-E^{2}FE:* Considering only the *recoverability* of the label space (i.e., formula (24)), theoretically equivalent to PLST.
2) *P-LinearE^{2}FE:* Considering only the *predictability* of the latent space for linear function-based encoding (i.e., formula (25)), similar to CPLST.
3) *OP-E^{2}FE:* Overemphasizing the *predictability* of the latent space (i.e., formula (27)).
4) *LinearE^{2}FE:* Linear function-based encoding (i.e., formula (20)).
5) *πLinearE^{2}FE:* Identical to LinearE^{2}FE except that the linear decoding matrix is learned with *a priori* knowledge.
6) *E^{2}FE:* End-to-end feature-aware label space encoding (i.e., formula (12)).
7) *πE^{2}FE:* Identical to E^{2}FE except that the linear decoding matrix is learned with *a priori* knowledge.
8) *kernel-E^{2}FE:* Kernel version of E^{2}FE (i.e., formula (23)).
9) *kernel-πE^{2}FE:* Identical to *kernel*-E^{2}FE except that the linear decoding matrix is learned with *a priori* knowledge.

In our experiments, each data set is evenly and randomly divided into five parts. Five runs of each algorithm are then performed on the data set, taking each time one part for testing and the rest for training without duplication. Experimental results are measured with widely used metrics in the field of multilabel classification, i.e., *label-based macroF1* and *example-based Accuracy* [52], and then averaged over the five runs. Higher *label-based macroF1* and *example-based Accuracy* means better performance. Specifically, for each run, *label-based macroF1* is calculated as follows:

$$\text{macroF1} = \frac{1}{d_y}\sum_{i=1}^{d_y}\frac{2p_i r_i}{p_i + r_i}$$
$$\text{s.t. } p_i = \frac{|G_i \cap P_i|}{|P_i|}, r_i = \frac{|G_i \cap P_i|}{|G_i|} \tag{28}$$

where d_y is the number of all labels, G_i and P_i are, respectively, the sets of the ground truth and the predicted positive instances for the ith label, and $\cap$ and $\cup$ are the operations of intersection and union between two sets. Meanwhile, *example-based Accuracy* is given by the following formula:

$$\text{Accuracy} = \frac{1}{n_t}\sum_{j=1}^{n_t}\frac{\left|G'_j \cap P'_j\right|}{\left|G'_j \cup P'_j\right|} \tag{29}$$

where n_t is the test set size, and G'_j and P'_j are, respectively, the ground truth and the predicted label set of the jth test instance.

Moreover, for each run of any algorithm, we conduct 5-fold cross validation on the training set for selecting model parameters via grid search in predefined value ranges. Specifically, α in the proposed E^{2}FE and its variants is selected from $\{10^{-1}, 10^0, \ldots, 10^4\}$, τ for MLC-BMaD is chosen from $\{0.1, 0.2, \ldots, 1.0\}$, and the predefined sparsity level in CS is selected from $\{1, 2, \ldots, M\}$ with M being the maximal number of labels in an instance, and so on. Additionally, for η in πE^{2}FE/πLinearE^{2}FE/*kernel*-πE^{2}FE, we set $\eta = \xi(1/\max(\vec{\lambda}))$ for each data set and select ξ from $\{0, 2^{-10}, 2^{-9}, \ldots, 2^{-1}, 1\}$ via cross validation. Following most previous works, such as [8], [31], [33], and [35], we utilize L-RR as predictive models to learn the mappings from instance features to code vectors. As for *kernel*-CPLST, *kernel*-E^{2}FE, and *kernel*-πE^{2}FE, we empirically utilize the Gaussian kernel function and set the smoothing parameter σ as twice the mean Euclidean distance between feature vectors for each data set. Accordingly, we utilize kernel ridge regression as predictive models for them to learn the nonlinear mappings from instance features to code vectors. Moreover, following PLST, CPLST, and ML-CSSP, we round each continuous entry

[1]516-D feature vector: 60-D Gabor, 192-D FCTH, 80-D Edge Histogram, 120-D Color Layout, and 64-D RGB Color Histogram

TABLE IV
EXPERIMENTAL RESULTS: ***Label-Based macroF1*** ON *Delicious*, *CAL500*, *Mediamill*, *ESPGame*, *Bibtex*, AND *Bookmarks*, WITH VARYING d_z/d_y

Datasets		*delicious*					*CAL500*					*mediamill*				
d_z/d_y		10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
BR [50]	L-SVM	0.0951					0.1397					0.0866				
	L-RR	0.0377					0.0569					0.0447				
CS [30]		0.0063	0.0208	0.0422	0.0466	0.0415	0.0677	0.0820	0.0906	0.0976	0.1142	0.0052	0.0144	0.0138	0.0319	0.0304
PLST [31]		0.0234	0.0256	0.0271	0.0278	0.0284	0.0604	0.0605	0.0606	0.0609	0.0608	0.0422	0.0439	0.0448	0.0447	0.0447
CPLST [33]		0.0339	0.0341	0.0341	0.0341	0.0341	0.0640	0.0643	0.0644	0.0645	0.0645	0.0432	0.0446	0.0447	0.0447	0.0447
MLC-BMaD [34]		0.0238	0.0259	0.0297	0.0344	0.0347	0.0485	0.0444	0.0420	0.0472	0.0468	0.0398	0.0408	0.0408	0.0408	0.0408
ML-CSSP [35]		0.0160	0.0216	0.0277	0.0324	0.0319	0.0453	0.0498	0.0507	0.0528	0.0543	0.0354	0.0395	0.0426	0.0433	0.0427
R-E^2FE (∼PLST)		0.0234	0.0257	0.0271	0.0278	0.0285	0.0592	0.0590	0.0592	0.0593	0.0593	0.0422	0.0439	0.0448	0.0447	0.0447
P-LinearE^2FE (∼CPLST)		0.0391	0.0398	0.0399	0.0399	0.0400	0.0795	0.0954	0.1008	0.1003	0.1003	0.0420	0.0437	0.0446	0.0446	0.0447
OP-E^2FE		0.0449	0.0470	0.0476	0.0478	0.0475	0.1034	0.1080	0.1088	-	-	0.0433	0.0447	0.0448	0.0449	0.0448
LinearE^2FE		0.0413	0.0417	0.0416	0.0416	0.0416	0.1061	0.1115	0.1110	0.1101	0.1101	0.0440	0.0449	0.0451	0.0449	0.0448
E^2FE		**0.0530**	**0.0569**	**0.0577**	**0.0578**	**0.0578**	**0.1198**	**0.1247**	**0.1263**	**0.1258**	**0.1256**	**0.0549**	**0.0575**	**0.0577**	**0.0577**	**0.0577**
kernel-CPLST [33]		0.0354	0.0377	0.0383	0.0389	0.0393	0.0754	0.0774	0.0774	0.0774	0.0774	0.0594	0.0688	0.0720	0.0751	0.0756
kernel-E^2FE		**0.0500**	**0.0569**	**0.0591**	**0.0599**	**0.0599**	**0.1160**	**0.1208**	**0.1215**	**0.1272**	**0.1307**	**0.0692**	**0.0814**	**0.0945**	**0.0997**	**0.1003**

Datasets		*ESPGame*					*bibtex*					*bookmarks*				
d_z/d_y		5%	10%	15%	20%	25%	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
BR [50]	L-SVM	0.0688					0.3023					0.1860				
	L-RR	0.0017					0.0613					0.0415				
CS [30]		0.0005	0.0011	0.0014	0.0022	0.0022	0.0170	0.0377	0.0916	0.1010	0.1017	0.0090	0.0248	0.0271	0.0582	0.0611
PLST [31]		0.0017	0.0017	0.0017	0.0017	0.0017	0.0365	0.0503	0.0540	0.0553	0.0557	0.0248	0.0357	0.0397	0.0403	0.0406
CPLST [33]		0.0017	0.0017	0.0017	0.0017	0.0017	0.0443	0.0560	0.0581	0.0588	0.0588	0.0384	0.0400	0.0401	0.0402	0.0403
MLC-BMaD [34]		0.0017	0.0017	0.0017	0.0017	0.0017	0.0341	0.0505	0.0524	0.0550	0.0582	0.0325	0.0354	0.0385	0.0416	0.0415
ML-CSSP [35]		0.0011	0.0016	0.0014	0.0014	0.0015	0.0281	0.0330	0.0439	0.0480	0.0471	0.0184	0.0292	0.0300	0.0302	0.0339
R-E^2FE (∼PLST)		0.0017	0.0017	0.0017	0.0017	0.0017	0.0358	0.0498	0.0538	0.0552	0.0556	0.0249	0.0358	0.0398	0.0404	0.0407
P-LinearE^2FE (∼CPLST)		0.0017	0.0017	0.0017	0.0017	0.0017	0.0443	0.0548	0.0574	0.0593	0.0601	0.0389	0.0404	0.0417	0.0421	0.0422
OP-E^2FE		0.0021	0.0021	0.0021	0.0021	0.0024	0.0536	0.0810	0.0906	0.0958	0.0981	0.0400	0.0434	0.0451	0.0454	0.0459
LinearE^2FE		0.0018	0.0017	0.0017	0.0017	0.0017	0.0440	0.0564	0.0604	0.0602	0.0601	0.0397	0.0419	0.0423	0.0424	0.0425
E^2FE		**0.0026**	**0.0025**	**0.0025**	**0.0025**	**0.0025**	**0.0595**	**0.0888**	**0.1169**	**0.1286**	**0.1369**	**0.0472**	**0.0706**	**0.0752**	**0.0764**	**0.0775**
kernel-CPLST [33]		0.0019	0.0019	0.0019	0.0019	0.0019	0.0503	0.0698	0.0728	0.0742	0.0744	0.0410	0.0448	0.0462	0.0472	0.0477
kernel-E^2FE		**0.0040**	**0.0043**	**0.0043**	**0.0044**	**0.0045**	**0.0629**	**0.0930**	**0.1247**	**0.1396**	**0.1472**	**0.0492**	**0.0699**	**0.0738**	**0.0756**	**0.0770**

of the decoding results into its nearest 0 or 1 to get the binary label vectors for test instances.

B. Experimental Results of LSDR

We run all algorithms on the six data sets with different values of d_z/d_y (mostly from 10% to 50%) where d_z and d_y are, respectively, the dimensionality of the latent space and that of the label space. Particularly, for *ESPGame*, d_z/d_y is varied from 5% to 25%, as it has a much larger vocabulary.

1) Performance Comparison With Baselines: The experimental results of compared baselines and variants of the proposed E^2FE are reported in Tables IV and V.

A close look at the achieved results reveals: 1) the proposed E^2FE as well as its linear function-based variant LinearE^2FE generally outperform the compared baselines on each data set, which clearly demonstrates their effectiveness; 2) E^2FE outperforms LinearE^2FE on all data sets, reflecting the superiority of learning code vectors in an end-to-end manner rather than a function-based manner; 3) E^2FE outperforms R-E^2FE and LinearE^2FE outperforms P-LinearE^2FE, which implies that jointly considering *predictability* and *recoverability* will obtain better performance; 4) OP-E^2FE yields inferior performance to E^2FE and cannot even perform LSDR on *CAL500* when $d_z/d_y \geq 40\%$, as the dimensionality of the feature space will be smaller than d_z; that points out the weakness of OP-E^2FE and further validates the superiority of keeping a good tradeoff between *predictability* and *recoverability*; 5) R-E^2FE yields nearly the same performance as PLST, as predicted by our theoretical analyses about their equivalence; 6) with an identical objective function but different orthogonality constraints, P-LinearE^2FE seems to be slightly superior to CPLST, which validates the reasonableness of assuming the columns of the code matrix to be orthonormal; 7) *kernel*-E^2FE outperforms E^2FE on nearly all data sets, showing its effectiveness to handle the nonlinear correlations between the feature space and the latent space; moreover, *kernel*-E^2FE achieves superior performance to *kernel*-CPLST, which outperforms CPLST. And 8) On all data sets, as d_z/d_y increases, the performance of E^2FE does not vary dramatically due to the orthonormality constraint in formula (4), which leads E^2FE to compactly encode the label space with a smaller d_z. Similar phenomenon occurs when applying PLST and CPLST, because both are also orthogonally constrained. Actually, we find that this phenomenon still remains when $d_z/d_y > 50\%$.

Actually, to evaluate the significance of the performance improvements gained by E^2FE over the baselines, we also perform *paired-sample t-test* [53] for both *label-based macroF1* and *example-based Accuracy* on all data sets with varying d_z/d_y. Experimental results show that nearly all *P-values* in significance tests are less than the typical significance level 0.01, and thus, the performance improvements gained by E^2FE over baselines are statistically significant. For details about the experiments of significance tests, one can refer to the supplementary material.

2) Experimental Validation for Enhancing Decoding Matrix with a Priori Knowledge: Here, on each data set we compare LinearE^2FE, E^2FE, and *kernel*-E^2FE with their counterparts that learn the linear decoding matrix with *a priori* knowledge, i.e., πLinearE^2FE, πE^2FE, and *kernel*-πE^2FE, as presented in Tables VI and VII. Note that in all pairwise comparisons,

TABLE V

EXPERIMENTAL RESULTS: ***Example-Based Accuracy*** ON *Delicious*, *CAL500*, *Mediamill*, *ESPGame*, *Bibtex*, AND *Bookmarks*, WITH VARYING d_z/d_y

Datasets		*delicious*					*CAL500*					*mediamill*				
d_z/d_y		10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
BR [50]	L-SVM	0.1500					0.2436					0.3621				
	L-RR	0.0958					0.1995					0.4188				
CS [30]		0.0254	0.0540	0.0890	0.0974	0.0964	0.1130	0.1299	0.1626	0.1904	0.1835	0.0115	0.0304	0.0352	0.1425	0.1426
PLST [31]		0.0870	0.0898	0.0907	0.0911	0.0912	0.2099	0.2103	0.2103	0.2106	0.2104	0.4160	0.4184	0.4187	0.4188	0.4187
CPLST [33]		0.0954	0.0955	0.0955	0.0955	0.0955	0.2003	0.2007	0.2009	0.2010	0.2010	0.4167	0.4187	0.4187	0.4188	0.4187
MLC-BMaD [34]		0.0593	0.0700	0.0855	0.0873	0.0875	0.1286	0.1215	0.1194	0.1244	0.1255	0.3989	0.3995	0.3995	0.3995	0.3995
ML-CSSP [35]		0.0684	0.0785	0.0851	0.0893	0.0904	0.1806	0.1880	0.1913	0.1958	0.1966	0.3466	0.4053	0.4073	0.4140	0.4081
R-E^{2}FE (∼PLST)		0.0870	0.0898	0.0908	0.0911	0.0913	0.2100	0.2098	0.2099	0.2101	0.2100	0.4159	0.4183	0.4188	0.4188	0.4187
P-LinearE^{2}FE (∼CPLST)		0.0984	0.1007	0.1011	0.1011	0.1011	0.2084	0.2189	0.2226	0.2223	0.2223	0.4137	0.4162	0.4182	0.4182	0.4186
OP-E^{2}FE		0.1085	0.1091	0.1093	0.1094	0.1073	0.2283	0.2262	0.2251	-	-	0.4172	0.4186	0.4189	0.4189	0.4189
LinearE^{2}FE		0.1068	0.1055	0.1049	0.1048	0.1048	0.2318	0.2301	0.2291	0.2281	0.2281	0.4182	0.4190	0.4191	0.4190	0.4189
E^{2}FE		**0.1187**	**0.1196**	**0.1197**	**0.1196**	**0.1184**	**0.2405**	**0.2411**	**0.2421**	**0.2396**	**0.2392**	**0.4353**	**0.4379**	**0.4378**	**0.4378**	**0.4378**
kernel-CPLST [33]		0.1116	0.1162	0.1175	0.1181	0.1186	0.2139	0.2148	0.2148	0.2148	0.2148	0.4489	0.4561	0.4572	0.4579	0.4580
kernel-E^{2}FE		**0.1281**	**0.1291**	**0.1311**	**0.1312**	**0.1293**	**0.2421**	**0.2414**	**0.2397**	**0.2399**	**0.2398**	**0.4606**	**0.4647**	**0.4681**	**0.4685**	**0.4686**

Datasets		*ESPGame*					*bibtex*					*bookmarks*				
d_z/d_y		5%	10%	15%	20%	25%	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
BR [50]	L-SVM	0.0628					0.2827					0.1679				
	L-RR	0.0572					0.1816					0.1597				
CS [30]		0.0053	0.0057	0.0048	0.0107	0.0106	0.0771	0.0964	0.1435	0.1520	0.1553	0.0155	0.0629	0.0643	0.0940	0.0963
PLST [31]		0.0576	0.0576	0.0576	0.0576	0.0576	0.1434	0.1657	0.1760	0.1768	0.1772	0.1472	0.1530	0.1572	0.1575	0.1578
CPLST [33]		0.0578	0.0578	0.0578	0.0578	0.0578	0.1639	0.1768	0.1793	0.1802	0.1800	0.1542	0.1570	0.1572	0.1572	0.1573
MLC-BMaD [34]		0.0574	0.0573	0.0573	0.0575	0.0573	0.1365	0.1757	0.1767	0.1746	0.1799	0.1494	0.1538	0.1576	0.1597	0.1596
ML-CSSP [35]		0.0409	0.0548	0.0478	0.0541	0.0499	0.1212	0.1205	0.1433	0.1563	0.1573	0.1053	0.1493	0.1249	0.1063	0.1438
R-E^{2}FE (∼PLST)		0.0575	0.0575	0.0575	0.0575	0.0575	0.1429	0.1653	0.1756	0.1766	0.1769	0.1473	0.1532	0.1574	0.1577	0.1580
P-LinearE^{2}FE (∼CPLST)		0.0574	0.0577	0.0578	0.0578	0.0579	0.1598	0.1726	0.1756	0.1783	0.1792	0.1552	0.1574	0.1582	0.1588	0.1590
OP-E^{2}FE		0.0622	0.0621	0.0621	0.0621	0.0694	0.1751	0.2041	0.2126	0.2167	0.2181	0.1565	0.1616	0.1623	0.1625	0.1626
LinearE^{2}FE		0.0597	0.0593	0.0588	0.0586	0.0584	0.1618	0.1768	0.1820	0.1816	0.1813	0.1559	0.1592	0.1591	0.1590	0.1589
E^{2}FE		**0.0701**	**0.0701**	**0.0700**	**0.0701**	**0.0701**	**0.1835**	**0.2149**	**0.2356**	**0.2440**	**0.2493**	**0.1659**	**0.1913**	**0.1933**	**0.1937**	**0.1939**
kernel-CPLST [33]		0.0646	0.0645	0.0645	0.0645	0.0645	0.1739	0.1976	0.2005	0.2006	0.2011	0.1588	0.1638	0.1641	0.1643	0.1645
kernel-E^{2}FE		**0.0832**	**0.0834**	**0.0834**	**0.0834**	**0.0834**	**0.1910**	**0.2232**	**0.2485**	**0.2596**	**0.2640**	**0.1685**	**0.1937**	**0.1952**	**0.1958**	**0.1961**

TABLE VI

PERFORMANCE COMPARISONS BETWEEN LINEARE^{2}FE, E^{2}FE, *Kernel*-E^{2}FE AND πLINEARE^{2}FE, πE^{2}FE, *Kernel*-πE^{2}FE ON *Delicious*, *CAL500*, *Mediamill*, *ESPGame*, *Bibtex*, AND *Bookmarks* WITH VARYING d_z/d_y, IN TERMS OF ***Label-Based macroF1***

Datasets	*delicious*					*CAL500*					*mediamill*				
d_z/d_y	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
LinearE^{2}FE	0.0413	0.0417	0.0416	0.0416	0.0416	0.1061	0.1115	0.1110	0.1101	0.1101	0.0440	0.0449	0.0451	0.0449	0.0448
πLinearE^{2}FE	**0.0590**	**0.0595**	**0.0594**	**0.0593**	**0.0593**	**0.1415**	**0.1457**	**0.1479**	**0.1477**	**0.1478**	**0.0580**	**0.0605**	**0.0607**	**0.0604**	**0.0603**
Relative Improvement	43.0%	42.7%	42.5%	42.6%	42.6%	33.3%	30.8%	33.2%	34.1%	34.1%	31.8%	34.7%	34.7%	34.7%	34.7%
E^{2}FE	0.0530	0.0569	0.0577	0.0578	0.0578	0.1198	0.1247	0.1263	0.1258	0.1256	0.0549	0.0575	0.0577	0.0577	0.0577
πE^{2}FE	**0.0698**	**0.0727**	**0.0735**	**0.0738**	**0.0717**	**0.1841**	**0.1874**	**0.1883**	**0.1925**	**0.1923**	**0.0685**	**0.0718**	**0.0720**	**0.0721**	**0.0721**
Relative Improvement	31.8%	27.8%	27.5%	27.8%	24.0%	53.7%	50.3%	49.0%	53.0%	53.1%	24.9%	24.8%	24.8%	24.9%	24.8%
kernel-E^{2}FE	0.0500	0.0569	0.0591	0.0599	0.0599	0.1160	0.1208	0.1215	0.1272	0.1307	0.0692	0.0814	0.0945	0.0997	0.1003
kernel-πE^{2}FE	**0.0768**	**0.0820**	**0.0834**	**0.0838**	**0.0843**	**0.1827**	**0.1864**	**0.1927**	**0.2000**	**0.2024**	**0.0838**	**0.0975**	**0.1117**	**0.1172**	**0.1177**
Relative Improvement	53.4%	44.0%	41.1%	39.8%	40.6%	57.4%	54.3%	58.6%	57.3%	54.9%	21.1%	19.8%	18.2%	17.5%	17.3%

Datasets	*ESPGame*					*bibtex*					*bookmarks*				
d_z/d_y	5%	10%	15%	20%	25%	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
LinearE^{2}FE	0.0018	0.0017	0.0017	0.0017	0.0017	0.0440	0.0564	0.0604	0.0602	0.0601	0.0397	0.0419	0.0423	0.0424	0.0425
πLinearE^{2}FE	**0.0029**	**0.0028**	**0.0028**	**0.0028**	**0.0028**	**0.0801**	**0.1076**	**0.1186**	**0.1217**	**0.1216**	**0.0518**	**0.0569**	**0.0582**	**0.0593**	**0.0592**
Relative Improvement	64.4%	63.5%	64.0%	64.0%	64.0%	81.9%	90.9%	96.4%	102.2%	102.3%	30.4%	35.8%	37.6%	39.9%	39.4%
E^{2}FE	0.0026	0.0025	0.0025	0.0025	0.0025	0.0595	0.0888	0.1169	0.1286	0.1369	0.0472	0.0706	0.0752	0.0764	0.0775
πE^{2}FE	**0.0037**	**0.0037**	**0.0038**	**0.0038**	**0.0038**	**0.1204**	**0.1874**	**0.2264**	**0.2458**	**0.2583**	**0.0751**	**0.0883**	**0.0948**	**0.0976**	**0.0989**
Relative Improvement	43.5%	46.5%	48.3%	48.2%	48.0%	102.1%	111.0%	93.7%	91.2%	88.7%	59.1%	25.1%	26.0%	27.7%	27.6%
kernel-E^{2}FE	0.0040	0.0043	0.0043	0.0044	0.0045	0.0629	0.0930	0.1247	0.1396	0.1472	0.0492	0.0699	0.0738	0.0756	0.0770
kernel-πE^{2}FE	**0.0054**	**0.0057**	**0.0058**	**0.0059**	**0.0060**	**0.1252**	**0.1936**	**0.2346**	**0.2575**	**0.2740**	**0.0764**	**0.0876**	**0.0936**	**0.0974**	**0.0990**
Relative Improvement	36.4%	33.8%	34.4%	33.6%	33.6%	99.0%	108.2%	88.2%	84.5%	86.1%	55.4%	25.5%	26.9%	28.7%	28.6%

we also present the relative performance improvements gained by πLinearE^{2}FE/πE^{2}FE/*kernel*-πE^{2}FE.

From the comparisons, we can see that in nearly all cases, πLinearE^{2}FE outperforms LinearE^{2}FE, πE^{2}FE outperforms E^{2}FE, and *kernel*-πE^{2}FE outperforms *kernel*-E^{2}FE. Specifically, on average, considering *a priori* knowledge for learning the decoding matrix can achieve a relative improvement of 48.1% for *label-based macroF1* and 33.9% for *example-based Accuracy*. Meanwhile, the maximal gained relative improvement for the former is 111%, and that for the latter is 86%. Such significant performance improvements well demonstrate the effectiveness of our proposal to further consider the eigenvalues corresponding to each column of the code matrix as *a priori* knowledge to enhance the linear decoding matrix in E^{2}FE and its variants.

C. Analyses of Training Costs

1) Comparison with Baselines: For E^{2}FE and compared baselines, apart from classification performance, here we also compare their training costs theoretically and experimentally.

Considering that the training costs of all algorithms mainly differ in those of performing LSDR, i.e., learning the code

TABLE VII

PERFORMANCE COMPARISONS BETWEEN LINEARE^{2}FE, E^{2}FE, *Kernel*-E^{2}FE AND πLINEARE^{2}FE, πE^{2}FE, *Kernel*-πE^{2}FE ON *Delicious*, *CAL500*, *Mediamill*, *ESPGame*, *Bibtex* AND *Bookmarks* WITH VARYING d_z/d_y, IN TERMS OF ***Example-Based Accuracy***

Datasets	*delicious*					*CAL500*					*mediamill*				
d_z/d_y	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
LinearE^{2}FE	0.1068	0.1055	0.1049	0.1048	0.1048	0.2318	0.2301	0.2291	0.2281	0.2281	0.4182	0.4190	0.4191	0.4190	0.4189
πLinearE^{2}FE	**0.1465**	**0.1457**	**0.1453**	**0.1453**	**0.1453**	**0.2555**	**0.2495**	**0.2500**	**0.2497**	**0.2497**	**0.4256**	**0.4278**	**0.4277**	**0.4275**	**0.4274**
Relative Improvement	37.2%	38.0%	38.6%	38.6%	38.6%	10.2%	8.4%	9.1%	9.5%	9.5%	1.8%	2.1%	2.0%	2.0%	2.0%
E^{2}FE	0.1187	0.1196	0.1197	0.1196	0.1184	0.2405	0.2411	0.2421	0.2396	0.2392	**0.4353**	**0.4379**	**0.4378**	**0.4378**	**0.4378**
πE^{2}FE	**0.1971**	**0.1975**	**0.1974**	**0.1974**	**0.1980**	**0.3121**	**0.3044**	**0.3039**	**0.3042**	**0.3040**	0.4280	0.4303	0.4304	0.4305	0.4305
Relative Improvement	66.0%	65.1%	64.9%	65.0%	67.2%	29.8%	26.2%	25.5%	27.0%	27.1%	-1.7%	-1.7%	-1.7%	-1.6%	-1.7%
kernel-E^{2}FE	0.1281	0.1291	0.1311	0.1312	0.1293	0.2421	0.2414	0.2397	0.2399	0.2398	**0.4606**	**0.4647**	**0.4681**	**0.4685**	**0.4686**
kernel-πE^{2}FE	**0.2230**	**0.2244**	**0.2246**	**0.2246**	**0.2247**	**0.3221**	**0.3187**	**0.3166**	**0.3103**	**0.3050**	0.4533	0.4605	0.4629	0.4633	0.4632
Relative Improvement	74.1%	73.8%	71.3%	71.2%	73.7%	33.1%	32.0%	32.1%	29.3%	27.2%	-1.6%	-0.9%	-1.1%	-1.1%	-1.1%

Datasets	*ESPGame*					*bibtex*					*bookmarks*				
d_z/d_y	5%	10%	15%	20%	25%	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
LinearE^{2}FE	0.0597	0.0593	0.0588	0.0586	0.0584	0.1618	0.1768	0.1820	0.1816	0.1813	0.1559	0.1592	0.1591	0.1590	0.1589
πLinearE^{2}FE	**0.1024**	**0.1018**	**0.1015**	**0.1013**	**0.1012**	**0.2000**	**0.2276**	**0.2358**	**0.2399**	**0.2407**	**0.1756**	**0.1833**	**0.1835**	**0.1835**	**0.1832**
Relative Improvement	71.4%	71.7%	72.5%	72.8%	73.2%	23.6%	28.7%	29.5%	32.1%	32.8%	12.6%	15.1%	15.3%	15.4%	15.3%
E^{2}FE	0.0701	0.0701	0.0700	0.0701	0.0701	0.1835	0.2149	0.2356	0.2440	0.2493	0.1659	0.1913	0.1933	0.1937	0.1939
πE^{2}FE	**0.1223**	**0.1303**	**0.1303**	**0.1303**	**0.1302**	**0.2449**	**0.2973**	**0.3219**	**0.3296**	**0.3333**	**0.2089**	**0.2256**	**0.2285**	**0.2294**	**0.2295**
Relative Improvement	74.4%	86.0%	86.0%	85.9%	85.8%	33.4%	38.4%	36.6%	35.1%	33.7%	26.0%	18.0%	18.2%	18.4%	18.3%
kernel-E^{2}FE	0.0832	0.0834	0.0834	0.0834	0.0834	0.1910	0.2232	0.2485	0.2596	0.2640	0.1685	0.1937	0.1952	0.1958	0.1961
kernel-πE^{2}FE	**0.1334**	**0.1335**	**0.1336**	**0.1336**	**0.1336**	**0.2490**	**0.3045**	**0.3296**	**0.3398**	**0.3463**	**0.2138**	**0.2273**	**0.2296**	**0.2308**	**0.2311**
Relative Improvement	60.4%	60.1%	60.3%	60.2%	60.2%	30.4%	36.4%	32.6%	30.9%	31.2%	26.9%	17.3%	17.6%	17.9%	17.9%

TABLE VIII

TIME COMPLEXITY OF COMPARED ALGORITHMS TO PERFORM LSDR

	Time Complexity
CS [30]	$\mathcal{O}(nd_yd_z)$
PLST [31]	$\mathcal{O}(nd_yd_z)$
CPLST [33]	$\mathcal{O}(\min\{n^2d_x, nd_x^2\}) + \mathcal{O}(2nd_xd_y + d_xd_y^2) + \mathcal{O}(d_y^3)$
MLC-BMaD [34]	$\mathcal{O}(nd_y^2d_z)$
ML-CSSP [35]	$\mathcal{O}(nd_yd_z) + \mathcal{O}(d_z\log d_z)$
E^{2}FE	$\min\{\mathcal{O}(nd_x^2) + \mathcal{O}(n(d_x+d_y)^2) + \mathcal{O}(d_xd_z^2 + d_yd_z^2),$ $\mathcal{O}(\min\{n^2d_x, nd_x^2\}) + \mathcal{O}(n^2d_x + n^2d_y) + \mathcal{O}(nd_z^2)\}$

vectors of training instances and the decoding process, here we summarize the time complexity of each algorithm, as presented in Table VIII. From the time complexity analysis, it can be seen that MLC-BMaD has the highest time complexity, while CS, PLST, and ML-CSSP have the lowest.

Moreover, in Table IX, we also report the average time costs for E^{2}FE and the compared baselines on performing LSDR and training predictive models over five runs on *delicious*, *ESPGame*, and *bookmarks*, which have the largest label sets, with $d_z/d_y = 10\%$. As a reference, the time costs of BR are also provided. All algorithms are conducted with MATLAB R2013a on a server with two Intel Xeon E5-2430 CPUs and 64-GB RAM, except that BR with L-SVM is conducted using LIBLINEAR [51]. Looking at the results of this comparison, we can draw the following conclusions: 1) compared with BR, nearly all LSDR methods can help to reduce the total training costs; 2) for performing LSDR, E^{2}FE generally needs slightly higher costs than CS, PLST, CPLST, and ML-CSSP, though with superior classification performance; also, its training cost is much lower than MLC-BMaD; and 3) like the previous theoretical analysis, the training cost of MLC-BMaD is the highest while those of CS, PLST, and ML-CSSP are the lowest.

2) Evaluation of the Newly Proposed Optimization Method: To evaluate the optimization method proposed in this paper for efficiently learning the code matrix in cases with $n \gg d_x + d_y$, we also conduct experiments on *delicious*, *ESPGame*, and *bookmarks* to compare its efficiency with that of the optimization method presented in our conference paper [36]. Here, we, respectively, denote the former as E^{2}FE and the latter as FaIE.

Considering that FaIE needs to calculate the matrix $\mathbf{U} \in \mathbb{R}^{n\times n}$ and thus needs much memory space for large data sets, to avoid biases brought by high memory space costs, here we follow the experimental settings in [36] and sample 5000 training instances for evaluating the time costs of both optimization methods to perform LSDR. Note that here $n \gg d_x + d_y$ is still ensured for the sampled training instances. Experimental results on the three data sets with $d_z/d_y = 10\%$ are reported in Table X. It can be seen that the time costs of the newly proposed optimization method are significantly lower than those of the one presented in [36]. That clearly demonstrates the effectiveness of the newly proposed optimization method for cases with $n \gg d_x + d_y$.

D. Parameter Sensitivity Analyses

For a more detailed view, we also conduct experiments to see the effects of α (i.e., formula (12)) on the performance of the proposed E^{2}FE. Fig. 2 shows how the performance of E^{2}FE changes as α varies in $\{10^{-2}, 10^{-1}, \ldots, 10^4, 10^5\}$ in a run on the largest *delicious*, *ESPGame*, and *bookmarks* with $d_z/d_y = 10\%$. It can be seen that on these three data sets, the performance of E^{2}FE, in terms of *label-based macroF1* and *example-based Accuracy*, first increases and then decreases as α increases from 10^{-2} to 10^5. That further demonstrates the reasonableness of jointly considering the *recoverability* of the label space and the *predictability* of the latent space, as a good tradeoff between both yields superior performance. Moreover, we can observe that for these three data sets, the optimal α value for E^{2}FE is near $[10^3, 10^4]$.

By fixing $\alpha = 10^3$, we further analyse the effects of η (i.e., formula (18)) on the performance of πE^{2}FE. Specif-

TABLE IX

AVERAGE TRAINING COSTS (IN SECONDS) OF COMPARED ALGORITHMS ("PERFORMING LSDR + TRAINING PREDICTIVE MODELS") WITH $d_z/d_y = 10\%$

		delicious		*ESPGame*		*bookmarks*	
BR [50]	L-SVM	187.400	(0.000 + 187.400)	20,487.185	(0.000 + 20,487.185)	24,809.120	(0.000 + 24,809.120)
	L-RR	15.428	(0.000 + 15.428)	54.666	(0.000 + 54.666)	13.310	(0.000 + 13.310)
CS [30]		**1.782**	**(0.150 + 1.632)**	**7.235**	**(1.229 + 6.006)**	**2.316**	**(0.169 + 2.147)**
PLST [31]		2.271	(0.637 + 1.635)	11.341	(5.310 + 6.031)	2.360	(0.190 + 2.170)
CPLST [33]		2.799	(1.033 + 1.766)	11.594	(5.563 + 6.031)	3.570	(1.426 + 2.143)
MLC-BMaD [34]		68.625	(66.937 + 1.688)	585.494	(579.388 + 6.106)	14.475	(12.344 + 2.131)
ML-CSSP [35]		2.574	(0.780 + 1.794)	11.074	(5.061 + 6.013)	2.328	(0.188 + 2.140)
E^2FE		3.787	(1.984 + 1.803)	16.936	(10.877 + 6.059)	7.039	(4.877 + 2.162)

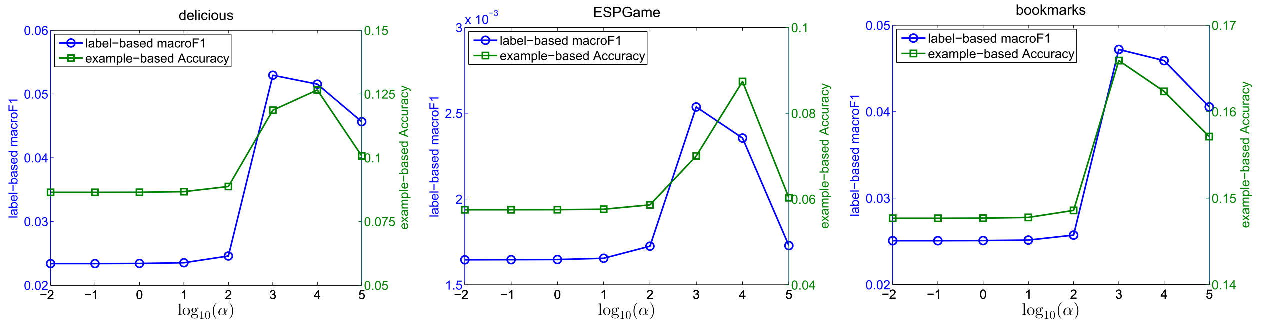

Fig. 2. Effects of α on the performance of E^2FE on *delicious* (left), *ESPGame* (center), and *bookmarks* (right), with $d_z/d_y = 10\%$.

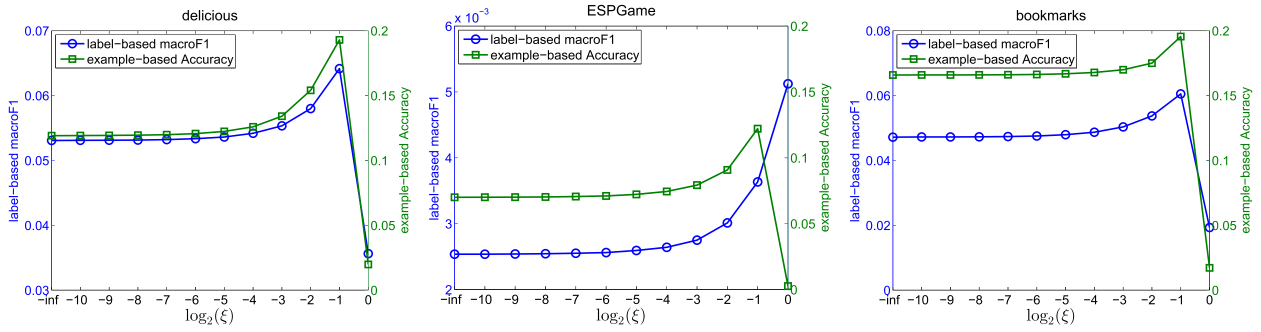

Fig. 3. Effects of $\eta = \xi(1/\max(\vec{\lambda}))$ on the performance of πE^2FE on *delicious* (left), *ESPGame* (center), and *bookmarks* (right), with $d_z/d_y = 10\%$.

TABLE X

AVERAGE TIME COSTS (IN SECONDS) FOR THE NEWLY PROPOSED OPTIMIZATION METHOD (E^2FE) AND THAT PRESENTED IN OUR PREVIOUS CONFERENCE PAPER (FaIE) TO PERFORM LSDR WITH $d_z/d_y = 10\%$

	deliciousⓈ	*ESPGame*Ⓢ	*bookmarks*Ⓢ
E^2FE	1.077	2.646	0.683
FaIE [36]	3.869	7.800	2.153

ically, on the largest *delicious*, *ESPGame*, and *bookmarks* with $d_z/d_y = 10\%$, we rewrite $\eta = \xi(1/\max(\vec{\lambda}))$ and vary ξ in $\{0, 2^{-10}, 2^{-9}, \ldots, 2^{-1}, 1\}$ based on Lemma 3 to see how the corresponding learned linear decoding matrix affects the performance of πE^2FE, as shown in Fig. 3. It can be seen that on the three data sets, as η increases from 0 (i.e., $\xi = 2^{-\text{inf}} = 0$) to $(1/\max(\vec{\lambda}))$ (i.e., $\xi = 2^0 = 1$), the performance of πE^2FE, in terms of *label-based macroF1* and *example-based Accuracy*, tends to first increase and then decrease in most cases, with the optimal η value being near $(0.5/\max(\vec{\lambda}))$ (i.e., $\xi = 2^{-1}$). Actually, $\eta = 0$ makes πE^2FE degenerate to E^2FE, and it generally yields inferior performance than $\eta > 0$, which further demonstrates the reasonableness of considering the eigenvalues with respect to columns of the code matrix as a priori knowledge to learn an enhanced linear decoding matrix.

VIII. DISCUSSION

The proposed E^2FE assumes that the columns of the to-be-learned code matrix $\mathbf{Z}$ are orthonormal. Though this assumption seems to be strong, it is still reasonable and brings useful properties to E^2FE.

1) As each column of $\mathbf{Z}$ denotes one dimension of the latent space, adding an orthonormality assumption, similar to PLST and CPLST, allows us to mitigate the redundant information among dimensions of the latent space and then enable E^2FE to encode the label space more compactly.
2) As can be seen in formula (2)–(6), adding the orthonormality assumption can simplify the objective function of E^2FE and enable it to be transformed into an eigenvalue problem for efficient optimization.

TABLE XI

PERFORMANCE COMPARISONS BETWEEN E^2FE AND E^2FE_{NoOrth} ON ALL DATA SETS, WITH $p = 5$ FOR *ESPGame* AND $p = 10$ FOR OTHERS

		d_z/d_y	$p\%$	$2p\%$	$3p\%$	$4p\%$	$5p\%$
label-based macroF1	*delicious*	E^2FE	**0.0530**	**0.0569**	**0.0577**	**0.0578**	**0.0578**
		E^2FE_{NoOrth}	0.0336	0.0401	0.0444	0.0462	0.0438
	CAL500	E^2FE	**0.1198**	**0.1247**	**0.1263**	**0.1258**	**0.1256**
		E^2FE_{NoOrth}	0.0966	0.1086	0.1074	0.1122	0.1120
	mediamill	E^2FE	**0.0549**	**0.0575**	**0.0577**	**0.0577**	**0.0577**
		E^2FE_{NoOrth}	0.0436	0.0448	0.0459	0.0451	0.0453
	ESPGame	E^2FE	**0.0026**	**0.0025**	**0.0025**	**0.0025**	**0.0025**
		E^2FE_{NoOrth}	0.0018	0.0018	0.0017	0.0018	0.0018
	bibtex	E^2FE	**0.0595**	**0.0888**	**0.1169**	**0.1286**	**0.1369**
		E^2FE_{NoOrth}	0.0420	0.0727	0.0847	0.0929	0.1000
	bookmarks	E^2FE	**0.0472**	**0.0706**	**0.0752**	**0.0764**	**0.0775**
		E^2FE_{NoOrth}	0.0298	0.0455	0.0459	0.0471	0.0488
example-based Accuracy	*delicious*	E^2FE	**0.1187**	**0.1196**	**0.1197**	**0.1196**	**0.1184**
		E^2FE_{NoOrth}	0.1037	0.1073	0.1082	0.1093	0.1045
	CAL500	E^2FE	**0.2405**	**0.2411**	**0.2421**	**0.2396**	**0.2392**
		E^2FE_{NoOrth}	0.2261	0.2294	0.2256	0.2267	0.2250
	mediamill	E^2FE	**0.4353**	**0.4379**	**0.4378**	**0.4378**	**0.4378**
		E^2FE_{NoOrth}	0.4192	0.4201	0.4204	0.4192	0.4199
	ESPGame	E^2FE	**0.0701**	**0.0701**	**0.0700**	**0.0701**	**0.0701**
		E^2FE_{NoOrth}	0.0613	0.0635	0.0605	0.0585	0.0590
	bibtex	E^2FE	**0.1835**	**0.2149**	**0.2356**	**0.2440**	**0.2493**
		E^2FE_{NoOrth}	0.1501	0.2018	0.2121	0.2167	0.2233
	bookmarks	E^2FE	**0.1659**	**0.1913**	**0.1933**	**0.1937**	**0.1939**
		E^2FE_{NoOrth}	0.1521	0.1627	0.1636	0.1645	0.1660

3) By enabling the objective function to be transformed into an eigenvalue problem, from formula (15) and (16), we can see that adding the orthonormality assumption actually helps to ensure E^2FE obtaining global optima.

Here, we also try dropping the orthonormality assumption from the objective function of E^2FE and conduct experiments on all data sets to evaluate it. We denote it as E^2FE_{NoOrth}, with its objective function given as follows:

$$\begin{aligned}\Psi &= \max_{\mathbf{Z},\mathbf{Q}} -\|\mathbf{Y}-\mathbf{ZQ}\|_{\text{fro}}^2 + \alpha\mathbf{Tr}[\mathbf{Z}^T\mathbf{HZ}]\\ &= \min_{\mathbf{Z},\mathbf{Q}} \|\mathbf{Y}-\mathbf{ZQ}\|_{\text{fro}}^2 - \alpha\mathbf{Tr}[\mathbf{Z}^T\mathbf{HZ}]\\ &\quad\text{s.t. } \forall i \in \{1,2,\ldots,d_z\},\quad \mathbf{Z}_{\cdot,i}^T\mathbf{Z}_{\cdot,i} = 1. \end{aligned} \tag{30}$$

Note that $\mathbf{Z}_{\cdot,i}^T\mathbf{Z}_{\cdot,i} = 1$ is still required due to formula (7). Like other matrix factorization methods [54], [55], we can derive the code matrix $\mathbf{Z}$ and the linear decoding matrix $\mathbf{Q}$ by using gradient descent methods to iteratively and alternatively optimize one while keeping the other fixed until convergence. For more details, one can refer to the supplementary material.

The performance of E^2FE_{NoOrth} on all data sets, using random initial values for $\mathbf{Z}$ and $\mathbf{Q}$ to perform optimization, is presented in Table XI. It can be seen that E^2FE_{NoOrth} is inferior to E^2FE. In fact, even using the derived $\mathbf{Z}$ and $\mathbf{Q}$ of E^2FE as initial values, E^2FE_{NoOrth} can hardly gain performance improvement over E^2FE. It is mainly attributed to that, without the orthonormality assumption: 1) more redundant information rather than complementary information exists between dimensions of the latent space and 2) global optima cannot be ensured for $\mathbf{Z}$.

Moreover, we also evaluate the time costs for E^2FE_{NoOrth} to perform LSDR on the largest *delicious*, *ESPGame*, and *bookmarks* with $d_z/d_y = 10\%$, as reported in Table XII. We can see that E^2FE_{NoOrth} costs much more time than E^2FE, because its objective function is more complex for optimization. Its time costs are even higher than those of the BR model L-RR (Table IX).

TABLE XII

AVERAGE TIME COSTS (IN SECONDS) FOR PERFORMING LSDR IN E^2FE AND E^2FE_{NoOrth} WITH $d_z/d_y = 10\%$

	$delicious$	$ESPGame$	$bookmarks$
E^2FE	1.984	10.877	4.877
E^2FE_{NoOrth}	145.253	215.199	41.434

Therefore, dropping the orthonormality assumption does not bring substantial performance improvements and instead will increase the time costs for optimization. On the contrary, keeping the orthonormality assumption gains a good tradeoff between efficiency and effectiveness for LSDR.

IX. CONCLUSION

Aiming to address the multilabel classification problem with many classes, in this paper, we have proposed an effective method termed E^2FE to perform LSDR via end-to-end feature-aware label space encoding. In contrast to most previous works, E^2FE requires no encoding functions, and it directly learns a feature-aware code matrix via jointly maximizing the *recoverability* of the label space and the *predictability* of the latent space. Subsequently, a linear decoding matrix is further learned for efficiently recovering the predicted label vectors of unseen instances from their corresponding code vectors generated by trained predictive models. The proposed E^2FE has close connections to several previous works. It can also be specified to learn an encoding function as previous works, or extended with kernel tricks to handle nonlinear correlations between the feature space and the latent space.

REFERENCES

[1] L. Wang, X. Y. Zhang, and C. Pan, "MSDLSR: Margin scalable discriminative least squares regression for multicategory classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2711–2717, Dec. 2016.

[2] J. Ortigosa-Hernández, I. Inza, and J. A. Lozano, "Semisupervised multiclass classification problems with scarcity of labeled data: A theoretical study," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2602–2614, Dec. 2016.

[3] D. Zhang, J. Han, J. Han, and L. Shao, "Cosaliency detection based on intrasaliency prior transfer and deep intersaliency mining," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 6, pp. 1163–1176, Jun. 2016.

[4] D. Zhang, J. Han, C. Li, J. Wang, and X. Li, "Detection of co-salient objects by looking deep and wide," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 120, no. 2, pp. 215–232, Nov. 2016.

[5] X. Qian, Y. Zhao, and J. Han, "Image location estimation by salient region matching," *IEEE Trans. Image Processing*, vol. 24, no. 11, pp. 4348–4358, Nov. 2015.

[6] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Multilabel text classification for automated tag suggestion," in *Proc. ECML/PKDD Discovery Challenge Workshop*, 2008.

[7] K. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multi-label classification of music into emotions," in *Proc. Int. Conf. Music Inf. Retrieval*, 2008.

[8] A. Kapoor, R. Viswanathan, and P. Jain, "Multilabel classification using Bayesian compressed sensing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012.

[9] B. Taskar, C. Guestrin, and D. Koller, "Max-margin Markov networks," *Adv. Neural Inf. Process. Syst.*, vol. 16, p. 25, Feb. 2003.

[10] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognit.*, vol. 37, no. 9, pp. 1757–1771, Sep. 2004.

[11] T. Evgeniou, C. Micchelli, and M. Pontil, "Learning multiple tasks with kernel methods," *J. Mach. Learn. Res.*, vol. 6, pp. 615–637, Apr. 2005.

[12] I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, and Y. Singer, "Large margin methods for structured and interdependent output variables," *J. Mach. Learn. Res.*, vol. 6, no. 2, pp. 1453–1484, 2006.

[13] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *Int. J. Data Warehousing Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[14] A. Argyriou, T. Evgeniou, and M. Pontil, "Convex multi-task feature learning," *Mach. Learn.*, vol. 73, no. 3, pp. 243–272, 2008.
[15] B. Hariharan, L. Zelnik-Manor, S. Vishwanathan, and M. Varma, "Large scale max-margin multi-label classification with priors," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 423–430.
[16] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data Mining and Knowledge Discovery Handbook*. New York, NY, USA: Springer, 2010, pp. 667–685.
[17] T. Zhou and D. Tao, "Multi-label subspace ensemble," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2012, pp. 1444–1452.
[18] C.-S. Ferng and H.-T. Lin, "Multilabel classification using error-correcting codes of hard or soft bits," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 11, pp. 1888–1900, Nov. 2013.
[19] W. Bi and J. T. Kwok, "Mandatory leaf node prediction in hierarchical multilabel classification," *Adv. Neural Inf. Process. Syst.*, vol. 25, no. 12, pp. 2275–2287, 2014.
[20] Q. Wu, Y. Ye, H. Zhang, T. Chow, and S. Ho, "Ml-tree: A tree-structure-based approach to multilabel learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 430–443, 2015.
[21] Y. Luo, D. Tao, C. Xu, C. Xu, H. Liu, and Y. Wen, "Multiview vector-valued manifold regularization for multilabel image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 5, pp. 709–722, May 2013.
[22] S. Chen, X. Yang, and Y. Tian, "Discriminative hierarchical K-means tree for large-scale image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 9, pp. 2200–2205, Sep. 2015.
[23] X. Cao, C. Zhang, H. Fu, X. Guo, and Q. Tian, "Saliency-aware nonparametric foreground annotation based on weakly labeled data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 6, pp. 1253–1265, Jun. 2016.
[24] F. Charte, A. Rivera, M. del Jesus, and F. Herrera, "Li-mlc: A label inference methodology for addressing high dimensionality in the label space for multilabel classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 10, pp. 1842–1854, Aug. 2014.
[25] D. Bouzas, N. Arvanitopoulos, and A. Tefas, "Graph embedded non-parametric mutual information for supervised dimensionality reduction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 951–963, May 2015.
[26] R. Mall and J. A. K. Suykens, "Very sparse LSSVM reductions for large-scale data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 1086–1097, May 2015.
[27] J. Wu and H. Yang, "Linear regression-based efficient SVM learning for large-scale classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2357–2369, Oct. 2015.
[28] Z. Lai, W. K. Wong, Y. Xu, J. Yang, and D. Zhang, "Approximate orthogonal sparse embedding for dimensionality reduction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 723–735, Apr. 2016.
[29] Y. Yu, K. I. Diamantaras, T. McKelvey, and S. Y. Kung, "Class-specific subspace kernel representations and adaptive margin slack minimization for large scale classification," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published.
[30] D. Hsu, S. Kakade, J. Langford, and T. Zhang, "Multi-label prediction via compressed sensing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 772–780.
[31] F. Tai and H.-T. Lin, "Multilabel classification with principal label space transformation," *Neural Comput.*, vol. 24, no. 9, pp. 2508–2542, 2012.
[32] T. Zhou, D. Tao, and X. Wu, "Compressed labeling on distilled labelsets for multi-label learning," *Mach. Learn.*, vol. 88, nos. 1–2, pp. 69–126, 2012.
[33] Y. Chen and H. Lin, "Feature-aware label space dimension reduction for multi-label classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1529–1537.
[34] J. Wicker, B. Pfahringer, and S. Kramer, "Multi-label classification using Boolean matrix decomposition," in *Proc. Annu. ACM Symp. Appl. Comput.*, 2012, pp. 179–186.
[35] W. Bi and J. Kwok, "Efficient multi-label classification with many labels," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 405–413.
[36] Z. Lin, G. Ding, M. Hu, and J. Wang, "Multi-label classification via feature-aware implicit label space encoding," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 325–333.
[37] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Effective and efficient multilabel classification in domains with large number of labels," in *Proc. ECML/PKDD Workshop Mining Multidimensional Data*, 2008, pp. 30–44.
[38] O. Dekel and O. Shamir, "Multiclass-multilabel classification with more classes than examples," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2010, pp. 137–144.
[39] K. Balasubramanian and G. Lebanon, "The landmark selection method for multiple output prediction," in *Proc. Int. Conf. Mach. Learn.*, 2012, pp. 1–2.
[40] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comput. Harmon. Anal.*, vol. 26, no. 3, pp. 301–321, 2009.
[41] I. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Springer-Verlag, 1986.
[42] G. Tsoumakas and I. Katakis, "Multi label classification: An overview," *Int. J. Data Warehous. Min.*, vol. 3, no. 3, pp. 1–13, 2007.
[43] Y. Zhang and J. Schneider, "Multi-label output codes using canonical correlation analysis," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 873–882.
[44] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1990.
[45] R. B. Lehoucq and D. C. Sorensen, "Deflation techniques for an implicitly restarted Arnoldi iteration," *SIAM. J. Matrix Anal. Appl.*, vol. 17, no. 4, pp. 789–821, 1996.
[46] M. Hu, Y. Chen, and J. T. Y. Kwok, "Building sparse multiple-kernel SVM classifiers," *IEEE Trans. Neural Netw.*, vol. 20, no. 5, pp. 827–839, May 2009.
[47] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "Mulan: A java library for multi-label learning," *J. Mach. Learn. Res.*, vol. 12, pp. 2411–2414, Jun. 2011.
[48] L. Von Ahn and L. Dabbish, "Labeling images with a computer game," in *Proc. Conf. Human Factors Comput. Syst.*, 2004, pp. 319–326.
[49] M. Lux and S. A. Chatzichristofis, "Lire: Lucene image retrieval: An extensible java CBIR library," in *Proc. 16th ACM Int. Conf. Multimedia*, 2008, pp. 1085–1088.
[50] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker, "Multi-label classification via calibrated label ranking," *Mach. Learn.*, vol. 73, no. 2, pp. 133–153, 2008.
[51] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.
[52] M. Zhang and Z. Zhou, "A review on multi-label learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1819–1837, Aug. 2014.
[53] J. McDonald, *Handbook of Biological Statistics*. Baltimore, MD, USA: Sparky House, 2014.
[54] G. Ding, Y. Guo, J. Zhou, and Y. Gao, "Large-scale cross-modality search via collective matrix factorization hashing," *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 5427–5440, Jun. 2016.
[55] X. Lu, X. Zheng, and X. Li, "Latent semantic minimal hashing for image retrieval," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 355–368, Jul. 2017.

Zijia Lin (S'13) received the B.Sc. degree from the School of Software, Tsinghua University, Beijing, China, in 2011, and the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, in 2016.

His current research interests include multimedia information retrieval and machine learning.

Guiguang Ding (M'13) received the Ph.D. degree in electronic engineering from Xidian University, Xi'an, China.

He is currently an Associate Professor with the School of Software, Tsinghua University, Beijing, China. Prior to joining the School of Software in 2006, he had been a Post-Doctoral Research Fellow with the Department of Automation, Tsinghua University. He has authored or co-authored over 70 referred journal and conference papers in the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, *Computer Vision and Image Understanding*, *ACM Transactions on Intelligent Systems and Technology*, the International Conference on Computer Vision, the Conference on Computer Vision and Pattern Recognition, the International Conference on Machine Learning, the International Joint Conference on Artificial Intelligence, and Association for the Advancement of Artificial Intelligence, and applied for over 20 Patents.

Jungong Han is currently a Senior Lecturer with the Department of Computer Science and Digital Technologies, Northumbria University, Newcastle upon Tyne, U.K. His current research interests include multimedia content identification, multisensor data fusion, computer vision, and multimedia security.

Dr. Han is an Associate Editor of *Neurocomputing* (Elsevier), and an Editorial Board Member of *Multimedia Tools and Applications* (Springer).

Ling Shao (M'09–SM'10) was a Senior Scientist with Philips Research, Eindhoven, The Netherlands, from 2005 to 2009, a Senior Lecturer with The University of Sheffield, Sheffield, U.K., from 2009 to 2014, a Professor with Northumbria University, Newcastle upon Tyne, U.K., from 2014 to 2016. He is currently a Professor with the School of Computing Sciences, University of East Anglia, Norwich, U.K. His current research interests include computer vision, image/video processing, and machine learning.

Prof. Shao is a fellow of the British Computer Society and the Institution of Engineering and Technology. He is an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and several other journals.