ABSTRACT

Title of dissertation: SPARSE AND DEEP REPRESENTATIONS FOR FACE RECOGNITION AND OBJECT DETECTION

Hongyu Xu
Doctor of Philosophy, 2019

Dissertation directed by: Professor Rama Chellappa
Department of Electrical and Computer Engineering

Face recognition and object detection are two very fundamental visual recognition applications in computer vision. How to learn good feature representations using machine learning has become the cornerstone of perception-based systems. A good feature representation is often the one that is robust and discriminative to multiple instances of the same category. Starting from features such as intensity, histogram etc. in the image, followed by hand-crafted features, to the most recent sophisticated deep feature representations, we have witnessed the remarkable improvement in the ability of a feature learning algorithm to perform pattern recognition tasks such as face recognition and object detection. One of the conventional feature learning methods, dictionary learning has been proposed to learn discriminative and sparse representations for visual recognition. These dictionary learning methods can learn both representative and discriminative dictionaries, and the associated sparse representations are effective for vision tasks such as face recognition. More recently, deep features have been widely adopted by the computer vision community owing

to the powerful deep neural network, which is capable of distilling information from high dimensional input spaces to a low dimensional semantic space. The research problems which comprise this dissertation lie at the cross section of conventional feature and deep feature learning approaches. Thus, in this dissertation, we study both sparse and deep representations for face recognition and object detection.

First, we begin by studying the topic of spare representations. We present a simple thresholded feature learning algorithm under sparse support recovery. We show that under certain conditions, the thresholded feature exactly recovers the nonzero support of the sparse code. Secondly, based on the theoretical guarantees, we derive the model and algorithm named Dictionary Learning for Thresholded Features (DLTF), to learn the dictionary that is optimized for the thresholded feature. The DLTF dictionaries are specifically designed for using the thresholded feature at inference, which prioritize simplicity, efficiency, general usability and theoretical guarantees. Both synthetic simulations and real-data experiments (*i.e.* image clustering and unsupervised hashing) verify the competitive quantitative results and remarkable efficiency of applying thresholded features with DLTF dictionaries.

Continuing our focus on investigating the sparse representation and its application to computer vision tasks, we address the sparse representations for unconstrained face verification/recognition problem. In the first part, we address the video-based face recognition problem since it brings more challenges due to the fact that the videos are often acquired under significant variations in poses, expressions, lighting conditions and backgrounds. In order to extract representations that are robust to these variations, we propose a structured dictionary learning framework.

Specifically, we employ dictionary learning and low-rank approximation methods to preserve the invariant structure of face images in videos. The learned structured dictionary is both discriminative and reconstructive. We demonstrate the effectiveness of our approach through extensive experiments on three video-based face recognition datasets.

Recently, template-based face verification has gained more popularity. Unlike traditional verification tasks, which evaluate on image-to-image or video-to-video pairs, template-based face verification/recognition methods can exploit training and/or gallery data containing a mixture of both images or videos from the person of interest. In the second part, we propose a regularized sparse coding approach for template-based face verification. First, we construct a reference dictionary, which represents the training set. Then we learn the discriminative sparse codes of the templates for verification through the proposed template regularized sparse coding approach. Finally, we measure the similarity between templates.

However, in real world scenarios, training and test data are sampled from different distributions. Therefore, we also extend the dictionary learning techniques to tackle the domain adaptation problem, where the data from the training set (source domain) and test set (target domain) have different underlying distributions (domain shift). We propose a domain-adaptive dictionary learning framework to model the domain shift by generating a set of intermediate domains. These intermediate domains bridge the gap between the source and target domains. Specifically, we not only learn a common dictionary to encode the domain-shared features but also learn a set of domain specific dictionaries to model the domain shift. This separation

enables us to learn more compact and reconstructive dictionaries for domain adaptation. The domain-adaptive features for recognition are finally derived by aligning all the recovered feature representations of both source and target along the domain path. We evaluate our approach on both cross-domain face recognition and object classification tasks.

Finally, we study another fundamental problem in computer vision: generic object detection. Object detection has become one of the most valuable pattern recognition tasks, with great benefits in scene understanding, face recognition, action recognition, robotics and self-driving vehicles, etc. We propose a novel object detector named "Deep Regionlets" by blending deep learning and the traditional regionlet method. The proposed framework "Deep Regionlets" is able to address the limitations of traditional regionlet methods, leading to significant precision improvement by exploiting the power of deep convolutional neural networks. Furthermore, we conduct a detailed analysis of our approach to understand its merits and properties. Extensive experiments on two detection benchmark datasets show that the proposed deep regionlet approach outperforms several state-of-the-art competitors.

SPARSE AND DEEP REPRESENTATIONS FOR FACE RECOGNITION AND OBJECT DETECTION

by

Hongyu Xu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Professor Rama Chellappa, Chair/Advisor
Professor Joseph F. JaJa
Professor Behtash Babadi
Dr. Carlos D. Castillo
Professor Dinesh Manocha (Dean's Representative)

© Copyright by
Hongyu Xu
2019

Dedication

To my beloved parents, grandparents & family

For their love, endless support, understanding, encouragement & sacrifices

Acknowledgments

My deep gratitude goes to all who have made this dissertation possible and because of whom my Ph.D. experience has been one that I will cherish forever.

First and foremost I would like to thank my advisor, Professor Rama Chellappa, without whom this dissertation will not be possible. I have been unbelievably fortunate to have an advisor who gave me the freedom to explore on my own, and at the same time the encouragement and support to fight when my steps faltered. I will always remember his words "play bold and reach the top.", which will inspire me in the future. His dedication, discipline and passion towards his career and his deep philosophic words about life has greatly influenced me. I feel honored to have worked with such a fantastic professor, a caring advisor and also an admirable individual.

I am honored to have Professor Joseph JaJa, Professor Dinesh Manocha, Professor Behtash Babadi and Dr. Carlos D. Castillo in my dissertation committee. I am thankful to them for serving in my committee, sparing their valuable time reviewing the manuscript and providing insightful and diverse feedback.

I would like to thank my mentors at Snap Research. Xutao Lv and Xiaoyu Wang have played a pivotal role in my research career, especially the shout out goes to Xutao. Xutao took the pain of explaining every tiny detail in object detection to me and I would always be grateful to him for that. Their support was extremely important as it shaped the latter part of my PhD research.

I would like to thank all the members of Prof. Chellappa's group for the

friendly atmosphere, enjoyable group lunches, and helpful discussions, with a special mention of Jingjing Zheng, who helped me to appreciate the rigors of graduate studies. I benefited greatly from collaborating with her. Her understanding of paper writing and presentation has been one of the most important reasons that my papers were accepted.

Friends helped my stay through difficult times and my PhD life would not be complete without their friendship. I am grateful to have my best friends, Xin Zhang and Yali Wan, for their unconditional love and support. I am more than lucky to have you in my life.

My research was supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA R&D Contract No. 2014-14071600012 and Department of Interior/Interior Business Center(DOI/IBC) contract number D17PC00345. I appreciate the funding support for my research work.

Last, I owe my deepest thanks to my family - my parents and my grandparents, who have always stood by me and guided me through my life, and have pulled me through against impossible odds at times. Words cannot express the gratitude I owe them.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

Face recognition and object detection are two very fundamental visual recognition applications in computer vision. How to learn good feature representations using machine learning, has become the cornerstone of perception-based systems. A good feature representation is often the one that is robust and discriminative to multiple instances of the same category. Starting from primitive features such as pixel intensities or histograms, followed by hand-crafted features such as Histogram of Gradients (HOG) [45], Scale Invariant Feature Transform (SIFT) [126, 127] and Speeded-Up Robust Features (SURP) [13], to the sophisticated deep feature representations, we have witnessed remarkable performance improvements in tasks such as face recognition and object detection.

Motivated by sparse coding theory, dictionary learning has been proposed to learn discriminative and sparse representations for visual recognition [1, 156, 212]. These dictionary learning methods can learn both representative and discriminative dictionaries, and the associated sparse representations have been deployed for tasks such as face recognition. More recently, deep features have been widely adopted by the computer vision community owing to the powerful deep neural network, which

is capable of distilling information from high dimensional input spaces to a low dimensional semantic space.

The research problems which comprise this dissertation lie at the cross section of conventional feature learning and deep feature learning. Thus, in this dissertation, we study both sparse and deep representations for face recognition and object detection.

1.2 Proposed Approaches and Contributions

In this section, we briefly describe the problem addressed in this dissertation along with robust solutions to address them.

1.2.1 Learning Simple Thresholded Features with Sparse Support Recovery

We begin this dissertation by studying the topic of spare representations in Chapter 2. We present a simple thresholded feature learning algorithm under sparse support recovery. First, we show that under certain conditions, the thresholded feature exactly recovers the nonzero support of the sparse code. The support recovery is a core problem in sparse signal recovery: if the nonzero support set is correctly identified, sparse representation can be obtained using least squares method. Moreover, the support itself makes a useful feature in certain scenarios, such as quantization and hashing. Second, based on the theoretical guarantees, we derive an algorithm named Dictionary Learning for Thresholded Features (DLTF), to learn

the dictionary that is optimized for the thresholded feature. The DLTF dictionaries are specifically designed for using the thresholded feature at inference, which prioritize simplicity, efficiency, general usability and theoretical guarantees. Last but not least, we derive a novel efficient $O(m \log m)$ algorithm for the $(k, 2)$ norm proximal subproblem. Both synthetic simulations and real-data experiments (*i.e.* image clustering and unsupervised hashing) verify the competitive quantitative results and remarkable efficiency of applying thresholded features with DLTF dictionaries.

1.2.2 Sparse Representations for Face Recognition

Continuing our focus on investigating the sparse representation and its application to computer vision, we consider the sparse representations for the unconstrained face verification/recognition problem. In the first part of Chapter 3, we consider the video-based face recognition problem. Given a video sequence, we aim to recognize the subject in the video. While video provides more samples from frames containing the person of interest, it brings more challenges as different video sequences of the same subject may contain great variations in resolution, illumination, pose and facial expressions. Therefore, it is important to represent and model the same subject against these intra-person variations. To address this, we propose a structured dictionary learning framework for video-based face recognition. The learned dictionary has the following advantages. We employ dictionary learning and low-rank approximation methods to preserve the invariant structure of face images in videos. The learned dictionary is both discriminative and reconstructive. Thus, we

not only minimize the reconstruction error of all the face images but also encourage a sub-dictionary to represent the corresponding subject from different videos. Moreover, by introducing the low-rank approximation, the proposed method is able to discover invariant structured information from different videos of the same subject. To this end, an efficient alternating algorithm is employed to learn the structured dictionary. Experimental results on the three benchmark video-based face recognition datasets show that the proposed framework yields favorable performance over state-of-the-art methods.

In the second part of Chapter 3, we study the template-based face verification problem. Recently, template-based face recognition has gained more popularity in computer vision community. A **template** is a mixture of different media data such as a single image or an image set or video clips containing the person of interest. The notion of a template is useful in real world applications because it provides more flexibility and longitudinal access control of data from subjects. Different from the traditional face verification scenario which verifies whether two images or videos in a pair belong to the same subject as in Labeled Face in the Wild dataset [91] or YouTube Face dataset [193], template-based face verification evaluates the pair over templates as introduced in [103]. It is noted that the performance may deteriorate if we treat a template as the image set containing the independent samples and apply existing dictionary-based methods [81]. To overcome such limitations, we propose a regularized sparse coding approach for template-based unconstrained face verification. The proposed approach adapts to training and gallery data using three steps. First, we construct a reference dictionary, which represents

the training set efficiently. Then we learn the discriminative sparse codes of the templates for verification using the template regularized sparse coding approach. Finally, we measure the similarity between templates by computing the reference score and template adaptive scores. An efficient algorithm is also employed to learn the template regularized sparse codes. Extensive experiments on template-based verification benchmark dataset JANUS IJB-A [103] demonstrate that the proposed approach outperforms several recent literatures.

1.2.3 Cross-domain Visual Recognition via Domain Adaptive Dictionary Learning

In Chapter 4, we consider scenarios in which the training data (**source domain**) and test data (**target domain**) are sampled from different underlying distribution. For instance, training and testing images may be acquired under different environments, viewpoints and illumination conditions in application such as face recognition, object classification, human detection and video concept detection. This is known as the domain adaptation problem. Furthermore, we focus on the more challenging unsupervised settings where the samples in the target domain are unlabeled. It would be also highly desirable for recognition systems to automatically adapt to a different domain without any additional labeling efforts.

We propose a novel domain-adaptive dictionary learning approach to generate a set of intermediate domains which bridge the gap between source and target domains. Our approach learns two types of dictionaries: a common dictionary

and a domain-specific dictionary. The common dictionary shared by all domains is used to extract domain-shared features, whereas the domain-specific dictionary which is incoherent to the common dictionary models the domain shift. The separation of the common dictionary from domain-specific dictionary enables us to learn more compact and reconstructive dictionaries for deriving domain-adaptive features. Meanwhile, our approach gradually recovers the feature representations of both source and target data along the domain path. Final domain adaptive features are derived by aligning all the recovered domain data. Extensive experiments on cross-domain face recognition and object classification show that the proposed approach significantly outperforms state-of-the-art methods.

1.2.4 Deep Regionlets: Blended Representation and Deep Learning for Generic Object Detection

Finally in Chapter 5, we study another fundamental problem in computer vision: generic object detection. Object detection has become one of the most valuable pattern recognition tasks, with applications in scene understanding, face recognition, action recognition, robotics and self-driving vehicles, etc. In this chapter, we propose a novel object detector named "**Deep Regionlets**" by blending deep learning and the traditional regionlet method [185, 226]. The proposed framework "Deep Regionlets" is able to address the limitations of traditional regionlet methods, leading to significant precision improvement by exploiting the power of deep convolutional neural networks. More specifically, we design a region selection

network, which **first** performs **non-rectangular** regions selection within the detection bounding box generated from a detection window proposal. It provides more flexibility in modeling objects with variable shapes and deformable parts. We also propose a deep regionlet learning module, including feature transformation and a gating network. The gating network serves as a soft regionlet selector and lets the network focus on features that benefit detection performance. Furthermore, we conduct a detailed analysis of our approach to understand its merits and properties. Extensive experiments on two detection benchmark datasets, PASCAL VOC [58] and Microsoft COCO [118] show that the proposed deep regionlet approach outperforms several state-of-the-art competitors.

1.3 Organization

The rest of the dissertation is organized as follows. In Chapter 2, we present a simple thresholded feature learning algorithm under spare support recovery. In the first part of Chapter 3, we present a structured dictionary learning method for video-based face recognition. In the second part, we present a template regularized sparse coding framework to address the template-based face verification problem. In Chapter 4, we consider the domain adaption problem where data from the training set and test set have different underlying distributions. We propose a domain adaptive dictionary learning method to bridge the domain shift. We address the generic object detection problem in Chapter 5 and propose a novel object detector which blends deep learning and the traditional regionlet method. Finally, in Chapter 6,

we conclude the dissertation and discuss possible future research directions.

Chapter 2: Learning Simple Thresholded Features with Sparse Support Recovery

2.1 Introduction

Let $\Omega_k = \{\boldsymbol{z} \in \mathbb{R}^m : ||\boldsymbol{z}||_0 \leq k\}$. For a data sample $\boldsymbol{x} \in \mathbb{R}^n$, the *sparse coding* technique [1] aims to find the sparse code $\boldsymbol{z} \in \Omega_k$ to represent $\boldsymbol{x}$ compactly, i.e., $\boldsymbol{x} \approx W\boldsymbol{z}$, using a dictionary $W \in \mathbb{R}^{n\times m} = [\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_m]$, where each atom $\boldsymbol{w}_i \in \mathbb{R}^n$ is assumed to have unit ℓ_2-norm, $i = 1, ..., m$, to avoid scale ambiguity. With a properly designed or learned W, sparse coding is known to be powerful in numerous reconstruction or discriminative tasks such as signal sensing, classification and clustering [12, 37, 94, 148, 172, 189, 200, 201, 214, 216] . One crucial drawback of sparse coding lies in its prohibitive cost of computing the sparse code at test time, which calls for iterative greedy or convex optimization algorithms [17, 57]. This drawback limits the applicability of sparse codes in large-scale, high-dimensional problems, or when nearly real-time processing is desired.

Among a few fast sparse coding approximations, the simplest choice is arguably the *thresholded feature* [31, 49, 59, 136, 171]:

$$\bar{\boldsymbol{z}} := \max_k(W^\top \boldsymbol{x}), \tag{2.1}$$

where $\max_k$ retains the k largest-magnitude entries while setting others to zero[1]. The threshold feature $\bar{\boldsymbol{z}}$ also belongs to Ω_k, and is extremely efficient and easy to implement as it involves only a matrix-vector multiplication and a $\max_k$ operation. [38,39] showed that such a simple encoding displays remarkable discriminative ability, and can often achieve comparable results to standard sparse coding, provided that the number of labeled samples and the dictionary size are large enough. [59] pointed out that the thresholded feature corresponds to an inexact approximation of sparse coding, where only one iteration of proximal gradient algorithm is unfolded. One may also notice the interesting resemblance of (2.1) to a standard linear fully connected layer plus neurons in deep learning [107], on which we will discuss more later.

While dictionary learning [10, 59, 133, 139, 148, 154, 155, 157, 200, 201] has been well developed for standard sparse coding [1], the choice of W remains relatively unexplored for the thresholded feature. [49] used standard dictionaries, leading to a fairly rough approximation to the exact iterative solution, with sub-optimal results. [59] relied on supervised joint training to learn W, which is similar to learning a single-layer neural network classifier and does not generalize to unsupervised feature learning. Moreover, it is unclear how "roughly" $\bar{\boldsymbol{z}}$ approximates $\boldsymbol{z}$, and in what sense the former can be treated as a reliable substitute for the latter, leaving the effectiveness and robustness of threshold features under question.

This chapter answers the above questions. Firstly, we show that under certain conditions, the thresholded feature $\bar{\boldsymbol{z}}$ exactly recovers the nonzero support of the

[1] [59] exploited a "soft" version of the thresholded feature, which can be analyzed similarly.

sparse code $\boldsymbol{z}$. The support recovery is a core problem in sparse signal recovery [210]: if the nonzero support set is correctly identified, $\boldsymbol{z}$ can be obtained using the least squares method. Moreover, the support itself makes a useful feature in certain scenarios, such as quantization and hashing [37].

Secondly, based on the theoretical guarantees, we derive the model and algorithm for *Dictionary Learning for Thresholded Features* (DLTF), to learn the dictionary that is optimized for the thresholded feature. It is important to note that DLTF is not "yet another" way of *standard dictionary learning*, whose inference relies on iterative sparse solvers. Instead, it is a new type of dictionary learning, specifically designed for using the thresholded feature (2.1) at inference, which prioritize (extreme) simplicity, efficiency, general usability and theoretical guarantees.

Last but not least, in particular, we derive a novel efficient $O(m \log m)$ algorithm for the $(k, 2)$ norm proximal subproblem. Both synthetic simulations and real-data experiments (*i.e.* image clustering and unsupervised hashing) verify the competitive quantitative results and demonstrate the remarkable efficiency of applying thresholded features with DLTF dictionaries.

2.2 Support Recovery Guarantees for Thresholded Features

We present two support recovery guarantees, called the *weak* and the *strong* recovery guarantees respectively. The weak recovery guarantee depends on *mutual incoherence* [51], and the strong recovery guarantee takes advantage of the *Restricted Isometry Property* (RIP) [22]. The two guarantees are called "weak" and "strong"

respectively because of different sample complexity requirements: based on the results of [214], to uniformly recover $\boldsymbol{z} \in \Omega_k$, the former requires $\mathcal{O}(k^2 \ln m)$ samples, while the latter gives rise to the lower sample complexity of $\mathcal{O}(k \ln m)$.

The two guarantees can be derived using classical techniques underlying the compressive sensing theory [63] and iterative thresholding algorithm [17, 69, 73]. *Full proof is detailed in appendices A and B.* It is noted that our goal here is not to provide a tighter bound than what is now available in the literature (*i.e.*, Theorem 5.16 of [63]), but rather to illustrate what factors or quantities will affect the support recovery in the special case of thresholded features. We will further discuss how these guarantees motivate the proposed DLTF model in the next section.

2.2.1 The Weak Recovery Guarantee

For a simple "noiseless" model: $\boldsymbol{x} = W\boldsymbol{z}$, without loss of generality, we assume that the non-zero entries of $\boldsymbol{z}$ are sorted by absolute magnitude in a decreasing order: $|z_1| \geq |z_2| ... \geq |z_k|$. We denote by $\text{supp}(\boldsymbol{z}) \in \mathbb{B}^m$ the sparse support of $\boldsymbol{z}$, i.e., its i-th entry is 1 if z_i is nonzero. We define the mutual incoherence [51] of W: $\mu_W = \max_{i \neq j} |\langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle|$.

Theorem 1. *If the sufficient condition* $k\mu_W \leq \frac{|z_k|}{2|z_1|}$ *holds, then* $supp(\boldsymbol{z}) = supp(\bar{\boldsymbol{z}})$, *where* $\bar{\boldsymbol{z}} := max_k(W^\top \boldsymbol{x})$, .

A special case when $\boldsymbol{z} \in \{0, 1\}^k$ follows immediately (the sgn indicator function is defined to yield output 1 when the input is nonzero, and 0 elsewhere):

Corollary 1.1 Assume $\boldsymbol{x} = W\boldsymbol{z}$, $\boldsymbol{z} \in \mathbb{B}^m$ and $||\boldsymbol{z}||_0 \leq k$. If $\mu_W \leq \frac{1}{2k}$, then $\boldsymbol{z}$

$= \text{sgn}(\bar{\boldsymbol{z}})$.

Theorem 1 can be further (loosely) extended to noisy case, when $\boldsymbol{x}$ is corrupted by the noise $\boldsymbol{e}$: $\boldsymbol{x} = W\boldsymbol{z} + \boldsymbol{e}$. Denote the mutual coherence between the dictionary and the noise: $\mu_e = \max_i |\langle \boldsymbol{w}_i, \boldsymbol{e} \rangle|$, $\boldsymbol{e}$ may follow any statistical distribution only if μ_e can be properly bounded.

Corollary 1.2 Assume $\boldsymbol{x} = W\boldsymbol{z} + \boldsymbol{e}$. If $k\mu_W \leq \frac{|z_k|}{2|z_1|} - \frac{\mu_e}{|z_1|}$, then $\text{supp}(\boldsymbol{z}) = \text{supp}(\bar{\boldsymbol{z}})$.

The noisy-case upper bound on $k\mu_W$ turns out to be close to the noiseless bound, if the magnitudes of all nonzero entries in $\boldsymbol{z}$ increase proportionally, so that $\frac{|z_k|}{2|z_1|}$ remains unchanged but $\frac{\mu_e}{|z_1|}$ vanishes. This is equivalent to improving the signal-to-noise ratio (SNR) of the input signal.

2.2.2 The Strong Recovery Guarantee

Recall that the RIP [22] condition assumes:

Assumption 1. *For* $\forall \boldsymbol{z} \in \Omega_k$, *there exists* $\delta_W \in (0, 1)$ *s.t.* $(1 - \delta_W) \leq \frac{||Wz||^2}{||z||^2} \leq (1 + \delta_W)$.

We further introduce the stronger guarantee form:

Theorem 2. *Assume* $\boldsymbol{x} = W\boldsymbol{z} + \boldsymbol{e}$, $\bar{\boldsymbol{z}} = max_k(W^\top \boldsymbol{x})$, $\boldsymbol{z} \in \Omega_k$. *If* $\delta_W \in (0, 1 - \frac{\sqrt{3}}{2})$, *then* $supp(\boldsymbol{z}) = supp(\bar{\boldsymbol{z}})$, *given that the smallest nonzero element in* $\boldsymbol{z}$ *is large enough:*

$$|z_k| \geq 2\sqrt{2\delta_W - \delta_W^2}||\boldsymbol{z}||_2 + 2\left\|max_{2\text{k}}(W^\top \boldsymbol{e})\right\|. \tag{2.2}$$

When condition (2.2) holds, a lower bound for $||\boldsymbol{z}||$ can also be derived as:

$$||\boldsymbol{z}|| \geq \frac{2\sqrt{k}}{1-2\sqrt{k(2\delta_W-\delta_W^2)}} \left\|\max_{2k}(W^\top \boldsymbol{e})\right\|. \tag{2.3}$$

To ensure $1-2\sqrt{k(2\delta_W-\delta_W^2)}>0$, k must be less than $\frac{1}{4(2\delta_W-\delta_W^2)}$. Consistent with the weak guarantee case, (2.2) and (2.3) also encourage small δ_W and k, uncorrelated small noise and high SNR. Different from Theorem 1, Theorem 2 enforces no extra requirement on the absolute magnitudes of the nonzero entries in $\boldsymbol{z}$ except for a lower bound. It is also noted that Theorem 2 is stricter than original RIP [22] constraint.

2.3 DLTF: A Dictionary Learning Model for Thresholded Features

2.3.1 Model Formulation

As illustrated by Theorems 1 and 2, in order to achieve perfect support recovery in the simple thresholded feature, two crucial points are (at least) required in addition to the sparsity of $\boldsymbol{z}$: 1) W has small μ_W and/or δ_W; 2) the residual $\boldsymbol{e}$ is small and nearly uncorrelated with W. Taking them into account, we design the *Dictionary Learning model for Thresholded Features*. Specifically, we follow [55] to encourage the Gram matrix of W to be close to the identity by minimizing $||W^\top W-I||^2$, which enforces W to have small μ_W or δ_W. Moreover, Theorem 2 suggests to minimize $\left\|\max_{2k}(W^\top \boldsymbol{e})\right\|$. It is noted that $||\max_k(.)||$ is a convex, sub-differentiable vector norm, under the name of $(k,2)$ symmetric gauge norm [15], or $(k,2)$ norm for short [174]. Thus we re-write $||\max_{2k}(W^\top \boldsymbol{e})||^2$ as $||W^\top \boldsymbol{e}||_{2k,2}^2$ in what

follows.

Let $X \in \mathbb{R}^{n\times N} = \{\boldsymbol{x}_i\}$ be the training set, and $Z \in \mathbb{R}^{m\times N} = \{\boldsymbol{z}_i\}$ be the corresponding sparse codes. The proposed DLTF approach is to learn W with the following properties: (1) X can be well approximated by WZ; (2) $\forall i$, supp($\boldsymbol{z}_i$) and supp($\max_k(W^\top \boldsymbol{x}_i)$) are as close as possible, which is achieved through Theorems 1, 2 by minimizing $||W^\top W - I||^2$ and $||W^\top \boldsymbol{e}||^2_{2k,2}$. In order to achieve the above goal, we formulate the objective function as follows:

$$\begin{aligned} \min_{W,Q,Z} \frac{\lambda}{2}\sum_{i=1}^{N} ||\boldsymbol{q}_i||^2_{2k,2} + ||W^\top W - I||^2 + \frac{\theta}{2}\|X - WZ\|^2 \\ s.t.\ Q = W^\top (X - WZ); \\ ||\boldsymbol{z}_i||_0 \leq k, , i = 1, 2, ..., N; \\ ||\boldsymbol{w}_j|| = 1, j = 1, ..., m. \end{aligned} \tag{2.4}$$

where $Q = W^\top(X - WZ) \in \mathbb{R}^{m\times N}$ and $\boldsymbol{q}_i$(i = 1, ..., N) is the i-th column. λ, θ are two scalars. We introduce Q to reduce the complexity of W since W is involved in both the non-smooth $(2k, 2)$ norm term and the diagonal penalty term simultaneously.

2.3.2 Algorithm Development

We apply the optimization framework of Alternating Direction Method of Multipliers (ADMM). The augmented Lagrangian function of (2.4) is:

Algorithm 1: Algorithm to solve proximal mapping (2.8) for ordered and positive vector.

Input: Vector $\boldsymbol{c} \in \mathbb{R}^m, \boldsymbol{c} \geq \mathbf{0}$ and $\boldsymbol{c}$ is in increasing order, parameter $\gamma \geq 0$.

Result: Problem solution $\boldsymbol{p}^*$.

$$\mathbf{u}_{1:m-k} = \boldsymbol{c}_{1:m-k'}, \mathbf{u}_{m-k'+1:m} = \tfrac{1}{1+\gamma}\boldsymbol{c}_{m-k'+1:m};$$

$$\boldsymbol{t}_{1:m-k'} = 1, \boldsymbol{t}_{m-k'+1:m} = 1+\gamma;$$

$$\boldsymbol{p}^* = \text{Reduce}(\boldsymbol{u}, \boldsymbol{t}, 1);$$

`Reduce(`$\boldsymbol{u}, \boldsymbol{t}, j$`)`

Let J be the dimension of $\mathbf{u}$;

while $j \leq J$ **do**

if $\mathbf{u}_j > \mathbf{u}_{j+1}$ **then**

$\boldsymbol{u}' = [\boldsymbol{u}_{1:j-1}, \frac{\boldsymbol{t}_j\boldsymbol{u}_j + \boldsymbol{t}_{j+1}\boldsymbol{u}_{j+1}}{\boldsymbol{t}_j + \boldsymbol{t}_{j+1}}, \boldsymbol{u}_{j+2:end}]$; `// Remove` $\mathbf{u}_{j+1}$

$\boldsymbol{t}' = [\boldsymbol{t}_{1:j-1}, \boldsymbol{t}_j + \boldsymbol{t}_{j+1}, \boldsymbol{t}_{j+2:end}]$; `// Remove` $\boldsymbol{t}_{j+1}$

$\boldsymbol{x} = \text{Reduce}(\boldsymbol{u}', \boldsymbol{t}', \max(1, j-1))$; `// Recursively invoke Reduce`

return $[\mathbf{x}_{1:j}, \mathbf{x}_j, \mathbf{x}_{j+1:end}]$; `// Duplicate` $\mathbf{x}_j$ `since` $\mathbf{x}_j = \mathbf{x}_{j+1}$

end

$j = j + 1$;

end

return u

$$\begin{aligned}
&\frac{\lambda}{2}\sum_{i=1}^{N}||\boldsymbol{q}_i||_{2k,2}^2 + ||W^\top W - I||^2 + \frac{\theta}{2}\|X - WZ\|^2 \\
&+ \langle Y, Q - W^\top(X - WZ)\rangle + \frac{\beta}{2}||Q - W^\top(X - WZ)||^2 \\
&s.t.\,||\boldsymbol{z}_i||_0 \leq k, , i = 1, 2, ..., N; \\
&||\boldsymbol{w}_j|| = 1, j = 1, 2, ..., m.
\end{aligned} \tag{2.5}$$

where $Y \in \mathbb{R}^{m\times N}$ is the Lagrange multiplier and β is a positive constant. We then sequentially solve the three subproblems at the t-th iteration ($t = 0, 1, ...$).

2.3.2.1 Z-subproblem

Solving Z is a standard sparse decomposition problem, which can be solved separately for each $\boldsymbol{z}_i$ using the iterative algorithm [17]:

$$\begin{aligned}
Z_{t+1} = \arg\min_{Z}\ &\frac{\theta}{2}\|X - W_tZ\|^2 + \langle Y_t, W_t^\top W_tZ\rangle \\
&+ \frac{\beta}{2}||W_t^\top W_tZ - W_t^\top X + Q_t||^2 \\
s.t.\,&||\boldsymbol{z}_i||_0 \leq k, , i = 1, 2, ..., N.
\end{aligned} \tag{2.6}$$

2.3.2.2 Q-subproblem

The Q update could also be solved separately for each $\boldsymbol{q}_i$:

$$\begin{aligned}
Q_{t+1} = \arg\min_{Q}\ &\sum_{i=1}^{N}||\boldsymbol{q}_i||_{2k,2}^2 \\
&+ \frac{\beta}{\lambda}||Q - (W_t^\top X - W_t^\top W_tZ_{t+1} - \frac{Y_t}{\beta})||^2
\end{aligned} \tag{2.7}$$

Let $\gamma \geq 0$, and $\boldsymbol{q}, \boldsymbol{c} \in \mathbb{R}^m$. Define the proximal mapping of the $(k', 2)$ norm for $\boldsymbol{q}$:

$$prox_{\gamma}^{k',2}(\boldsymbol{c}) = \arg\min_{\boldsymbol{q}} \gamma||\boldsymbol{q}||_{k',2}^2 + ||\boldsymbol{q} - \boldsymbol{c}||^2 \tag{2.8}$$

Problem (2.7) is converted to solving the proximal mapping (2.8) with $\gamma = \frac{\lambda}{\beta}$ and $k' = 2k$. To our best knowledge, only the basic subgradient method [174] was exploited for the optimization of (k,2) norm in literature. We present an efficient $O(m \log m)$ solution for (2.7) as described in next section.

2.3.2.3 W-subproblem

The W update solves the following manifold constrained problem:

$$\begin{aligned} W_{t+1} = \arg\min_{W} \; & ||W^\top W - I||^2 + \frac{\theta}{2}\|X - WZ_{t+1}\|^2 \\ & - \langle Y, W^\top (X - WZ_{t+1})\rangle \\ & + \frac{\beta}{2}||Q_{t+1} - W^\top (X - WZ_{t+1})||^2 \\ & s.t. \; ||\boldsymbol{w}_j|| = 1, j = 1, 2, ..., m. \end{aligned} \tag{2.9}$$

We apply the curvilinear search algorithm in [192] to solve (2.9) as it lies in the spherical constraint.

Furthermore, Y is updated as: $Y_{t+1} = Y_t + \beta(Q_{t+1} - W_{t+1}^\top X + W_{t+1}^\top W_{t+1} Z_{t+1})$.

2.4 Efficient $O(m \log m)$ Proximal Mapping of (k, 2) Norm

It is noted that solving (2.8) with subgradient descent is yet inefficient. Therefore, we propose an efficient proximal algorithm for the (k, 2) norm.

Theorem 3. *The proximal mapping* (2.8) *is solved by Algorithm 1 in* $O(m \log m)$ *time complexity.*

Proof sketch: To prove Theorem 3, we first establish:

Lemma 4. *For the problem* (2.8) *with* $\boldsymbol{c} \geq \mathbf{0}$*, the order of coordinates in optimal solution* $\boldsymbol{q}^*$ *is the same as the order of the corresponding coordinates in* $\boldsymbol{c}$*.*

Lemma 4 shows that the proximal mapping (2.8) will not change the sign of $\boldsymbol{c}$, i.e., for all i, $\text{sign}(prox_{\gamma}^{k',2}(\boldsymbol{c})_i) = \text{sign}(c_i)$. Then we only need to consider the magnitude of entries in $\boldsymbol{c}$. We can sort the entries of $\boldsymbol{c}$ (in magnitude). Therefore, with additional time complexity $O(m \log m)$ for sorting, we can convert (2.8) with any vector $\boldsymbol{c} \in \mathbb{R}^m$ to the proximal mapping for ordered and positive vector.

We then introduce the following lemma:

Lemma 5. *Optimization problem*

$$\min_{\mathbf{x} \in \mathbb{R}^J} \sum_{j=1}^{J} \mathbf{t}_j (\mathbf{x}_j - \mathbf{u}_j)^2 \tag{2.10}$$

$$s.\ t.\ \mathbf{x}_1 \leq \mathbf{x}_2 \leq ... \leq \mathbf{x}_J \tag{2.11}$$

can be solved by invoking the subroutine "Reduce($\boldsymbol{u}$*,* $\boldsymbol{t}$*, 1)" in Algorithm 1.*

To solve (2.10), the key step is applying Lemma 6 stated below to iteratively merge neighbor variables to obtain a reduced problem. When the reduced problem has input $\mathbf{u}'$ containing monotonically increasing elements, the solution $\mathbf{x}' = \mathbf{u}'$.

Lemma 6. *If* $\mathbf{u}_j > \mathbf{u}_{j+1}$*, then the optimal solution* $\mathbf{x}^*$ *to* (2.10) *should satisfy* $\mathbf{x}_j^* = \mathbf{x}_{j+1}^*$.

We present the detailed proofs of Lemmas 4, 5, and 6 in the appendix.

2.5 Experiments

The main purpose of this section is to demonstrate that DLTF possesses the capability to learn the dictionary, that reliably recovers the sparse support (see synthetic experiments) and benefits the practical utilization of thresholded features most (see real data experiments). It is important to note that DLTF is a new type of dictionary learning, specifically designed for using the thresholded feature (2.1) at inference, which prioritize (extreme) simplicity, efficiency, general usability and theoretical guarantees. Despite not being optimized for any specific task, DLTF shows competitive performance for a variety of real-data tasks with minimal time costs.

We compare DLTF with standard dictionary learning algorithms, and choose the popular KSVD [1] as a representative of the latter in most experiments. In a few real-data experiments (e.g, clustering), we compare the dictionaries computed by DLTF and KSVD, and use them to compute both thresholded features, and canonical sparse codes (via iterative algorithms). **To avoid confusion**, such a comparison is intended as an "ablation study" (altering training and testing components) to show that: (1) DLTF dictionary is much better suited for the thresholded feature than conventional dictionaries; (2) the results of applying thresholded feature solved with DLTF dictionary are superior to or comparable with applying sparse features solved with more expensive iterative algorithms. It does not contradict our default pipeline of training DLTF dictionary and computing the thresholded features at inference.

	$k = 4$	$k = 6$	$k = 8$	$k = 10$	$k = 12$
original	0.354	0.778	1.241	1.746	2.293
random	3.876	5.721	7.499	9.204	10.895
KSVD	0.724	1.988	3.897	6.110	7.430
DLTF	0.495	1.119	1.879	2.753	3.759

Table 2.1: The support recovery performance comparison, at different sparsity levels k, measured by *ave_dif*. The DLTF results are reported with $\lambda = 0.05$, $\theta = 0.01$.

2.5.1 Support Recovery in Thresholded Features: Synthetic Simulations

We first evaluate the performance of the support recovery on synthetic data. We generate an over-complete i.i.d. random Gaussian matrix as the dictionary $W_0 \in \mathbb{R}^{n\times m}$, and the sparse codes $Z \in \mathbb{R}^{m\times N} = \{\boldsymbol{z}_i\}$, where each $\boldsymbol{z}_i$ has only k nonzero entries with the value 1 and random locations. We then synthesize $X \in \mathbb{R}^{n\times N} = \{\boldsymbol{x}_i\}$ by: $X = W_0 Z + E$, where each entry of the noise matrix E is i.i.d. sampled from $\mathcal{N}(0, 0.01)$. By default, we fix $n = 64$, $m = 128$, and $N = 10,\ 000$ for the training set. A testing set of $10,\ 000$ samples are generated separately.

In order to compute the thresholded features $\bar{\boldsymbol{z}}_i = \max_k(W^\top \boldsymbol{x}_i)$, we compare DLTF with three baselines: two are intentionally chosen to roughly indicate the empirical performance "upper bound" and "lower bound", as well as directly employing the conventional KSVD dictionary:

- *Original* baseline: $W = W_0$. Note that the random Gaussian W_0 itself has

very small mutual incoherence/RIP constant, and it is known that X can be sparsely represented over W_0. Therefore, we expect W_0 to be nearly an optimal solution to (2.4), and this original baseline's results will be likely close to the best attainable support recovery performance.

- *Random* baseline: $W = W_r$ which is another independent random Gaussian matrix.

- *KSVD* baseline: $W = W_K$ is learned from X by KSVD and then applied to thresholded features.

For DLTF, we denote $W = W_{\text{DLTF}}$ as solved from (2.4), using random initializations. It is noted that the original baseline is an "ideal" case that exists only in simulations. Except for a handful of cases such as compressive sensing [11], it is unlikely to have such a pre-known dictionary in practice, over which the target signals can be accurately represented, and whose mutual coherence/RIP constant is as small as the random bases.

To evaluate the accuracy of the recovered support, we define the following metric that calculates the averaged per-sample support differences between $\bar{\boldsymbol{z}}_i$ and $\boldsymbol{z}_i$:

$$ave\_dif = \frac{1}{N}\sum_{i=1}^{N}\frac{|\text{supp}(\bar{\boldsymbol{z}}_i) \oplus \text{supp},(\boldsymbol{z}_i)|_1}{2} \tag{2.12}$$

where $\oplus$ denotes the element-wise XOR operation. The value of *ave_dif* ranges between $[0, k]$.

We first vary the sparsity level $k \in [4, 6, 8, 10, 12]$, and compare the support recovery results in Table 2.1. It can be seen that DLTF achieves overall compa-

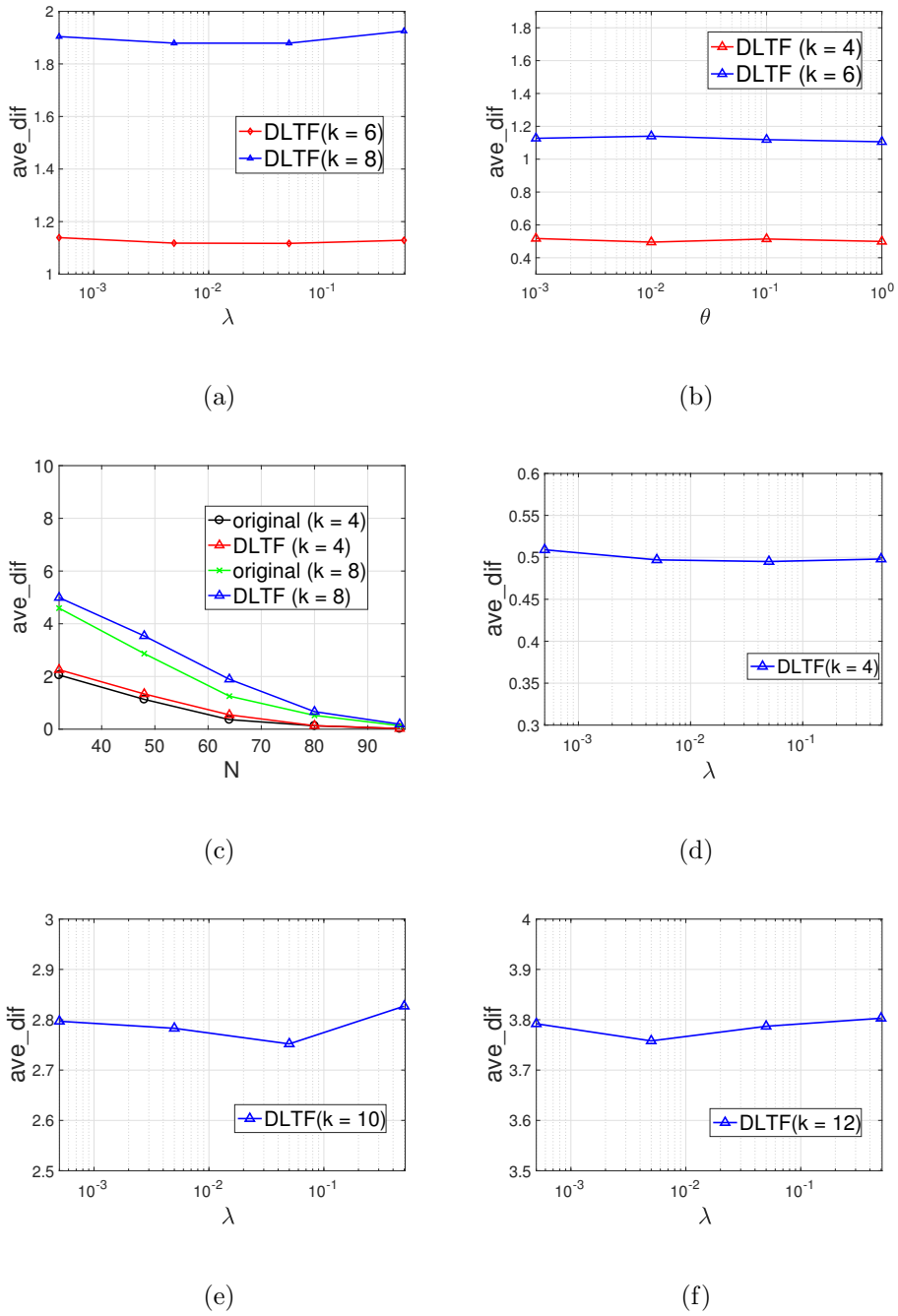


Figure 2.1: The support recovery performance comparison by varying: (a) the DLTF hyper-parameter λ; (b) the DLTF hyper-parameter θ; (c) the observed data dimension n. More comparison by varying λ for the following values: (d) $k = 4$; (e) $k = 10$; (f) $k = 12$.

rable results to the original baseline, especially when k is small. The performance gap slightly increases as k grows up. Compared to the original baseline with knowing W_0 as a prior, DLTF achieves competitive performance by only observing X. KSVD dictionaries perform poorly when applied to computing thresholded features, although the random baseline performs the worst. The above result clearly supports the necessity of DLTF.

2.5.2 Ablation Study

Next, we investigate the effects of varying several (hyper-)parameters in Figure 4.5(a)(b), all of which further verify the effectiveness and robustness of DLTF. It can be seen that DLTF maintains stable performance over a wide range of (λ, θ) values. When λ turns either too small or too large, the DLTF performance will be degraded a bit, which manifests the trade-off between the regularization effects of the first term (uncorrelated small noise) and the second term (small mutual coherence/RIP constant of W) in (2.4). Moreover, we further present the support recovery performance comparison plots by varying λ, at $k = 4$, 10 and 12, respectively in Figure 4.5 (d)(e)(f). The observations and conclusions are similar to $k = 6$ and $k = 8$ plots in Figure 4.5 (a)(b). We omit the same plots for θ, as we find the DLTF performance to be insensitive to varying θ at all k values. We choose $\lambda = 0.5$, $\theta = 0.01$ as default values hereinafter.

It is also noted that Figure 4.5(c) varies $n \in [32, 48, 64, 80, 108]$, when $m = 128$ is fixed. Both the original baseline and DLTF benefit from increasing n/m ratio,

and their performance difference seems to vanish as $n/m \to 1$.

Finally, we compute the mutual coherence of W_0 and W_{DLTF}, at $m = 128$, $n = 64$, which are approximately 0.49 and 0.56, respectively[2], respectively. It indicates that DLTF indeed finds a solution W with low mutual coherence. Besides, Corollary 1.1 suggests a sufficient recovery condition of $\mu_W \leq \frac{1}{2k}$, which cannot be met in this case for $\forall k > 1$. While the condition is not necessary, it implies that our simulation settings are challenging.

2.5.3 Complexity and Running Time Analysis

The time complexity of computing the thresholded feature by (2.1) for N samples is $\mathcal{O}(N(mn + m + k \log m))$, which is roughly equal to the complexity of running iterative sparse solvers e.g., orthogonal matching pursuit (OMP) as used in [1], for only one iteration. In practice, implementation differences, hyper-parameter choices and stop conditions[3] can dramatically affect the efficiency of iterative sparse solvers. However, it is self-evident that the efficiency advantage of (2.1) over iterative solvers is independent of implementations.

In all our experiments, we generally obverse the running time of thresholded features to be one or even two orders of magnitudes less than iterative comparison methods, e.g., KSVD (using OMP for testing). Our default testing environment is

[2]Results possess certain randomness. We did not compute the RIP constant as its calculation is NP-hard.

[3]Iterative algorithms may be stopped by measuring residual fitting errors, maximum sparsity, or iteration numbers, etc.

Training		DLTF	DLTF	KSVD	KSVD
Testing		Exact	TF	Exact	TF
$k = 5$	ACC	0.532	0.545	0.524	0.517
	NMI	0.525	0.532	0.526	0.521
$k = 10$	ACC	0.584	0.557	0.529	0.524
	NMI	0.552	0.525	0.520	0.517
$k = 30$	ACC	0.585	**0.594**	0.596	0.586
	NMI	0.545	**0.550**	0.561	0.533
$k = 40$	ACC	0.572	0.567	0.599	0.591
	NMI	0.534	0.531	0.585	0.536
Testing Time (s)		3.639	**0.394**	2.851	0.409

Table 2.2: The ACC and NMI comparison between DLTF and KSVD methods w.r.t. different k values, and the total testing time (in seconds) comparison, of different approaches.

Matlab 2016b on a Macbook Pro with 2.7GHz Intel Core i5 CPUs. For example, in the synthetic experiment of $k = 8$, computing thresholded feature for $N = 10{,}000$ samples only costs **0.25s**. In comparison, solving OMP for the same $N = 10{,}000$ samples takes **3.69s**.

2.5.4 Experiments on Image Clustering

Clustering is an unsupervised task for which sparse codes are known to be effective features [221]. We conduct our clustering experiments on a publicly available subset of MNIST[4], where the first 10, 000 training images of the original MNIST benchmark constitute the training set. A separate set of 10, 000 images is used as the testing set to evaluate the generalization performance. We reshape each 28×28 image into a vector, constructing $X \in \mathbb{R}^{n \times N}$ where $n = 784$, $N = 10, 000$.

Method	DLTF TF	KM	AE-1	AE-2	DEC
ACC	0.594	0.484	0.507	0.571	0.762
NMI	0.550	0.483	0.501	0.531	0.738
Time (s)	0.387	0.241	0.414	0.767	1.727

Table 2.3: The ACC, NMI and testing time comparison between DLTF TF ($k = 30$) and several other approaches.

We choose $m = 400$, and first design four methods for comparison, by altering ways of training (i.e., learning dictionary via DLTF or KSVD) and testing (i.e., computing sparse codes iteratively or thresholded features):

[4] `http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html`

- *DLTF Exact:* we apply DLTF to learn $W_{\text{DLTF}} \in \mathbb{R}^{n \times m}$ from the training set X, and then exactly solve sparse code features of testing images w.r.t. W_{DLTF} to global optima, using the iterative algorithm [17]. K-Means clustering is applied to cluster the sparse codes features.

- *DLTF TF:* we apply DLTF to learn W_{DLTF} from the training set X, and then compute the threshold features of the testing images via (2.1), w.r.t. W_{DLTF}. K-Means clustering is applied to cluster the thresholded features.

- *KSVD Exact:* replacing DLTF with KSVD [1] in the DLTF Exact approach.

- *KSVD TF:* replacing DLTF with KSVD in DLTF TF.

In order to validate the effectiveness and efficiency of thresholded features and DLTF, two "exact" baselines are further compared. Two standard clustering metrics, the accuracy (ACC) and normalized mutual information(NMI) are used [221]. Larger ACC and NMI indicate better clustering performance.

We vary $k \in [5, 10, 30, 40]$ and report results in Table 2.2. We first compare two "TF" methods. DLTF TF is observed to outperform KSVD TF at most k values, and achieves the best ACC/NMI among all TF results at $k = 30$. Second, taking two "Exact" baselines into account, it is encouraging to see that DLTF TF consistently achieves similar performance to both, sometimes even superior. Interestingly, we find DLTF TF to usually perform comparably to, or even better than DLTF Exact ($k = 10, 30$). In contrast, KSVD TF always obtains inferior performance than KSVD Exact. The observation demonstrates that W_{DLTF} is better suited for thresholded features.

DLTF TF achieves its best performance at $k = 30$. In comparison, the two KSVD methods seem to favor larger $k = 40$. To understand why DLTF TF may prefer smaller k, the reason may be that the support recovery is more reliable with higher sparsity, based on Theorems 1, 2. On the other hand, small k may cause information loss in sparse codes. A "medium" k of 30 seems to best balance the trade-off here.

Moreover, the efficiency of DLTF/thresholded features is further evidenced by the total running time (averaged over different k cases) on the 10, 000-sample testing set. The running times of the thresholded feature (both DLTF TF and KSVD TF) are one order of magnitude faster than their exact counterparts. DLTF hence possesses the most competitive performance-efficiency trade-off among the four.

We next compare DLTF TF (with the best $k = 30$) with other types of clustering models, from the simplest/fastest K-Means to sophisticated neural network (NN) clustering models that typically involve very high complexity:

- *KM:* we first obtain a PCA matrix $\in \mathbb{R}^{m \times n}$ from X, and apply it to reduce the dimension of each testing image to $\mathbb{R}^m$. K-Means clustering is then applied.

- *AE-1:* an auto-encoder (AE) with one hidden layer $\in \mathbb{R}^m$. ReLU is used. K-Means clustering is applied to the hidden activations of the testing images.

- *AE-2:* an AE with two hidden layers, both $\in R^m$. K-Means clustering is applied to cluster the activations of the second hidden layer.

- *Deep Embedded Clustering (DEC):* a latest deep network clustering model that

simultaneously learns feature representations and cluster assignments [194]. We used their original MNIST model structure. Note that DEC is much more heavily parametrized than DLTF and other baselines[5].

To ensure a fair comparison with non-NN models, all NN models are first trained, and then tested in CPU mode using the Matlab Neural Network Toolbox.

From Table 2.3, DLTF TF largely surpasses AE-1 in term of both ACC and NMI, especially considering the fact that the two have the identical amount of parameters and that ReLU is also known to introduce sparsity. It is further interesting to find that DLTF TF even achieves more favorable clustering quality than AE-2, with lower complexity and half running time in practice. Those results demonstrates that DLTF TF is both efficient and effective for clustering.

The state-of-the-art deep clustering model DEC achieves better ACC/NMI results than DLTF TF and else. However, it involves many more parameters and higher complexity: the testing time of DEC is thus five times that of DLTF TF. Moreover, DEC is the only approach here that is specifically optimized for clustering, while DLTF and others all lead to general-purpose unsupervised features. As a minimal-complexity and general-purpose feature extraction way, DLTF TF provides a *complementary effort* to dedicated and highly-complex deep clustering models. We also recognize that, to obtain stronger clustering performance, DLTF can be easily integrated with other regularizations, e.g. graph Laplacian [221], as well

[5]DEC was originally trained and evaluated on the full MNIST set of 70,000 samples. For fair comparison with others, we re-train DEC on the given 10,000-sample training set and test on the 10,000-sample testing set.

as can be further optimized in a (clustering) task-driven way, e.g., using bi-level optimization [133].

2.5.5 Experiments on Unsupervised Hashing

Methods	LSH	SH	SpH	SparseH	DLTF
$m = 32$	0.166	0.089	0.145	0.149	0.148
$m = 64$	0.159	0.121	0.189	0.234	0.223
$m = 128$	0.268	0.165	0.232	0.312	0.296
$m = 256$	0.332	0.201	0.255	0.386	0.355

Table 2.4: The mAP comparison on the CIFAR-10 dataset for unsupervised hashing, at different code lengths m.

Hashing is one of the popular solutions for approximate nearest neighbor search because of its low storage cost and fast retrieval speed. Many methods have been proposed to learn effective hash function. Hash codes of the same cluster are similar to each other while the hash codes in different clusters are dissimilar. In this section, we evaluate the proposed DLTF approach for the task of unsupervised hashing, inspired by the previous success of sparse hashing [224], as well as the nonzero support itself serving as a natural binary feature. It is also more challenging as unsupervised hashing relies on only unlabeled data to generate binary hashing codes. We choose local sensitive hashing (LSH) [25], spectral hashing (SH) [191], and spherical hashing (SpH) [88] as three classical baselines, following their default settings. We also implement a sparse coding-based hashing (SparseH) approach, that follows the

pipeline of [224]: a dictionary is learned on the training set, which is then used to solve sparse codes of the query images iteratively; sparse codes are binarized by encoding non-zero entries to ones and zeros elsewhere. Finally, DLTF is adapted for hashing by solving the dictionary W from (2.4), computing $\bar{\boldsymbol{z}}$ via (2.1), and using $\text{sgn}(\bar{\boldsymbol{z}})$ as binary codes. We choose $k = \frac{m}{2}$ [6]at each code length m. $\lambda = 0.1$ and $\theta = 0.01$ are fixed.

We test on the CIFAR-10 [106] benchmark, which has been widely used to evaluate both supervised and unsupervised hashing methods. We use GIST descriptors $\in \mathbb{R}^{512}$ to represent each image, and discard the label. A query set is formed by randomly choosing 1,000 samples, and a non-overlapping training set is constructed using the rest. We vary the hashing code length m from 32 bits to 256 bits, to evaluate the performance of all methods on compact codes and relatively long codes. The *mean average precision* (mAP) is evaluated at different numbers of bits. As seen from Table 2.4, while SparseH maintains the best performance among all in most cases (except being outperformed by LSH at $m = 32$), DLTF produces comparable mAPs and usually ranks only next to SparseH. Notice that the mAP difference between SparseH and DLTF is minimal at small $m = 32$ or 64.

Moreover, hashing applications emphasize high query efficiency and low latency. SparseH inevitably suffers from the heavy computational overhead of iterative sparse coding inference. DLTF achieves comparable results with one to two orders

[6] $k = m/2$ is chosen to meet the $\{0,1\}$ bit balance, a desirable property for code efficiency. While the sufficient recovery conditions may not be satisfied for the large k, DLTF achieves good empirical performance.

of magnitude less testing time (similar to the previous time comparison between DLTF and KSVD baselines).

2.6 Concluding Remarks

In this chapter, motivated by support recovery theoretical guarantees, we proposed a novel approach to learn a dictionary which is optimized for applying the thresholded feature. The competitive performance and superior efficiency of the proposed approach are extensively studied in both synthetic simulations and real-data experiments. In future work, we seek more elaborating formulations of DLTF. For example, [120] suggested that minimizing $||W^\top W - I||_\infty$ could suppress μ_W better, although minimizing the former term is also accompanied with higher complexity.

Chapter 3: Sparse Representations for Face Recognition

3.1 Introduction

Face identification and verification are two main tasks in face-based biometrics. Face identification aims to recognize a person from a set of gallery (images or videos) and match the closest one to the probe, while verification determines whether a given pair of images or videos is from the same subject or not. In this chapter, we address the unconstrained face verification/recognition problem where the face images have been acquired under significant variations in pose, expressions, lighting conditions and background.

Compared with single image-based face recognition, a video provides more samples from frames containing the person of interest. However, it brings more challenges as videos are often acquired in unconstrained environments, under significant variations in poses, expressions, lighting conditions and backgrounds. These variations result in large intra-personal variations within a video sequence. Therefore, it is important to represent and model the same subject against these variations in videos. Video-based face recognition has become a very popular topic of research in recent years [32, 33, 84, 93, 100, 180, 181, 193, 211]. Given a video sequence, the objective is to recognize the person in the video. It is often interchanged with image-set

based face recognition [24, 30, 42, 90, 102, 129, 130, 182, 183, 208], when the image sets are sampled from videos.

Numerous methods have been proposed to exploit useful information contained in videos. Early approaches [6, 100, 102, 113, 161] addressed this problem through learning probabilistic models. This was then followed by computing the similarity between two videos to perform recognition. Later, more sophisticated statistical model-based approaches [24, 90, 93, 130, 180–183] were proposed to learn discriminative and compact representations for each subject.

Dictionary-based methods have been shown to achieve impressive performance in various tasks, such as image-based face recognition, object and action recognition [1, 48, 80, 98, 131, 198, 202, 207, 212, 215, 220]. This is under the assumption that images could be well represented by an approximately learned dictionary and related sparse codes. However, there are only a few reported efforts that use dictionaries for video-based face recognition [32, 128, 211]. Recently, [32] proposed to partition videos into several clusters and learned a separate sub-dictionary for each cluster. One limitation of this method is that the number of clusters needs to be predefined. [128] jointly learned a global projection matrix and a set of sub-dictionaries to encode the new features with discriminative sparse coefficients. However, this method suffers from high computational complexity. In addition, information useful for dictionary learning may be lost after projecting all the samples onto the same subspace. [211] learned a sub-dictionary along with a low-rank representation for each subject. However, the sub-dictionaries were independently learned and are not discriminative enough for classification.

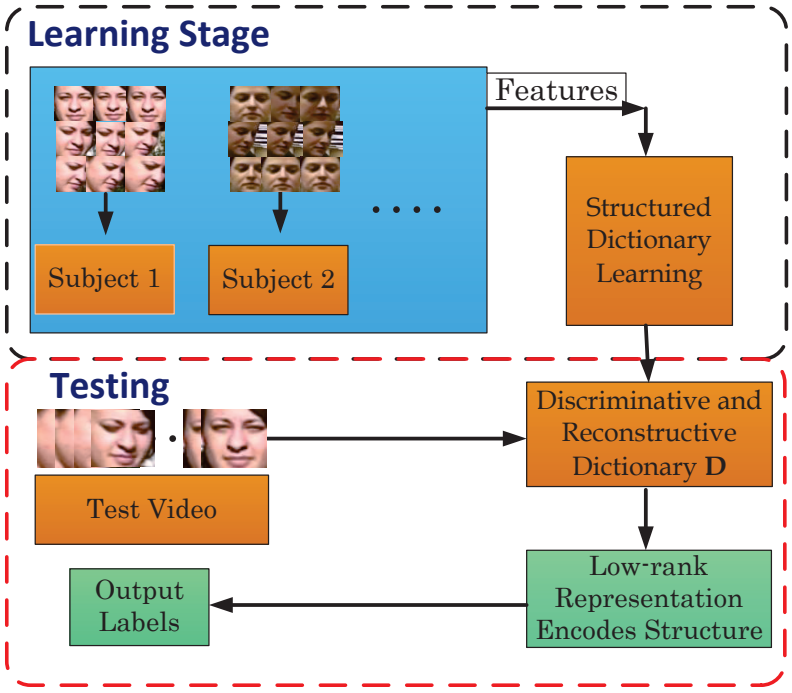


Figure 3.1: Overview of the proposed structured dictionary learning approach for video-based face recognition

In order to overcome the challenges discussed above, we propose a structured dictionary learning approach for video-based face recognition. The learned dictionary has the following three properties. First, it is reconstructive. We minimize the errors of all the face images when reconstructed from the dictionary, which encourages the learned dictionary to be reconstructive. Second, it is discriminative. For face images from each subject, we not only enforce the corresponding sub-dictionary to represent them well, but also enforce other sub-dictionaries not to be used for reconstruction. This will encourage different sub-dictionaries to encode features from different subjects. Third, it is capable of discovering structured information from different videos of the same subject. This is achieved by minimizing the rank of the representation matrix of face images from each subject. It is known that face images from one subject in different videos share some similar characteristics (*i.e.* consecutive pose change or similar facial appearance), which could be exploited to derive a low-dimensional subspace representation. Motivated by this underlying feature, we regularize the representation matrix of face images from the same subject in videos to produce a matrix of lower rank compared to the original data matrix. Figure 3.1 shows the overview of our approach.

Next, we study the other main task in the face-based biometrics: face verification, under even more challenging and unconstrained conditions. It is called **template**-based face verification problem. The problem of traditional face verification is to verify whether two images or videos in a pair belong to the same subject over image-to-image pairs as in Labeled Face in the Wild dataset [91], or over video-to-video pairs as in the Youtube Faces database [193], whereas template-

based face verification performs verification over **templates**, introduced in [103]. In this context, a template is a mixture of different media data such as images or frames sampled from multiple image sets or video clips containing the person of interest. Template representation is important in real world as it provides more flexibility and longitudinal access control of data from subjects.

Numerous methods have been proposed for improving the performance of face verification systems. To summarize, most existing approaches can be categorized into feature-based and metric learning-based methods. The first category, which includes LBP [3], SIFT [127], Fisher vector faces [167] and most recently the deep features [173], aims to derive robust and discriminative descriptors to represent face images. The common objective of the second category is to learn a good metric from the training data [19,41,47]. Some representative methods include cosine similarity metric learning [140], pairwise constrained component analysis [137] and logistic discriminant metric learning [79]. While dictionary learning techniques have shown impressive performance for face recognition [131,199,202,211,212], there are only a few reported works based on dictionaries for the face verification problem [52,80,81].

In this chapter, we tackle the problem of template-based face verification by taking advantage of dictionary learning techniques. This is due to the fact that image or video samples could be represented well by a learned dictionary and corresponding sparse codes. Yet dictionary learning methods have not been exploited for template-based face verification. Two issues arise when existing dictionary-based methods such as [81] are used for template-based face verification. First, the dictionary learned by random sampling of the training data is not able to adequately

represent the training set of face templates when several hundreds subjects are involved. Second, the sparse codes of all the samples from the same template are independently calculated, even though these samples are from the same subject. This may degrade the performance, especially when each template has significantly varying number of samples acquired from unconstrained environments. It is better to exploit this intra-class relationship among samples from the same template.

In order to overcome the limitations discussed above, we propose a novel template regularized sparse coding framework for template-based unconstrained face verification. The proposed approach consists of three steps. First, we construct a reference dictionary to adequately represent the training set. Then we exploit the intra-class relationship of the template by regularizing the sparse codes of the samples in one template to be similar, which results in more discriminative sparse codes. Finally, we measure the similarity between templates.

We make the following contributions on both video-based face recognition and template-based face verification problems:

- For the video-based face recognition problem, we present a dictionary learning approach with both discriminative and reconstructive properties. The learned dictionary reveals that the structural information from video face images could be used for recognition directly. Our method learns a low-rank representation for video face images of the same subject, using an efficient alternating optimization algorithm. Furthermore, we demonstrate that the proposed algorithm achieves state-of-the-art methods on three benchmark databases for

video-based face recognition.

- We propose a dictionary learning framework for template-based face verification problem. Our method learns a reference dictionary, which adequately represents the training set. Furthermore, we construct two template adaptive dictionaries to adapt the pair of templates. In addition, we propose a novel template regularized sparse coding method, which is able to capture the information in the samples in one template. An efficient algorithm is employed to learn discriminative sparse codes. Finally, we demonstrate that the proposed framework outperforms several state-of-the-art methods on benchmark datesets for template-based face verification.

The rest of the chapter is organized as follows: In Section 3.2, we review dictionary learning methods and several state-of-the-art methods for both video-based face recognition and template-based face verification. In Section 3.3, we present the structured dictionary learning approach followed by an efficient optimization algorithm. We also evaluate the proposed method for video-based face recognition on three benchmark databases in Section 3.3. In Section 3.4, we present the template regularized sparse coding approach. We then evaluate the proposed method for face verification on the benchmark template-based dataset [103] in Section 3.4. Section 3.5 concludes the chapter with a brief summary.

3.2 Related Work

Dictionary Learning: Dictionary learning [1,48,80,98,104,131,199,202,207, 211,212,215,220] has attracted great interest in subspace modeling for classification purpose. It overcomes the limitation of PCA subspaces by using non-orthogonal atoms (columns) in the dictionary to provide more flexibility to model the data. K-SVD [1] is one of the most commonly used techniques to learn a dictionary. Several algorithms have been developed to make the dictionary more discriminative [98,131,207,212,215]. [98] proposed a Label Consistent K-SVD to learn a compact dictionary by incorporating the training labels. [215] presented a structured low-rank representation based on a dictionary to boost the classification performance. [207] integrated the Fisher discrimination criterion with dictionary learning, which resulted in a more discriminative dictionary and sparse codes.

However, only a few works have been reported for the face verification problem [52,80,81]. One of the first methods [81] which adopted dictionary learning for face verification measured the similarity between the pair of images over the sparse codes using a reference dictionary. Subsequently, this work was extended by learning the local sparse codes from the patches of the face images. Although effective, learning patch-based sparse codes is sensitive to local perturbations. [80] generalized the dictionary learning framework to verification problems by adding a pairwise constraint. However, it suffers from high computational complexity. Furthermore, all these methods addressed the verification problem in image-to-image settings and are not directly applicable to template-based face verification [103].

Video-based Face Recognition: Existing video-based face recognition approaches [24,29,90,93,100,102,113,130,180–183] can be categorized into two classes: parametric and non-parametric. Early approaches [100,102,113] computed the similarity between the query video and training videos based on probabilistic models. Such methods were based on the assumption that a strong statistical correlation existed between the training and testing videos. To overcome the drawback of the probabilistic approaches, non-parametric approaches [24, 90, 93, 130, 180–183] represented the face images from videos as subspaces or manifolds. Linear/affine subspace-based methods [24,28,30,90,102,206] modeled the video face images as a linear or affine subspace. Among them, [24, 30, 90, 206] used convex geometry to represent videos from one subject, yielding improved performance over parametric approaches. However, to address the limitation of linear subspace models, more sophisticated nonlinear models have been extensively studied [29,83,93,180,182,183]. To preserve the nonlinear structure, [83,93,180,182] employed the concept of Grassmann manifolds, which is a special type of Riemannian manifold. [183] proposed more general discriminative analysis on Riemannian Manifold, which achieved encouraging results. A multi-kernel method combined with order statistics to perform classification was presented in [130]. Finally, deep learning approaches [84,129] have achieved state-of-the-art performance.

Template-based Face Verification: Several state-of-the-art methods for template-based face verification are briefly reviewed [27, 40, 159]. [27] addressed the template-based face verification problem through Joint Bayesian Metric Learning [23, 26] of deep CNN features. The triplet similarity embedding method [159]

learned an embedding matrix, which projects the original feature to a low-dimensional space. Template adaptation [40] learned two linear SVM classifiers, where each of them is designed using the positive features from one template in the pair to the large negative features from the training set. Then the final similarity score is calculated by fusing the two SVM margins evaluated on other mated template.

3.3 Learning Structured Dictionary for Video-based Face Recognition

In this section, we detail the proposed structured dictionary learning framework. The dictionary learned by our method is both discriminative and reconstructive for video-based face recognition.

3.3.1 Problem Formulation

Assume that we have videos from P different subjects, and each video contains a sequence of face images. Let the data matrix $X = [X_1, ..., X_P] \in \mathbb{R}^{d \times N}$ denote face images from P different subjects from the given videos, where N is the total number of images. Each $X_i = [x_{i1}, x_{i2}, ..., x_{iN_i}] \in \mathbb{R}^{d \times N_i}, 1 \leq i \leq P$ be the features of face images from i-th subject identity, and each column is the feature vector extracted from one frame.

We learn a dictionary $D \in \mathbb{R}^{d \times n}$ with both discriminative and reconstructive powers. The dictionary can be further decomposed into a set of sub-dictionaries as $D = [D_1, ..., D_P]$, where n is the number of atoms (columns) in the dictionary;

and $D_i \in \mathbb{R}^{d \times n_i}$ is the i-th sub-dictionary corresponding to the i-th subject. We reconstruct the features of face images from each subject X_i using the dictionary D, and obtain the corresponding encoding coefficients $Z_i \in \mathbb{R}^{n \times N_i}$. We can write the coefficient matrix Z_i over the dictionary D as $Z_i = [Z_i^1, Z_i^2, ..., Z_i^P]^T$, where Z_i^j denotes the coefficients of X_i over the sub-dictionary D_j.

We propose to learn a structured dictionary with following attributes: First, D should have small reconstruction errors for the training samples from all subjects. Second, each sub-dictionary D_i should represent face images only from the i-th subject, while different sub-dictionaries should be exclusive to each other. In order to achieve the above goal, the objective function for learning the dictionary D and representation coefficients Z is formulated as:

$$\begin{aligned} \min_{D,Z,E^1,E^2} & \sum_{i=1}^{P} (\|Z_i\|_* + \lambda_1 \|E_i^1\|_1 + \lambda_2 \|E_i^2\|_1) \\ s.t. \;\; & X_i = DZ_i + E_i^1, \\ & X_i = D_i Z_i^i + E_i^2, \quad \forall \; i, \;\; 1 \leq i \leq P \end{aligned} \tag{3.1}$$

where $E_i^1 \in \mathbb{R}^{d \times N_i}$ and $E_i^2 \in \mathbb{R}^{d \times N_i}$ are the reconstruction errors of X_i using the dictionary D and sub-dictionary D_i respectively. The parameters λ_1 and λ_2 balance two types of reconstruction error terms. The objective function in (3.1) leads to a dictionary D with both discriminative and reconstructive powers at the same time, and has three terms:

1. The first term denotes the nuclear norm of Z_i, which is the low-rank approximation of representation Z_i. Minimization of this term enforces the representation Z_i of samples from the i-th subject to lie on the same low-dimensional

subspace.

2. The second term E_i^1 is the l_1 norm of the reconstruction error of X_i with respect to dictionary D. We encourage D to be reconstructive, by minimizing the reconstruction errors for samples from all different subjects.

3. The third term is the l_1 norm of the reconstruction error of X_i with respect to the i-th sub-dictionary D_i. By minimizing this reconstruction error term, we encourage the i-th sub-dictionary D_i to represent the samples from its own class, while discouraging the usage of sub-dictionaries $D_j (j \neq i)$ from other classes for reconstruction. This regularization will make the dictionary discriminative.

3.3.2 Optimization

In this section, we present an efficient algorithm to solve the optimization problem in (3.1). The proposed algorithm uses the inexact Augmented Lagrange Multiplier (ALM) method to take advantage of its properties such as efficiency and convergence, for solving low-rank related problems [21, 119].

In order to make the objective function separable, we first introduce auxiliary variables W_i to replace Z_i $(1 \leq i \leq P)$. Denote $W = \{W_1, ..., W_P\}$, then the

function in (3.1) could be rewritten as:

$$\begin{aligned} \min_{D,Z,E^1,E^2,W} & \sum_{i=1}^{P}(\|W_i\|_* + \lambda_1\|E_i^1\|_1 + \lambda_2\|E_i^2\|_1) \\ s.t.\ \ & X_i = DZ_i + E_i^1, \\ & X_i = D_iZ_i^i + E_i^2 \\ & Z_i = W_i, \qquad\qquad \forall\ i,\ \ 1 \le i \le P \end{aligned} \tag{3.2}$$

The augmented Lagrangian function L of (3.2) is:

$$\begin{aligned} & L(D, Z, E^1, E^2, W, Y^1, Y^2, Y^3, \mu) \\ & = \sum_{i=1}^{P}(\|W_i\|_* + \lambda_1\|E_i^1\|_1 + \lambda_2\|E_i^2\|_1) \\ & + \sum_{i=1}^{P}(\langle Y_i^1, X_i - DZ_i - E_i^1\rangle + \langle Y_i^2, X_i - D_iZ_i^i - E_i^2\rangle + \langle Y_i^3, Z_i - W_i\rangle) \\ & + \frac{\mu}{2}\sum_{i=1}^{P}(\|X_i - DZ_i - E_i^1\|_F^2 + \|X_i - D_iZ_i^i - E_i^2\|_F^2 + \|Z_i - W_i\|_F^2) \end{aligned} \tag{3.3}$$

where $Y^1 = \{Y_1^1, ..., Y_P^1\}$, $Y^2 = \{Y_1^2, ..., Y_P^2\}$, $Y^3 = \{Y_1^3, ..., Y_P^3\}$ are all the multipliers, $\langle A, B\rangle = trace(A^TB)$ and μ is a positive scalar.

The optimization problem in (3.3) can be decomposed into two sub-problems and solved using the alternating method as in [215]. In the first sub-problem, the dictionary D is fixed and the optimal Z_i, E_i^1 and E_i^2 $(1 \le i \le P)$ are computed. In the second sub-problem, Z_i, E_i^1 and E_i^2 $(1 \le i \le P)$ are fixed, and the dictionary D is updated. We alternate these steps until convergence.

3.3.3 Computing Representation Z

Given dictionary D, the augmented Lagrangian function of (3.3) could be decomposed as the summation of P different sub-functions, where each sub-function

is only associated with one class label i ($1 \leq i \leq P$). Therefore, all the variables Z_i, E_i^1, E_i^2 and W_i ($1 \leq i \leq P$) in the sub-functions could be updated in a class by class fashion. When updating class i, variables Z_i, E_i^1, E_i^2 and W_i could be obtained as follows:

$$\begin{aligned} E_i^1 &= \arg\min_{E_i^1} \lambda_1 \|E_i^1\|_1 + \langle Y_i^1, X_i - DZ_i - E_i^1 \rangle + \frac{\mu}{2}\|X_i - DZ_i - E_i^1\|_F^2 \\ &= \arg\min_{E_i^1} \|E_i^1\|_1 + \frac{\mu}{2\lambda_1}\|(X_i - DZ_i + \frac{Y_i^1}{\mu}) - E_i^1\|_F^2 \end{aligned} \tag{3.4}$$

Similar to E_i^1, E_i^2 is updated as:

$$\begin{aligned} E_i^2 &= \arg\min_{E_i^2} \lambda_2 \|E_i^2\|_1 + \langle Y_i^2, X_i - DZ_i - E_i^2 \rangle + \frac{\mu}{2}\|X_i - DZ_i - E_i^2\|_F^2 \\ &= \arg\min_{E_i^2} \|E_i^2\|_1 + \frac{\mu}{2\lambda_2}\|(X_i - DZ_i + \frac{Y_i^2}{\mu}) - E_i^2\|_F^2 \end{aligned} \tag{3.5}$$

W_i is updated as:

$$\begin{aligned} W_i &= \arg\min_{W_i} \|W_i\|_* + \langle Y_i^3, Z_i - W_i \rangle + \frac{\mu}{2}\|Z_i - W_i\|_F^2 \\ &= \arg\min_{W_i} \|W_i\|_* + \frac{\mu}{2}\|(Z_i + \frac{Y_i^3}{\mu}) - W_i\|_F^2 \end{aligned} \tag{3.6}$$

Specifically, (3.4), (3.5) and (3.6) can be solved by singular value thresholding operation as in [119].

Note that when updating Z_i with other variables fixed, Z_i^i is also the corresponding component in Z_i with respect to the i-th sub-dictionary D_i. Here, we construct a matrix M such that $D_i Z_i^i = DM_i Z_i$, $M_i = \text{diag}(0, ..., 0, I_{n_0}, 0, ..., 0) \in \mathbb{R}^{n \times n}$; where $I_{n_0} \in \mathbb{R}^{n_0 \times n_0}$ located between index $n_0(i-1)+1$ and $n_0 i$. Then we could rewrite (3.3) as:

$$\begin{aligned} Z_i = \arg\min_{Z_i} &\langle Y_i^1, X_i - DZ_i - E_i^1 \rangle + \langle Y_i^2, X_i - DM_i Z_i - E_i^2 \rangle \\ &+ \langle Y_i^3, Z_i - W_i \rangle + \frac{\mu}{2}(\|X_i - DZ_i - E_i^1\|_F^2 \\ &+ \|X_i - DM_i Z_i - E_i^2\|_F^2 + \|Z_i - W_i\|_F^2) \end{aligned} \tag{3.7}$$

The optimization problem in (3.7) is a quadratic form in Z_i. Consequently, we can derive the optimal Z_i by setting the first-order derivative with respect to Z_i to be zero. The optimal solution is obtained as:

$$\begin{aligned} Z_i = &\left[D^T D + (DM_i)^T (DM_i) + I\right]^{-1} \left[D^T (X_i - E_i^1) + (DM_i)^T (X_i - E_i^2)\right. \\ &\left. + W_i + \frac{1}{\mu}(D^T Y_i^1 + (DM_i)^T Y_i^2 - Y_i^3)\right] \end{aligned} \tag{3.8}$$

The optimization procedure of the first sub-problem is summarized in Algorithm 2.

3.3.4 Updating Dictionary D

With a fixed Z_i, E_i^1 and E_i^2 $(1 \leq i \leq P)$, D is the only variable in (3.3). Denote $A_i = M_i Z_i$, then we could rewrite $D_i Z_i^i = DA_i$, for $A_i = [A_i^1, A_i^2, ..., A_i^P]^T \in \mathbb{R}^{n \times N_i}$; where its component A_i^i corresponding to D_i is equal to Z_i^i, and other components $A_i^j (j \neq i)$ are all zeros. Then the optimization function of D is

$$\begin{aligned} \min_D &\sum_{i=1}^{P} (\langle Y_i^1, X_i - DZ_i - E_i^1 \rangle + \langle Y_i^2, X_i - DA_i - E_i^2 \rangle) \\ &+ \frac{\mu}{2} \sum_{i=1}^{P} (\|X_i - DZ_i - E_i^1\|_F^2 + \|X_i - DA_i - E_i^2\|_F^2) \end{aligned} \tag{3.9}$$

The function in (3.9) is a quadratic form in variable D and the optimal solution is obtained as

$$D = \left[\frac{1}{\mu}(Y_i^1 Z_i^T + Y_i^2 A_i^T) + (X_i - E_i^1) Z_i^T + (X_i - E_i^2) A_i^T\right] \times \left[\sum_{i=1}^{P} (Z_i Z_i^T + A_i A_i^T)\right]^{-1} \tag{3.10}$$

The overall approach is summarized in Algorithm 3.

Algorithm 2: First Sub-problem Optimization via Inexact ALM

Input: Training data $X = [X_1, ..., X_P]$, Dictionary D, parameter λ_1, λ_2

Output: Z_i, E_i^1, E_i^2, Y_i^1, Y_i^2, Y_i^3 $(1 \leq i \leq P)$

Initialize: $\forall i = 1, ..., P$, $Z_i = W_i = Y_i^3 = 0$, $E_i^1 = E_i^2 = Y_i^1 = Y_i^2 = 0$, $\mu = 10^{-6}$, $\mu_{max} = 10^7$, $\rho = 1.25$

for class $i = 1, ..., P$ **do**

  Update Z_i,W_i,E_i^1 and E_i^2

  while not converged **do**

    Fix the others and update W_i according to (3.6)

    Fix the others and update E_i^1 according to (3.4)

    Fix the others and update E_i^2 according to (3.5)

    Fix the others and update Z_i by

    $$Z_i = \left[D^T D + (DM_i)^T (DM_i) + I\right]^{-1} [D^T (X_i - E_i^1) + (DM_i)^T (X_i - E_i^2) + W_i + \tfrac{1}{\mu}(D^T Y_i^1 + (DM_i)^T Y_i^2 - Y_i^3)]$$

    Update Multipliers

    $$Y_i^1 = Y_i^1 + \mu(X_i - DZ_i - E_i^1)$$

    $$Y_i^2 = Y_i^2 + \mu(X_i - D_i Z_i^i - E_i^2)$$

    $$Y_i^3 = Y_i^3 + \mu(Z_i - W_i)$$

    Update μ by

    $$\mu = \min(\rho\mu, \mu_{max}).$$

    Check the convergence condition:

    $$X_i - DZ_i - E_i^1 \to 0$$

    $$X_i - D_i Z_i^i - E_i^2 \to 0$$

    $$Z_i - W_i \to 0$$

  end while

end for

Algorithm 3: Overall Learning Framework

Input: Training data $X = [X_1, ..., X_P] \in \mathbb{R}^{d \times N}$, dictionary size n_0, parameter λ_1, λ_2

Initialize: Sub-dictionary D_i $(1 \leq i \leq P)$ by using k-SVD [1] algorithm, fix $\epsilon_d = 10^{-4}$

while not converged **do**

Update Z_i,W_i,E_i^1 E_i^2 $(1 \leq i \leq P)$ class by class using Algorithm 2.

Update Dictionary D according to (3.10)

Check the convergence conditions:

$\|D^{new} - D^{old}\|_F^2 < \epsilon_d$

end while

Output: Structured dictionary D and representation Z

3.3.5 Video-based Recognition

Once the discriminative and reconstructive dictionary D is learned, we predict the label of a given query video Y by computing the following terms:

$$Z = \arg\min_{Z} \|Z\|_* + \lambda_1 \|E\|_1 \quad s.t. \quad Y = DZ + E \tag{3.11}$$

where $Y = [y_1, ..., y_{N_y}] \in \mathbb{R}^{d \times N_y}$, N_y is the total number of face images. Note that during the training stage, D is learned such that each sub-dictionary D_i represents the i-th class, while different sub-dictionaries are exclusive to each other. Therefore,

we assign the label p^* with the smallest reconstruction error as:

$$p^* = \arg \min_{p \in 1,\ldots,P} \sum_{k=1}^{N_y} \|y_k - D_p z_k^p\|_2 \tag{3.12}$$

where y_k is the k-th face image vector in the query video and z_k^p is the sparse coefficient of y_k corresponding to the p-th sub-dictionary D_p.

3.3.6 Experiments

In this section, we present experimental results for video-based face recognition on three benchmark database, Honda/UCSD [113], CMU Mobo [76] and YouTube Celebrities [100] databases. We will first introduce these databases and their experimental settings. This is then followed by a discussion of the proposed approach.

Honda/UCSD [113]: There are in total 59 video sequences of 20 different subjects, where each subject has 2 or 3 video sequences. The video is acquired under large variations in expressions and head poses. Following the protocol in [113, 128, 129, 181], we select one sequence from each subject for training and test on the remaining sequences. We also evaluate our method with different lengths of training frames as in [32, 90, 208] by selecting 50 and 100 frames from each training video. The face detector presented in [178] was used to detect the faces. Faces were resized to 20×20 after histogram equalization to remove moderate illumination effect.

CMU Mobo [76]: It contains 96 video sequences of 24 subjects. Each subject has 4 video sequences captured in different walking situations. Face images are encoded using Local Binary Pattern (LBP) feature as in [90]. Following the standard protocol as in [24, 102], we randomly select one video from each subject to train while

Figure 3.2: Examples of YouTube Celebrities (YTC) database [100]

testing on the rest of all video sequences. This was repeated ten times.

Youtube Celebrities [100]: Youtube Celebrities Video is a widely used challenging database, which contains 1910 video clips of 47 subjects collected from YouTube. Some exemplar video frames are given in Figure 3.2. Each face is resized to 20×20 after using the face detector in [178] and pre-processed by histogram equalization as in [128, 130, 180, 181]. Intensity features are extracted for each face image. We conduct ten-fold cross validation experiments. For each subject, we randomly select 3 video clips for training and 6 for testing in each of the 10 folds. This setting ensures that both training and test data covered the whole video clips of each subject, which is the same with the protocol in [42, 93, 180, 181, 183] and similar to [129, 130].

We set all the sub-dictionaries to have the same number of atoms (columns), *i.e.* $n_i = n_0$. For Honda/UCSD and CMU Mobo databases, we run ten different trails under the standard settings and report the average recognition rate. The parameters λ_1,λ_2 have been empirically set to be 0.1 and 1 respectively. For a fair comparison with other dictionary learning approaches, the dictionary size n_0 is

set at 10 for the Honda/UCSD database and at 20 for the CMU Mobo database. For the YTC databese, we employ ten-fold cross validation and report the average recognition rate. Our rates are reported by settings $n_0 = 40$, $\lambda_1 = 0.02$ and $\lambda_2 = 0.1$.

3.3.7 Results and Analysis

Comparison with State-of-the-art Methods: In this section, we compare our results with several state-of-the-art listed next: Discriminant Canonical Correlation analysis (DCC) [102], Manifold-to-Manifold Distance (MMD) [182], Manifold Discriminative Analysis (MDA) [180], Covariance Discriminative Learning (CDL) [181], the linear version of Affine Hull-based Image Set Distance (AHISD) [24], Convex Hull-based Image Set Distance (CHISD) [24] and Sparse Approximated Nearest Points (SANP) [90], Joint Regularized Nearest Points (JRNP) [206], Dictionary-based Face Recognition from Video (DFRV) [32], Joint Dictionary and Subspace Learning (JDSSL) [211]. All the competing methods are implemented using the code provided by the authors except for JDSSL and JRNP. The parameters are tuned based on the settings reported in their papers. We implement the JDSSL following the algorithm in [211] and cite the results directly reported in JRNP [206] as a fair comparison for the Honda/UCSD database[1].

Honda/UCSD: The average recognition rates using 50, 100 and full length of training frames on Honda/UCSD are reported in Table 3.1. It is seen that most state-of-the-art methods achieve 100% rank-1 accuracy using full length of frames

[1]Results of JRNP [206] on CMU Mobo [76] and YTC [100] databases have not been reported because the experimental settings we used are different from the ones in [206].

Methods	DCC [102]	MMD [182]	MDA [180]	AHISD [24]	CHISD [24]	SANP [90]
50 Frames	76.9	69.3	74.4	87.2	82.1	84.6
100 Frames	84.6	87.2	94.8	84.6	84.6	92.3
Full Length	94.9	97.1	97.4	89.7	92.3	94.8
Year	2006	2008	2009	2010	2010	2011
Methods	DFRV [32]	CDL [181]	RNP [208]	JDSSL [211]	JRNP [206]	Ours
50 Frames	89.7	87.2	87.2	87.2	92.3	**93.6**
100 Frames	97.4	94.3	94.9	97.4	**100.0**	**100.0**
Full Length	97.4	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Year	2012	2012	2013	2014	2015	

Table 3.1: Video-based face recognition results for the Honda/UCSD database [113] using different number of frames in each image set for training. Rank-1 recognition accuracy results are presented.

for training. When the number of training frames is reduced to 50 and 100, the performance of other methods degrade. However, when the frame length is 100, our method could still achieve 100% accuracy as reported in [206], which demonstrates that our dictionary is able to preserve the subspace structure even with a small number of training samples. In particular, our method consistently outperforms all other dictionary-based approaches [32, 211]. This is because the learned dictionary by the proposed method is not only reconstructive and discriminative, but also can encourage the discriminative coefficients to be of low rank. Overall, our method achieves the best performance under all three settings.

CMU Mobo: We repeat ten trials by different randomly selected training

Methods	DCC [102]	MMD [182]	MDA [180]	AHISD [24]	CHISD [24]
Accuracy	88.9	92.5	94.4	92.9	96.5
Methods	SANP [90]	DFRV [32]	CDL [181]	JDSSL [211]	Ours
Accuracy	96.1	95.2	94.1	96.3	**98.2**

Table 3.2: Video-based face recognition results for the CMU Mobo database [76]. Rank-1 recognition accuracy results are presented.

Methods	DCC [102]	MMD [182]	MDA [180]	AHISD [24]	CHISD [24]	SANP [90]
Accuracy	66.8	65.3	67.0	63.7	66.5	65.0
Methods	CDL [181]	JDSSL [211]	PML [93]	DARG [183]	Ours	
Accuracy	70.1	70.1	70.4	72.5	**72.8**	

Table 3.3: Video-based face recognition results for the YTC database [100]. Rank-1 recognition accuracy results are presented.

and testing image sets. The average recognition rates of the proposed method along with other methods are reported in Table 3.2. As shown in Table 3.2, our method achieves very high performance of 98.2% and outperforms all other methods.

YouTube Celebrities: We used the cropped face samples of size 20×20 for consistency with Honda/UCSD database and reported results using 10-fold validation. These are the proposed settings used in [42, 180, 181]. We also compare with other state-of-the-art methods in [93] and [183]. Table 3.3 summarizes the average recognition rates of different methods.

It is noted that the performance of all the methods on YTC degenerates significantly compared with Honda/UCSD and CMU Mobo. This is due to the large

diversity and variations in appearance of each subject. Moreover, the high compression ratio, which result in low quality and resolution of the images, makes the recognition problem more difficult. It can be seen that our method outperforms the dictionary-based approach [211] by 2.7%, which demonstrates the effectiveness of the dictionary. In addition, our method achieves state-of-the-art performance compared to [93, 183][2].

Comparison with Different Dictionary Learning Approach: We further compare the proposed method with two different dictionary leaning strategies to further illustrate the effectiveness of our method.

1. Subject-specific Dictionary Learning (Subject DL): Instead of learning a global structured dictionary, we simply learn each sub-dictionary $D_i, i = \{1, ..., P\}$ independently by setting $\lambda_1 = 0$. Then we concatenate all the sub-dictionaries D_i together to construct D.

2. Non-structured Dictionary Learning (Non-structured DL): We only consider two terms of reconstruction errors using D and D_i and remove the nuclear term $\|Z_i\|_*$ in (3.1) without encouraging the representations to be low-rank. Then we perform recognition directly using (3.12).

Table 3.5 shows the average recognition rates of three different dictionary learning strategies. Our method consistently outperforms Subject DL and Non-structured DL on all three databases. Compared to Subject DL, the dictionary

[2]Note that results from recent approaches [84, 128, 129, 206] have not been reported here since they employed different protocols from the settings in this dissertation.

Methods	Honda/UCSD [113]	CMU Mobo [76]	YouTube Celebrities [100]
Subject DL	98.4	95.8	69.5
Non-structured DL	95.9	94.3	67.7
Our method	**100.0**	**98.2**	**72.8**

Table 3.4: Average recognition rates of different dictionary learning approaches on Honda/UCSD, CMU Mobo and YouTube Celebrities databases. Rank-1 recognition accuracy are presented.

learned in our method is both discriminative and reconstructive. First, it is designed to have small reconstruction errors for all the samples. Second, each sub-dictionary could represent the corresponding subject well while different sub-dictionaries would be exclusive to each other. In contrast, Subject DL only learns sub-dictionary for representing the corresponding subject. Moreover, Non-structured DL only focuses on reconstruction error of the samples. However, our method encourages face images from the same subject to have similar representation by enforcing them to lie in a low-dimensional subspace, which leads to independence from different subjects.

Parameter Sensitivity: In order to evaluate the effects of dictionary size n_0 and hyper-parameters λ_1, λ_2 on our method, we run different choices of parameters on the CMU Mobo database and plot the results in Figure 4.5.

Firstly, in Figure 4.5(a), we compare our method with JDSSL [211] and two different learning strategies (Subject DL and Non-structured DL) under the same number of sub-dictionary atoms for a fair comparison. It is seen that our approach outperforms [211] and the other two dictionary learning algorithms, by a large mar-

gin for all the different number of atoms. This is because we learn more discriminative and reconstructive dictionaries to preserve the structure of the samples from videos, while [211] only learned each sub-dictionary to encode the samples from the corresponding subject. We can also observe that increasing the size of sub-dictionary from five to twenty five can result in improving the recognition performance. All the methods achieve the best performance when $n_0 = 25$. It is also interesting to note that when the size of sub-dictionary is forty, the performance degenerates slightly for all the methods. With a large sized dictionary, some redundant atoms in sub-dictionaries may be learned without being useful for recognition, thus affecting the partition-based decision to be made.

We also evaluate our approach with varying values of parameters λ_1 and λ_2 as shown in Figure 4.5(b) and Figure 4.5(c). It is observed that the performance is more sensitive to the choice of λ_1, which is associated with the reconstruction error when using dictionary D for reconstruction. This is because our method learns a discriminative and reconstructive global dictionary instead of concatenating the sub-dictionaries together, which are learned class by class.

3.4 Template Regularized Sparse Coding for Face Verification

3.4.1 Task and Overall Approach

The definition of template-based face verification can be simplified as follows: given a training set and a pair of templates from the test set, the objective is to verify whether the pair of templates is from the same subject or not.

Our approach for template-based face verification (1) learns a reference dictionary $\mathbf{D}^R$ (with the help of hierarchical clustering), and (2) learns more discriminative sparse codes for verification purposes through the proposed *template regularized sparse coding* method, and (3) defines two distance-measures between template pairs through *reference score* and *template adaptive score* for computing the final similarity score.

The proposed approach consists of three steps. First, we learn two types of dictionaries: a reference dictionary and template adaptive dictionaries. The reference dictionary is learned only from the training set, which is disjoint from test templates. The reference dictionary is used for learning the sparse representations of the test templates. Two template adaptive dictionaries are constructed by augmenting the reference dictionary with each template in the test pair respectively. Adding only one template to construct the template adaptive dictionary would result in adapting the reference dictionary to better represent the other templates from the same subject.

Second, we perform sparse coding both on the reference dictionary and template adaptive dictionaries to obtain two types of sparse representations. In particular, we regularize the sparse codes of the samples in one template of the test pair to be similar to each other.

Third, by using the two sparse codes obtained as discussed above, we compute two different similarity scores: reference score and template adaptive score. The reference score is defined as the similarity between the sparse codes of two templates with respect to the reference dictionary. Template adaptive score measures the

difference between two types of sparse codes of each template in the pair with respect to two types of dictionaries.

The motivation behind the template adaptive score is that, if two templates in a pair are from the same subject, then the sparse coding coefficients of samples from one template corresponding to the augmented part (the added dictionary atoms from the other template) will have a significantly high value, while other coefficients corresponding to the reference dictionary will be smaller. On the other hand, if the two templates are not from the same subject, the regularized sparse codes of two templates will not change significantly. Therefore, a higher template adaptive score indicates that the template pair, very likely comes from the same subject.

We first present the notations used in this section. Let $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_P] \in \mathbb{R}^{d \times P}$ be the general template data matrix, where P is the total number of samples in the template (P varies from template to template). Each $\mathbf{x}_i \in \mathbb{R}^d, 1 \leq i \leq P$ is the feature encoded from image or video frames in the template with unit l_2-norm. We denote the training set as $\mathcal{T}$, and a pair of templates $\mathbf{X}^A = [\mathbf{x}_1^A, ..., \mathbf{x}_{P_A}^A] \in \mathbb{R}^{d \times P_A}$ and $\mathbf{X}^B = [\mathbf{x}_1^B, ..., \mathbf{x}_{P_B}^B] \in \mathbb{R}^{d \times P_B}$ from the test set.

3.4.2 Reference Dictionary and Template Adaptive Dictionaries Learning

The first step in the method is to learn a reference dictionary. Let n be the number of subjects in the training set and n_i be the number of templates from subject $i (\in [1, n])$. We define the data matrix $\mathbf{T}^i = [\mathbf{X}_1^i, ..., \mathbf{X}_{n_i}^i]$ to represent the

subject i, where $\mathbf{X}_j^i (j \in [1, n_i])$ is the j-th template from person i. Consequently, we represent the entire training set by $\mathcal{T} = [\mathbf{T}^1, ..., \mathbf{T}^n]$.

A good reference dictionary should be able to represent the training set with a compact set of dictionary atoms. To make the reference dictionary adequately represent the training set, we perform hierarchical adaptive clustering on the training set. More specifically, for each subject data matrix $\mathbf{T}^i, i \in [1, n]$, we adaptively determine the value of k_i and select k_i most representative samples by alternating the following two steps: (a) Increasing k to $k+1$ (b) Applying the "k-medoids" algorithm [144] on $\mathbf{T}^i$ until the stopping criterion in (3.13) is satisfied. The alternating procedure is illustrated in Algorithm 4.

After we select k_i representative samples from $\mathbf{T}^i, i \in [1, n]$ subject by subject, the reference dictionary $\mathbf{D}^R$ is constructed by concatenating all the representative samples learned in Algorithm 4, *i.e.* $\mathbf{D}^R = [\mathbf{c}_1^1, ..., \mathbf{c}_{k_1}^1 | ... | \mathbf{c}_1^n, ..., \mathbf{c}_{k_n}^n]$. We can rewrite the reference dictionary as $\mathbf{D}^R = [\mathbf{d}_1^R, ..., \mathbf{d}_N^R] \in \mathbb{R}^{d \times N}$, where $N = k_1 + ... + k_n$ is the total number of atoms (columns) in the dictionary.

Furthermore, given a test pair of templates $\mathbf{X}^A = [\mathbf{x}_1^A, ..., \mathbf{x}_{P_A}^A] \in \mathbb{R}^{d \times P_A}$ and $\mathbf{X}^B = [\mathbf{x}_1^B, ..., \mathbf{x}_{P_B}^B] \in \mathbb{R}^{d \times P_B}$, we construct two template adaptive dictionaries $\mathbf{D}^A$, $\mathbf{D}^B$ by augmenting the reference dictionary with samples from each template as follows: $\mathbf{D}^A = [\mathbf{D}^R | \mathbf{X}^B] \in \mathbb{R}^{d \times (N + P_B)}$ and $\mathbf{D}^B = [\mathbf{D}^R | \mathbf{X}^A] \in \mathbb{R}^{d \times (N + P_A)}$.

Algorithm 4: Adaptive selection of k_i representative samples

Input: Training data $\mathbf{T}^i = [\mathbf{X}_1^i, ..., \mathbf{X}_{n_i}^i]$ from subject i, stopping threshold τ.

Initialize: $k = 1$

while not converged **do**

Increase k to $k + 1$

Find k mediods $\{\mathbf{c}_1^i, ..., \mathbf{c}_k^i\}$ and corresponding clusters $\{\mathcal{C}_1^i, ..., \mathcal{C}_k^i\}$ by using "k-mediods" clustering algorithm [144].

Compute r as follows:

$$r = \max_{1 \leq m \leq k} \max_{\mathbf{x}_j^i \in \mathbf{C}_m^i} \|\mathbf{x}_j^i - \mathbf{c}_m^i\|_2 \tag{3.13}$$

Check the convergence condition: $r \leq \tau$

end while

Output: k_i and representative samples $\{\mathbf{c}_1^i, ..., \mathbf{c}_{k_i}^i\}$

3.4.3 Template Regularized Sparse Coding

In this section, we present our template regularized sparse coding algorithm for the reference dictionary $\mathbf{D}^R$ and template adaptive dictionaries $\mathbf{D}^A$ and $\mathbf{D}^B$. We learn the sparse codes of the samples in one template by regularizing them to be similar as they are all from the same subject. For simplicity of notation, we drop the superscript in $\mathbf{D}^R$, $\mathbf{D}^A$ and $\mathbf{D}^B$ and denote the given dictionary as $\mathbf{D}$. Let the template data matrix be $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_P] \in \mathbb{R}^{d \times P}$. The template regularized sparse

codes are obtained as follows:

$$\mathbf{Z}^* = \arg\min_{\mathbf{Z}} \sum_{i=1}^{P} (\|\mathbf{x}_i - \mathbf{D}\mathbf{z}_i\|_2^2 + \lambda_1 \|\mathbf{z}_i\|_1 + \lambda_2 \|\mathbf{z}_i\|_2^2) + \frac{\beta}{2} \sum_{i,j=1}^{P} (\|\mathbf{z}_i - \mathbf{z}_j\|_2^2 w_{i,j}) \tag{3.14}$$

where $\mathbf{Z} = [\mathbf{z}_1, ..., \mathbf{z}_P]$ are the corresponding sparse codes of $\mathbf{X}$ and λ_1, λ_2, β are the regularization parameters. The term $\|\mathbf{z}_i\|_1$ is the sparsity regularization term and the term $\|\mathbf{z}_i\|_2^2$ ensures the stability of the solution as in [133]. The last term is called the *template regularization term*, which sums the weighted difference of sparse codes of any two samples in the template. Let $\mathbf{W}$ be the matrix with entry $w_{i,j}$ in the i-th row and j-th column.

Constructing Matrix W: Given the sparse codes $\mathbf{z}_i$ and $\mathbf{z}_j$ of any two samples $\mathbf{x}_i$ and $\mathbf{x}_j$, $w_{i,j}$ is defined as follows:

$$w_{i,j} = \begin{cases} e^{-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}, \text{ if } i \neq j \\ \\ 0, \quad \text{otherwise} \end{cases} \tag{3.15}$$

It is inversely proportional to the Euclidean distance between their original feature (*i.e.* $\|\mathbf{x}_i - \mathbf{x}_j\|_2$). It means that when two samples are very close or similar in the original feature space, the penalty associated with the difference of their sparse codes will be large. As the pair of templates could have different template size, in order to reduce the effect of the template size, we further normalize each column in $\mathbf{W}$ by its l_2-norm.

Optimization: We now discuss the optimization of (3.14). Equation (3.14) can be rewritten as

$$\mathbf{Z}^* = \arg\min_{\mathbf{Z}} \sum_{i=1}^{P} (\|\mathbf{x}_i - \mathbf{D}\mathbf{z}_i\|_2^2 + \lambda_1 \|\mathbf{z}_i\|_1 + \lambda_2 \|\mathbf{z}_i\|_2^2) + \beta \operatorname{Tr}(\mathbf{Z}^T \mathbf{Z}\mathbf{L}) \tag{3.16}$$

where $\mathbf{L}$ is the Laplacian matrix $\mathbf{L} = \mathbf{A} - \mathbf{W}$ and $\mathbf{A}$ is a diagonal matrix whose diagonal elements are the sum of row elements of $\mathbf{W}$, i.e. $a_{i,i} = \sum_{j=1}^{P} w_{i,j}$.

Motivated by [80, 112], we optimize $\mathbf{z}_i$ in a column by column fashion. Given dictionary $\mathbf{D}$, when updating $\mathbf{z}_i$ by fixing other $\mathbf{z}_j (j \neq i)$, the objective function of (3.14) with respect to $\mathbf{z}_i$ is reduced to:

$$\mathbf{z_i^*} = \arg\min_{\mathbf{z}_i} \ \|\mathbf{x}_i - \mathbf{D}\mathbf{z}_i\|_2^2 + \lambda_1 \|\mathbf{z}_i\|_1 + \lambda_2 \mathbf{z}_i^T \mathbf{z}_i + 2\beta \mathbf{z}_i^T (\mathbf{Z}\mathbf{L}_i) - \beta \mathbf{z}_i^T \mathbf{z}_i \mathbf{L}_{i,i} \tag{3.17}$$

The minimization of (3.17) is a $\mathbf{L}_1$-regularized least squares problem and we compute $\mathbf{z}_i$ by feature-sign search algorithm proposed in [112].

The analytical solution of $\mathbf{z}_i$ could be derived by setting the first derivative of (3.17) with respect to $\mathbf{z}_i$ to be zero:

$$\mathbf{z}_i^* = [\mathbf{D}^T\mathbf{D} + (\lambda_2 + \beta \mathbf{L}_{i,i})\mathbf{I}]^{-1} (\mathbf{D}^T \mathbf{x}_i - \beta \sum_{\substack{k=1 \\ k \neq i}}^{P} \mathbf{z}_k \mathbf{L}_{k,i} - \frac{1}{2}\lambda \boldsymbol{\theta}) \tag{3.18}$$

where $\boldsymbol{\theta}$ is the coefficient sign vector of $\mathbf{z}_i$. We choose a small value of β to ensure that the Hessian matrix $[\mathbf{D}^T\mathbf{D} + (\lambda_2 + \beta \mathbf{L}_{i,i})\mathbf{I}]$ is positive semidefinite, which guarantees the convexity of (3.16).

Thus, we learn four sets of regularized sparse codes of the templates ($\mathbf{X}^A$ or $\mathbf{X}^B$) with respect to the reference dictionary $\mathbf{D}^R$ and the template adaptive dictionary ($\mathbf{D}^A$ or $\mathbf{D}^B$) which are denoted as follows:

$$\begin{aligned}
\mathbf{Z}^A &= [\mathbf{z}_1^A, ..., \mathbf{z}_{P_A}^A] \in \mathbb{R}^{N \times P_A} \ \text{ coded with } \ \mathbf{D}^R \\
\tilde{\mathbf{Z}}^A &= [\tilde{\mathbf{z}}_1^A, ..., \tilde{\mathbf{z}}_{P_A}^A] \in \mathbb{R}^{(N+P_B) \times P_A} \ \text{ coded with } \ \mathbf{D}^A \\
\mathbf{Z}^B &= [\mathbf{z}_1^B, ..., \mathbf{z}_{P_B}^B] \in \mathbb{R}^{N \times P_B} \ \text{ coded with } \ \mathbf{D}^R \\
\tilde{\mathbf{Z}}^B &= [\tilde{\mathbf{z}}_1^B, ..., \tilde{\mathbf{z}}_{P_B}^B] \in \mathbb{R}^{(N+P_A) \times P_B} \ \text{ coded with } \ \mathbf{D}^B
\end{aligned} \tag{3.19}$$

3.4.4 Reference Score and Template Adaptive Score

After we learn the template regularized sparse representations using (3.14), we evaluate how similar the test templates are, by computing the reference score and the template adaptive score. The reference score is defined as the average of the cosine similarity between all the sample pairs from the two templates as follows:

$$\mathrm{REF}(\mathbf{X}^A, \mathbf{X}^B) = \frac{1}{P_A \times P_B} \sum_{i=1}^{P_A} \sum_{j=1}^{P_B} \cos(\mathbf{z}_i^a, \mathbf{z}_j^b) \tag{3.20}$$

where $\cos(\mathbf{z}_i^A, \mathbf{z}_j^B)(i \in [1, P_A], j \in [1, P_B])$ is computed as the cosine similarity between two sparse codes as in [140]

$$\cos(\mathbf{z}_i^A, \mathbf{z}_j^B) = \frac{(\mathbf{z}_i^A)^T \mathbf{z}_j^B}{\|\mathbf{z}_i^A\|_2 \|\mathbf{z}_j^B\|_2} \tag{3.21}$$

In addition, in order to exploit the full power of the template regularized sparse codes $\tilde{\mathbf{Z}}^A$ and $\tilde{\mathbf{Z}}^B$, we also compute the template adaptive score of the template pair [81]. Following the notation in (3.19), let us first define the sample adaptive score of one sample $\mathbf{x}_i^A$ in the template $\mathbf{X}^A$ as

$$\mathrm{adapt}(\mathbf{x}_i^A) = 1 - \cos(\mathbf{z}_i^A, \tilde{\mathbf{z}}_i^A(1{:}\mathrm{N})) \tag{3.22}$$

where cos metric is defined in (3.21). Similar to sample $\mathbf{z}_i^B$ in the template $\mathbf{X}^B$, we have $\mathrm{adapt}(\mathbf{x}_i^B) = 1 - \cos(\mathbf{z}_i^B, \tilde{\mathbf{z}}_i^B(1{:}\mathrm{N}))$. Note that the higher sample adaptive score indicates more significant change from the sparse code.

Therefore, the template adaptive score of the template pair is computed as:

$$\mathrm{ADP}(\mathbf{X}^A, \mathbf{X}^B) = \frac{1}{P_A \times P_B} \sum_{i=1}^{P_a} \sum_{j=1}^{P_b} \frac{1}{2} [\mathrm{adapt}(\mathbf{x}_i^A) + \mathrm{adapt}(\mathbf{x}_j^B)] \tag{3.23}$$

Finally, the similarity score of the tested template pair is computed as the average of the reference score and the template adaptive score.

3.4.5 Experiments

In this section, we present the results of the proposed dictionary approach on the challenging IARPA Janus Benchmark A(IJB-A) [103] dataset. We will first introduce the dataset and experimental settings. This is then followed by a discussion of the experimental results.

The IARPA Janus Benchmark A(IJB-A) [103] dataset contains $5,397$ images and $2,042$ videos, which sampled to $20,412$ frames from total 500 subjects. Each subject has 11.4 images and 4.2 video clips on average. The smallest representation unit of each subject constitutes the **template**, which comprises a mixture of still images and sampled video frames.

The evaluation of verification protocol from IJB-A is over 10 splits. Each split consists of training and testing sets without any overlapping subjects between them. The test set in one split contains around $11,748$ pairs of templates ($1,756$ genuine and $9,992$ poster pairs). True Accept Rates(TAP) at different False Accept Rates(FAR) are reported in the evaluation metric.

In our experiment, the faces are represented with deep features extracted using the network discussed in [27]. More specifically, the deep CNN network is trained on the CASIA-WebFace dataset [209] with non-overlapped $490,356$ face images of $10,548$ subjects to IJB-A dataset. We use the network presented in [27] to extract

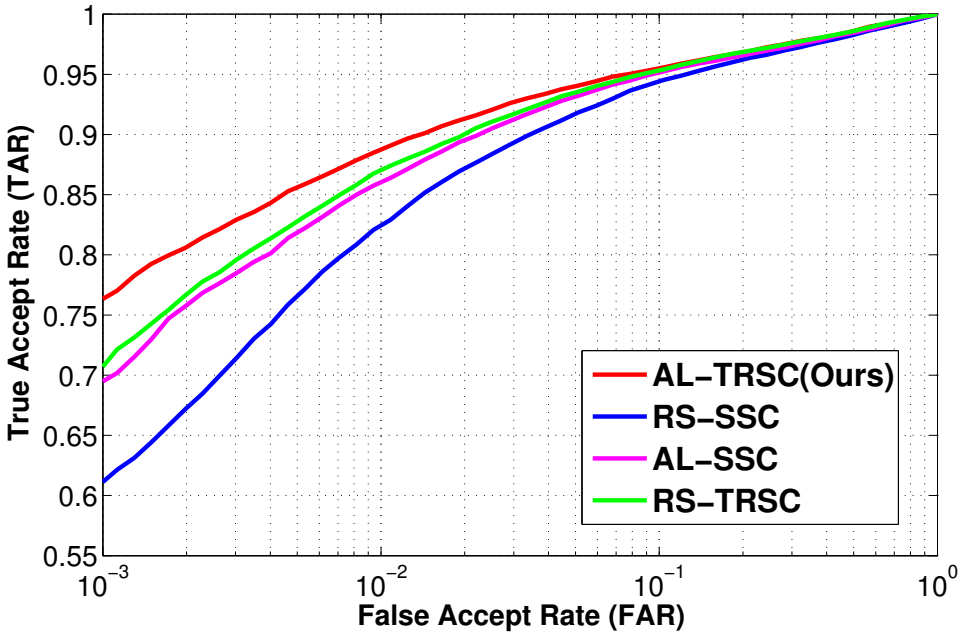


Figure 3.3: The average ROC curves of different dictionary learning and sparse coding strategies for the IJB-A [103] verification protocol over 10 splits

the 320-dimensional feature vector for each template in training and testing sets. Furthermore, following the setting in [40], in order to reduce the effect caused by the unbalanced size of different media (images or videos) in one template, we compute the mean feature to represent one video by averaging the features extracted from the same video clips. Finally, all the features in one template are normalized to have unit l_2-norm, which we call it the template media average features. The template media average features are used in all the experiments.

3.4.6 Results and Analysis

We perform two series of experiments to evaluate our approach for template-based face verification on IARPA Janus Benchmark A(IJB-A) [103] dataset.

Methods	FAR = 0.001	FAR = 0.01	FAR = 0.1
RS-SSC	0.613±0.059	0.824±0.026	0.944±0.007
AL-SSC	0.696±0.057	0.860±0.016	0.950±0.005
RS-TRSC	0.713±0.041	0.869±0.014	0.952±0.006
AL-TRSC(Ours)	**0.769±0.038**	**0.885±0.011**	**0.955±0.003**

Table 3.5: Verification accuracy comparison of different dictionary learning and sparse coding strategies for the IJB-A dataset [103]. The true accept rates(TAR) at false accept rate (FAR) of 0.001, 0.01 and 0.1 are reported.

Comparison of Different Dictionary Learning and Sparse Coding Strategies. To demonstrate the improvement of our approach (**AL-TRSC**) over [81] in both dictionary learning and template regularized sparse coding, we compare it with three methods:

- Random Sample + Single Sparse Coding (RS-SSC) [81]. We randomly select samples from the training set to generate the reference dictionary and independently compute the sparse codes of all the samples without the regularization term in (3.14).

- Adaptive Leaning + Single Sparse Coding (AL-SSC). We learn the reference dictionary as described in Section 3.4.2, followed by the same sparse coding strategy above.

- Random Sample + Template Regularized Sparse Coding (RS-TRSC). We

construct the reference dictionary by random sampling of the training set. However, we learn the template regularized sparse codes as described in Section 3.4.3

We plot the average ROC curves in Figure 3.3 of the four methods for the IJB-A dataset over 10 splits. In addition, we report the average TAR at FAR= 0.001, 0.01 and 0.1 in Table 3.5. First, our method (**AL-TRSC**) consistently outperforms AL-SSC, RS-TRSC and RS-SSC by a large margin. Compared with RS-TRSC, the reference dictionary, which is learned adaptively, is able to better represent the training set than random sampling. The AL-SSC algorithm only learns the sparse codes of all samples without template regularization. However, our method regularizes the sparse codes from one template to be close, which yields more discriminative sparse codes across template pairs. It is also noted that both AL-SSC and RS-TRSC achieve improvements over RS-SSC [81]. This demonstrates that both adaptive reference dictionary learning and template regularized sparse coding are indispensable for template-based face verification.

Comparison with State-of-the-art Approaches In order to evaluate the effectiveness of our approach (**AL-TRSC**) for template-based face verification, we further compare it with several state-of-the-art listed next: Joint Bayesian Metric Learning [27], Triplet Similarity Embedding (TSE) [159], Template Adaptation (TA) [40]. All the methods are implemented following the algorithm except [27]. The parameters are tuned based on the settings reported in their papers. We evaluate all the methods on the template media average features as a fair comparison,

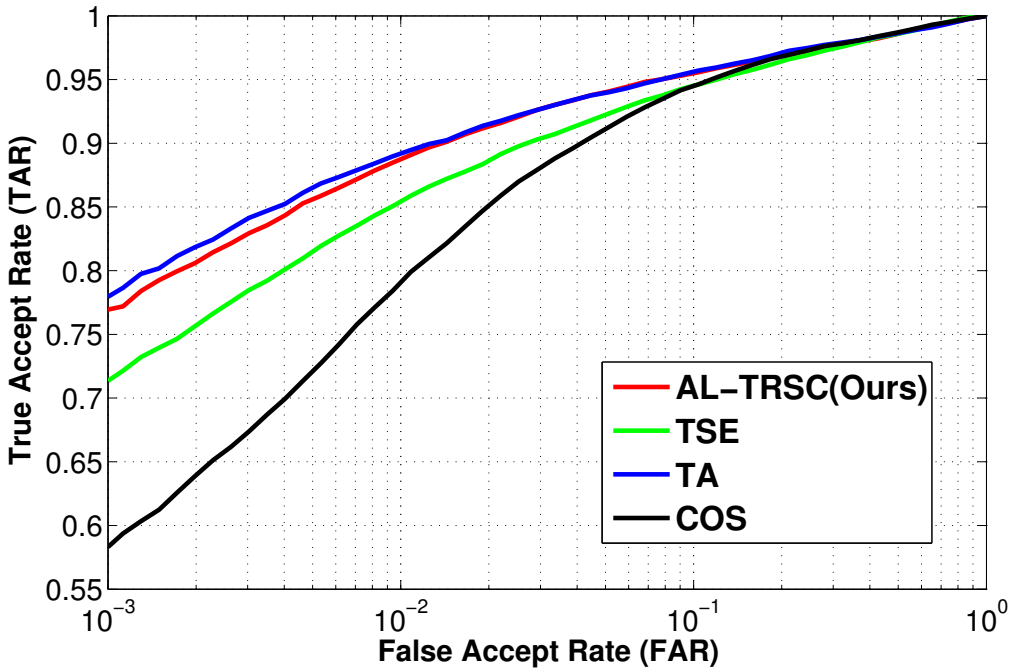


Figure 3.4: The average ROC curves of state-of-the-art and baseline methods for the IJB-A [103] verification protocol over 10 splits

Methods	FAR = 0.001	FAR = 0.01	FAR = 0.1
GOTS	0.198±0.008	0.406±0.014	0.627±0.012
COS	0.586±0.059	0.791±0.052	0.942±0.008
[27]	-	0.818±0.037	**0.961±0.010**
TSE [159]	0.718±0.039	0.855±0.019	0.945±0.005
TA [40]	**0.779±0.023**	**0.889± 0.012**	***0.955±0.007***
AL-TRSC(Ours)	***0.769±0.038***	***0.885±0.011***	***0.955±0.003***

Table 3.6: Verification accuracy comparison with state-of-the-art approaches for the IJB-A dataset [103]. The true accept rates (TAR) at false accept rate (FAR) of 0.001, 0.01 and 0.1 are reported.

which is the same as the setting in [40][3].

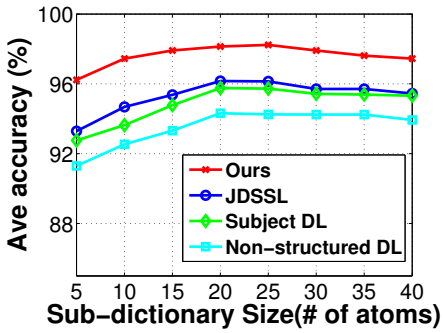
In addition, we also compare it with two baseline methods, the first one, COS computes the cosine similarity [140] from all the pair samples of two templates and average them to get the final similarity score between the two templates. The second baseline GOTS is from the commercial off-the-shelf matchers mentioned in NIST FRVT studies [77].

We plot the IJB-A average ROC curves over 10 splits of TSE [159], TA [40] and COS [140] in Figure 3.4. Furthermore, we also report the average TAR at FAR= 0.001, 0.01 and 0.1 in Table 3.6. All the methods [27, 40, 159] and ours improve the performance over COS and GOTS by a wide margin. Moreover, it can been seen that our method outperforms metric-based methods [27,159] and achieves results comparable to [40], which demonstrates the effectiveness of the proposed approach.

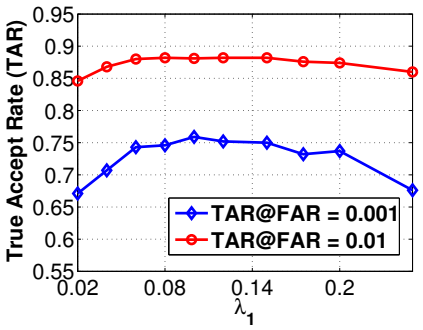
Parameter Sensitivity: In order to evaluate the effects of the stopping threshold τ in (3.13) and the hyper-parameters λ_1, λ_2, β in (3.14) of our method, we run different choice of parameters and plot the TAR with respect to the parameters at FAR = 0.001 and 0.01 in Figure 4.5.

Firstly, in Figure 4.5(a), it can been seen that both AL-SSC and AL-TRSC exhibit the same tendency with respect to τ. We observe that as τ decreases from 2.0 to 1.9, the verification performance improves. It is also interesting to note that when

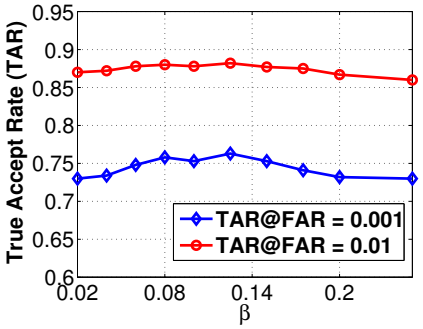
[3]Note that result DCNN_{ft+m+c} reported in [27] didn't use template media average features, all the other methods TA [40], TSE [159] and COS are evaluated on the same template media average features.



(a) stopping threshold τ



(b) parameter λ_1



(c) parameter β

Figure 3.5: The effects of stopping threshold τ, hyper-parameters λ_1 and β on IRAPA IJB-A dataset [103].

$\tau = 1.7$, the performance degenerates. With a large-sized reference dictionary, some atoms selected from the samples may not be useful for verification, thus affecting the regularized sparse coding. The final dictionary size is inverse proportional to the stopping threshold τ, and in order to balance the time and accuracy, we choose $\tau \in [1.85, 1.95]$ in all the experiments, which yields the reference dictionary size to be between 400 and 500.

We also evaluate our method by varying parameters λ_1 and β as shown in Figures 4.5(b) and 4.5(c). It is observed that the performance is more sensitive to

the choice of λ_1, which is associated with sparse penalty. Our results are generated by setting $\lambda_1 \in [0.08, 0.12]$ and $\beta = \{0.15, 0.1\}$. In addition, our approach is insensitive to the regularization parameter λ_2, which is set to 0.05 throughout all the experiments.

3.5 Concluding Remarks

In this chapter, we presented two dictionary learning-based methods for face recognition, especially for video-based face recognition and template-based face verification problems.

For video-based face recognition, we presented a novel structured dictionary learning framework. We encouraged our sub-dictionaries to represent the corresponding subject face images well, while also preserving the subspace structure by enforcing the representation to be low-rank. This approach yielded a dictionary with both discriminative and reconstructive properties for recognition purposes. Moreover, we proposed an efficient alternating optimization algorithm that converges reasonably faster. Finally, we extensively evaluated our approach on three benchmark databases for video-based face recognition. The experimental results clearly demonstrate the competitive performance over the state-of-the-art.

For template-based face verification, we presented a novel template regularized sparse coding approach. First, we adaptively learned a reference dictionary to adequately represent the training set. Then template adaptive dictionaries are generated by adapting the reference dictionary with the test template pair. Second,

we performed template regularized sparse coding on all the dictionaries to derive the discriminative template sparse codes for verification purpose. Finally, both reference and template adaptive scores are used to measure the similarity of the pair templates. We extensively evaluated our approach on the benchmark IARPA IJB-A dataset for template-based face verification. The experimental results clearly demonstrate competitive performance over state-of-the-art.

Chapter 4: Cross-domain Visual Recognition via Domain Adaptive Dictionary Learning

4.1 Introduction

Domain adaptation has been receiving significant attention in computer vision over the past decades. In real world scenarios, the assumption that the training data (***source*** domain) and test data (***target*** domain) are sampled from the same distribution is often challenged. For instance, training and testing images may be acquired under different environments, viewpoints and illumination conditions in application such as face recognition [4, 16], object recognition [70–72, 158], human detection [176] and video concept detection [53, 54, 205]. Recently, many works have been proposed to adapt the classifier trained using the source domain data to perform well on target samples [8, 9, 46, 62, 125, 141, 143, 147, 163, 205]. This is known as the ***domain adaptation*** (DA) problem. In this chapter, we focus on the more challenging unsupervised DA problem where the samples in the target domain are unlabeled. Moreover, it would be highly desirable for recognition systems to automatically adapt to a different domain without any additional labeling effort.

Recently, the most promising approaches for the unsupervised DA problem

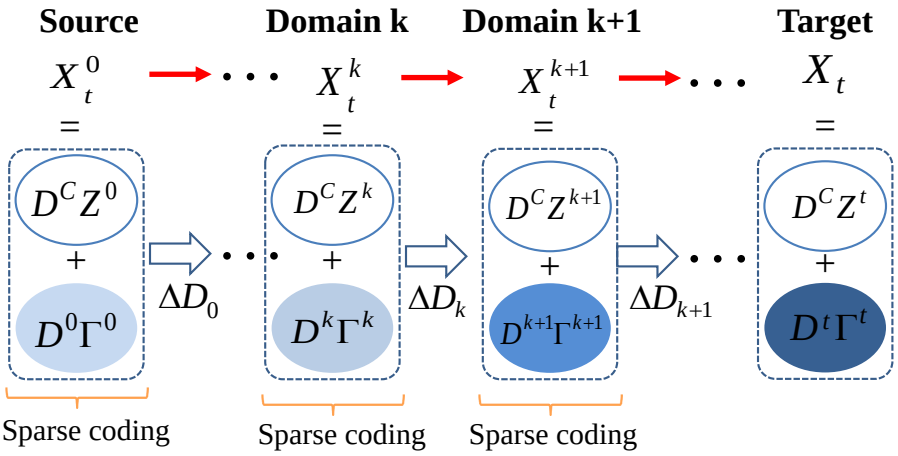


Figure 4.1: Our domain adaptive dictionary learning framework. The overall learning process consists of three steps: (1) Dictionary learning in source and target domains. At the beginning, we first learn the common dictionary $\mathbf{D}^C$, domain-specific dictionaries $\mathbf{D}^0$ and $\mathbf{D}^t$ for source and target domains. (2) Domain-adaptive sparse coding. At the k-th step, we enforce the recovered feature representations of target data in all available domains to have the same sparse codes, while adapting the newest obtained dictionary $\mathbf{D}^k$ to better represent the target domain. Then we multiply dictionaries in the k-th domain with the corresponding sparse codes to recover feature representations of target data $\mathbf{X}_t^k$ in this domain. (3) Dictionary updating. We update $\mathbf{D}^k$ to find the next domain-specific dictionary $\mathbf{D}^{k+1}$ by further minimizing the reconstruction error in representing the target data. Then we alternate between sparse coding and dictionary updating steps until the stopping criteria is satisfied.

focus on developing intermediate feature representations [71, 72, 141, 165] along a virtual path connecting the source and target domains. [72] generated intermediate subspaces by sampling the geodesic path connecting the source and target subspaces on the Grassmann manifold. Instead of sampling a few intermediate subspaces as in [72], [71] integrated an infinite number of intermediate subspaces to derive a geodesic flow kernel to model the domain shift. However, the subspaces obtained using principal component analysis (PCA) in both methods may not represent the original data well and some useful information for adaptation may be lost. In order to overcome the limitation of PCA subspaces, a recent work [141] used a dictionary to represent each domain, as non-orthogonal atoms (columns) in the dictionary provide more flexibility to model and adapt to the domain shift.

In this chapter, we propose a novel domain-adaptive dictionary learning approach to generate a set of intermediate domains which bridge the gap between source and target domains. Our approach defines two types of dictionaries: a common dictionary and a domain-specific dictionary. The common dictionary shared by all domains is used to extract domain-shared features, whereas the domain-specific dictionary which is incoherent to the common dictionary models the domain shift. The separation of the common dictionary from domain-specific dictionaries enables us to learn more compact and reconstructive dictionaries for deriving domain-adaptive features. All these dictionaries are learned using the procedure illustrated in Figure 4.1. First, we learn a common dictionary $\mathbf{D}^C$ by minimizing the reconstruction error of both source and target data. Then combined with the common dictionary, we learn a set of domain-specific dictionaries by alternating between the

following two steps: 1) domain-adaptive sparse coding: we learn domain-adaptive sparse codes $\mathbf{\Gamma}$ and $\mathbf{Z}$ by enforcing the feature representations of the target data to have the same sparse codes in all available domains. 2) dictionary updating: we update the current domain-specific dictionary to generate the next domain-specific dictionary such that the reconstruction error of target data is further minimized. This step not only guarantees that the next domain-specific dictionary will better represent the target data, but also ensures that the intermediate domains gradually adapt to the target domain. Finally, we apply domain-adaptive sparse codes combined with domain dictionaries to construct the final domain-adaptive features for recognition.

Ni et al.'s work in [141] may be the closest to our work in spirit. However, our approach differs in the following three aspects: (1) The separation of the common dictionary from domain-specific dictionaries. We aim to learn ***both*** the common dictionary and domain-specific dictionaries to represent each intermediate domain while [141] used only a ***single*** dictionary to represent each domain. Our approach has two advantages over [141]. First, our approach can better represent the domain data because the reconstruction error of domain data obtained using our method is smaller as shown in Figure 4.4 in Section 4.5. Second, the domain-specific dictionaries can better model the domain changes because the domain-shared features are accounted for separately. (2) The regularization of sparse coding. In each step, we regularize the representation of the target data along the path to have the same sparse codes, which are further used for dictionary updating in the next step. However, the sparse codes used in [141] for dictionary updating are only adaptive between

the ***neighboring*** domains. Therefore, the sparse representations of the target data in [141] are ***not*** domain-adaptive, while the sparse representations in our approach are ***domain-adaptive***. Moreover, the intermediate domains generated by our approach are smoother and incorporate the domain change in a better way, which will be verified and discussed in section 4.2. (3) The construction of final features. We use the domain-adaptive sparse codes across all the domains multiplied by the dictionaries to represent source and target data, while [141] only uses the sparse code decomposed with source and target dictionaries respectively to represent the new features. Therefore, compared to [141], our approach generates more robust and domain-adaptive features. We make the following contributions:

- We learn a common dictionary to extract the features shared by all the domains and a set of domain-specific dictionaries to encode the domain shift. The separation of the common dictionary from domain-specific dictionaries enables us to learn more compact and reconstructive representations for learning.

- We propose a new formulation to incrementally adapt the dictionaries learned from the source domain to reduce the reconstruction error of target data.

- We recover the feature of source and target data in all intermediate domains and extract novel domain-adaptive features by concatenating these intermediate features.

- We present empirical results for the tasks of object recognition and face recognition across pose, illumination, and blur variations, that are better than state-of-the-art algorithms.

The rest of the chapter is organized as follows: In Section 4.2, we review several dictionary learning methods for domain adaptation and other related domain adaptation approaches. In Section 4.3, we present our domain adaptive dictionary learning approach followed by an efficient optimization algorithm in Section 4.4. We evaluate the proposed method for domain adaptation problem on two benchmark databases in Section 4.5. Section 4.6 concludes the chapter with a brief summary.

4.2 Related Work

Recently, dictionary-based approaches [141, 147, 162] have been proposed for unsupervised DA. [147] learned a parametric modeled dictionary by aligning dictionaries from both domains. [162] jointly learned the projections of data in two domains, and a latent dictionary which can represent both domains in the projected low-dimensional space. [141] generated a set of intermediate domains and dictionaries which smoothly adapt the source domain to the target domain.

Another fruitful line of work is the subspace-based approaches [62, 71, 72, 163, 165]. [71, 72] created the intermediate domain subspaces along the geodesic on the Grassmann manifold connecting the source and target domains. [163] proposed to jointly learn domain-adaptive features and the classifiers on the target domain using an information-theoretic measure. [165] proposed an approach based on the parallel transport to incrementally learn the intermediate domains. [8, 70, 124, 125] attempted to match the distributions of the source and target samples by domain sample reweighting and feature matching.

Semi-supervised DA methods mainly focus on using samples with labels in the target domain to reduce the differences in data distribution [46, 70, 124, 143, 147, 158, 205]. Transformation-based methods [108, 158] learn linear or nonlinear transformations such that samples of the same class from different domains become closer. Classifier-based methods [5, 7, 99, 205] adapt the Support Vector Machine (SVM) trained in the source domain to correctly classify labeled target samples. A survey on visual domain adaptation could be found in [145].

4.3 Domain Adaptive Dictionary Learning

Let $\mathbf{X}_s \in \mathbb{R}^{d\times N_s}$, $\mathbf{X}_t \in \mathbb{R}^{d\times N_t}$ be the feature representations of source and target data respectively, where d is the feature dimension, N_s and N_t are the number of samples in the two domains. The feature representations of recovered source and target data in the k-th intermediate domain are denoted as $\mathbf{X}_s^k \in \mathbb{R}^{d\times N_s}$ and $\mathbf{X}_t^k \in \mathbb{R}^{d\times N_t}$ respectively. The common dictionary is denoted as $\mathbf{D}^C$, whereas source-specific and target-specific dictionaries are denoted as $\mathbf{D}^0$, $\mathbf{D}^t$ respectively. Similarly, we use $\mathbf{D}^k, k = 1...N$ to denote the domain-specific dictionary for the k-th domain, where N is the number of intermediate domains. We set all the dictionaries to be of the same size $\in \mathbb{R}^{d\times n}$.

Our objective is to learn the common dictionary and a set of domain-specific dictionaries for generating intermediate domains. Starting from $\mathbf{D}^0$ in the source domain, we sequentially learn the intermediate domain-specific dictionaries $\{\mathbf{D}^k\}_{k=1}^N$ to gradually reduce the reconstruction error of the target data. Our domain-adaptive

dictionary learning approach (**DADL**) consists of three steps: (1) Dictionary initialization. At the beginning, we first learn the common dictionary $\mathbf{D}^C$ and two domain-specific dictionaries $\mathbf{D}^0$, $\mathbf{D}^t$ for the source and target domains respectively. (2) Domain-adaptive sparse coding. At the k-th step, we learn domain-adaptive sparse codes of target data and recover the feature representations of target data in the k-th domain. (3) Dictionary updating. We update the current domain-specific dictionary $\mathbf{D}^k$ to find the next domain-specific dictionary $\mathbf{D}^{k+1}$ by further minimizing the residual error in representing the target data. We alternate between dictionary updating and sparse coding steps until the stopping criteria is satisfied.

4.3.1 Dictionary Learning in Source and Target Domains

At the beginning, we learn the common dictionary $\mathbf{D}^C$, source-specific dictionaries $\mathbf{D}^0$ and target-specific dictionary $\mathbf{D}^t$. Given source and target data $\mathbf{X}_s$ and $\mathbf{X}_t$, we solve for $\mathbf{D}^C$ by minimizing the reconstruction error of both source and target data as follows:

$$\begin{aligned} &\min_{\mathbf{D}^C,\mathbf{Z}^0,\mathbf{Z}^t} ||\mathbf{X}_s - \mathbf{D}^C\mathbf{Z}^0||_F^2 + ||\mathbf{X}_t - \mathbf{D}^C\mathbf{Z}^t||_F^2 \\ &s.t.\ \forall i,\ \ \|z_i^0\|_0 \leq T,\ \ \|z_i^t\|_0 \leq T \end{aligned} \tag{4.1}$$

where $\mathbf{Z}^0 = [z_1^0 ... z_{N_s}^0] \in \mathbb{R}^{n \times N_s}$, $\mathbf{Z}^t = [z_1^t ... z_{N_t}^t] \in \mathbb{R}^{n \times N_t}$ are sparse representations of $\mathbf{X}_s$ and $\mathbf{X}_t$ respectively, T specifies the sparsity that each sample has fewer than T dictionary atoms (columns) in its decomposition.

Given the learned $\mathbf{D}^C$ and corresponding sparse codes $\mathbf{Z}^0$ and $\mathbf{Z}^t$, we learn domain-specific dictionaries $\mathbf{D}^0$ and $\mathbf{D}^t$ by further reducing the reconstruction error

of source and target data. The objective function for learning $\mathbf{D}^0$ and $\mathbf{D}^t$ is given as follows:

$$\begin{aligned}
&\min_{\mathbf{D}^0,\mathbf{\Gamma}^0} \|\mathbf{X}_s - \mathbf{D}^C\mathbf{Z}^0 - \mathbf{D}^0\mathbf{\Gamma}^0\|_F^2 + \lambda\|\mathbf{D}^0\mathbf{D}^{C^T}\|_F^2 \\
&\min_{\mathbf{D}^t,\mathbf{\Gamma}^t} \|\mathbf{X}_t - \mathbf{D}^C\mathbf{Z}^t - \mathbf{D}^t\mathbf{\Gamma}^t\|_F^2 + \lambda\|\mathbf{D}^t\mathbf{D}^{C^T}\|_F^2 \\
&s.t. \forall i, \|z_i^0\|_0 + \|\alpha_i^0\|_0 \leq T, \quad \|z_i^t\|_0 + \|\alpha_i^t\|_0 \leq T
\end{aligned} \tag{4.2}$$

where $\mathbf{\Gamma}^0 = [\alpha_1^0 ... \alpha_{N_s}^0] \in \mathbb{R}^{n \times N_s}$ and $\mathbf{\Gamma}^t = [\alpha_1^t ... \alpha_{N_t}^t] \in \mathbb{R}^{n \times N_t}$ are sparse representations of $\mathbf{X}_s$ and $\mathbf{X}_t$ with respect to $\mathbf{D}^0$ and $\mathbf{D}^t$, and λ is the regularization parameter. The first term in both functions in (4.2) is the reconstruction error of domain data using both the common dictionary and corresponding domain-specific dictionary. The second term is the inner product of the atoms from different dictionaries, which encourages $\mathbf{D}^C$ to be incoherent to the domain-specific dictionaries. This incoherence term minimizes the correlation between $\mathbf{D}^C$ and $\{\mathbf{D}^0, \mathbf{D}^t\}$, thus it enables our approach to exploit domain-shared features and domain changes separately. We describe the optimization of the objective functions in (4.2) in Section 4.4.

4.3.2 Domain-adaptive Sparse Coding

At the k-th step, assume we have already generated (k-1) intermediate domains and domain-specific dictionaries denoted as $\{\mathbf{X}_t^i\}_{i=1}^{k-1}$ and $\{\mathbf{D}^i\}_{i=1}^{k-1}$ respectively. Now given a newly obtained domain-specific dictionary $\mathbf{D}^k$ for the k-th domain, we want to obtain sparse representations of target data $\mathbf{X}_t$ in the k-th domain. In order to achieve this goal, we not only reconstruct $\mathbf{X}_t$ using dictionaries from the k-th

domain, but also reconstruct the recovered target data $\mathbf{X}_t^i$ in each intermediate domain using dictionaries from that domain. Moreover, we regularize the sparse representation of $\mathbf{X}_s$, $\mathbf{X}_t$ and $\mathbf{X}_t^i$ to be the same. This regularization step ensures that the sparse representations of target data across all available domains are the same (i.e. ***domain-adaptive***). We solve for domain-adaptive sparse codes across all the available domains as follows:

$$\begin{aligned}\mathbf{Z}^k, \mathbf{\Gamma}^k =& \arg\min_{\mathbf{Z},\mathbf{\Gamma}} \|\mathbf{X}_t - \mathbf{D}^C\mathbf{Z} - \mathbf{D}^k\mathbf{\Gamma}\|_F^2 + \sum_{i=0}^{k-1} \|\mathbf{X}_t^i - \mathbf{D}^C\mathbf{Z} - \mathbf{D}^i\mathbf{\Gamma}\|_F^2 \\ &+ \|\mathbf{X}_t - \mathbf{D}^C\mathbf{Z} - \mathbf{D}^t\mathbf{\Gamma}\|_F^2 \quad s.t.\ \forall i, \|z_i\|_0 + \|\alpha_i\|_0 \leq T\end{aligned} \tag{4.3}$$

where $\mathbf{Z}^k = [z_1^k ... z_{N_t}^k], \mathbf{\Gamma}^k = [\alpha_1^k ... \alpha_{N_t}^k]$ are the solved sparse representations of target data in the k-th domain, $\mathbf{X}_t^i = \mathbf{D}^C\mathbf{Z}^i + \mathbf{D}^i\mathbf{\Gamma}^i$ are the recovered feature representations of target data in the i-th domain obtained in previous iteration steps. The objective function in (4.3) has two terms:

1. The first term is the reconstruction error of target data when encoded using dictionaries from the k-th domain. This term is called ***domain shifting*** term, because it adapts dictionaries in the k-th domain to better represent the target data.

2. The second term in (4.3) sums the reconstruction errors of recovered feature representations of target data in all the intermediate domains. The last term is the reconstruction error of target data in the target domain. These two terms are called ***domain adaptive*** terms. This is because we regularize both $\mathbf{X}_t$ and $\mathbf{X}_t^i$ to have the same sparse codes. It means that feature representations of recovered target data in different domains will have the same sparse

codes when encoded using dictionaries from each domain. This regularization will guarantee that sparse codes are domain-adaptive, such that the domain changes are encoded only in domain-specific dictionaries.

Then we recover the feature representations of target data in the k-th domain $\mathbf{X}_t^k$ as follows: $\mathbf{X}_t^k = \mathbf{D}^C\mathbf{Z}^k + \mathbf{D}^k\mathbf{\Gamma}^k$.

4.3.3 Domain-specific Dictionary Updating

After sparse coding at the k-th step, we will update $\mathbf{D}^k$ to find the next domain-specific dictionary $\mathbf{D}^{k+1}$ by further reducing the reconstruction error of target data in the k-th domain. Let $\mathbf{J}^k$ denote the target reconstruction residue in the k-th domain, which is computed as follows:

$$\mathbf{J}^k = \mathbf{X}_t - \mathbf{D}^C\mathbf{Z}^k - \mathbf{D}^k\mathbf{\Gamma}^k \tag{4.4}$$

where $\mathbf{Z}^k$ and $\mathbf{\Gamma}^k$ are the sparse codes obtained for reconstructing $\mathbf{X}_t$ in the k-th step. We further reduce the target reconstruction residue $\mathbf{J}^k$ by adjusting $\mathbf{D}^k$ by $\Delta\mathbf{D}^k \in \mathbb{R}^{d\times n}$, which is solved as:

$$\min_{\Delta\mathbf{D}^k} \|\mathbf{J}^k - \Delta\mathbf{D}^k\mathbf{\Gamma}^k\|_F^2 + \eta\|\Delta\mathbf{D}^k\|_F^2 \tag{4.5}$$

The objective function in (4.5) has two terms. The first term ensures that the adjustment $\Delta\mathbf{D}^k$ will further reduce the target reconstruction residue $\mathbf{J}^k$. While the second term penalizes the abrupt changes between two adjacent domain-specific dictionaries so that the intermediate domains smoothly adapt to the target domain. The parameter η controls the balance between these two terms. Since the problem in

(4.5) is a ridge regression problem, we solve for $\Delta\mathbf{D}^k$ by setting the first derivative to be zeros as in [141] and obtain:

$$\Delta\mathbf{D}^k = \mathbf{J}^k\mathbf{\Gamma}^{k^T}(\eta\mathbf{I} + \mathbf{\Gamma}^k\mathbf{\Gamma}^{k^T})^{-1} \tag{4.6}$$

where $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix. The next domain-specific dictionary $\mathbf{D}^{k+1}$ is obtained as: $\mathbf{D}^{k+1} = \mathbf{D}^k + \Delta\mathbf{D}^k$. In addition, we normalize each column in $\mathbf{D}^{k+1}$ to be a unit vector.

Proposition 1. *The residue* $\mathbf{J}^k$ *in* (4.4) *is non-increasing with respect to* $\mathbf{D}^C$, $\mathbf{D}^k$, $\mathbf{\Delta D}^k$ *and corresponding sparse codes* $\mathbf{Z}^k, \mathbf{\Gamma}^k$, *i.e.* $\|\mathbf{J}^k - \Delta\mathbf{D}^k\mathbf{\Gamma}^k\|_F^2 \leq \|\mathbf{J}^k\|_F^2$.

The non-increasing property of the residue $\mathbf{J}^k$ ensures that the source-specific dictionary $\mathbf{D}^0$ gradually adapts to the target-specific dictionary $\mathbf{D}^t$ through a set of intermediate domain-specific dictionaries $\mathbf{D}^k$. The proof is given in the Appendix.

After the domain-specific dictionary update, we increase k by 1, and alternate between the sparse coding step in section 4.3.2 and the dictionary updating step in section 4.3.3 until the stopping criteria is reached. We summarize our approach in Algorithm 5.

4.3.4 Derivation of New Features for Domain Data

Until now we have obtained the common dictionary $\mathbf{D}^C$, domain-specific dictionaries $\mathbf{D}^k, k \in [0, N]$. The transition path made up of $\mathbf{D}^c$ and the set of domain-specific dictionaries $\mathbf{D}^k$ models the domain shift. We will make use of it to derive new domain-adaptive representations for source and target data.

Algorithm 5: Our domain adaptive dictionary learning framework

Input: source data $\mathbf{X}_s$, target data $\mathbf{X}_t$, sparsity level T, parameter λ, η, stopping threshold δ

Output: $\mathbf{D}^C$, $\mathbf{D}^0$ and $\mathbf{D}^t$

compute $\mathbf{D}^C$ using (4.1)

compute $\mathbf{D}^0$, $\mathbf{D}^t$ by solving the objective function in (4.2).

$k = 0$

while stopping criteria is not reached **do**

    compute domain-adaptive sparse codes $\mathbf{Z}^k$, $\mathbf{\Gamma}^k$ using equation (4.3)

    compute the reconstruction error $\mathbf{J}^k$ using equation (4.4).

    compute the adjustment $\Delta\mathbf{D}^k$ using equation (4.6)

    $\mathbf{D}^{k+1} \leftarrow \mathbf{D}^k + \Delta\mathbf{D}^k$

    normalize $\mathbf{D}^{k+1}$ to have unit atoms.

    $\mathbf{X}_t^{k+1} \leftarrow \mathbf{D}^C\mathbf{Z}^k + \mathbf{D}^k\mathbf{\Gamma}^k$

    $k \leftarrow k + 1$

    Check the stopping criteria $\|\Delta\mathbf{D}^k\|_F \leq \delta$

end while

Final Output: $\mathbf{D}^C$, $\mathbf{D}^k, k \in [0, N]$ and $\mathbf{D}_t$.

Since the recovered feature representations of target data $\mathbf{X}_t^k, k \in [0, N]$ in all intermediate domains are already available, we first recover feature representations of source data $\mathbf{X}_s^k, k \in [0, N]$ in each intermediate domain. We iteratively recover $\mathbf{X}_s^k$ in a similar way as $\mathbf{X}_t^k$. The only difference is that all the dictionaries are already learned and fixed during the learning of $\mathbf{X}_s^k$. Specifically, at the k-th iterative step,

we obtain the sparse representations of source data that are adaptive across all domains by solving the following problem:

$$\mathbf{Z}_s^k, \mathbf{\Gamma}_s^k = \arg\min_{\mathbf{Z},\mathbf{\Gamma}} \|\mathbf{X}_s - \mathbf{D}^C\mathbf{Z} - \mathbf{D}^t\mathbf{\Gamma}\|_F^2 + \sum_{i=1}^{k-1} \|\mathbf{X}_s^i - \mathbf{D}^C\mathbf{Z} - \mathbf{D}^i\mathbf{\Gamma}\|_F^2$$
$$s.t. \quad \forall i, \quad \|z_i\|_0 + \|\alpha_i\|_0 \leq T \tag{4.7}$$

where $\mathbf{Z}_s^k = [z_{s_1}^k ... z_{s_{N_s}}^k], \mathbf{\Gamma}_s^k = [\alpha_{s_1}^k ... \alpha_{s_{N_s}}^k]$ are sparse representations of source data in the k-th domain, $\mathbf{X}_s^i = \mathbf{D}^C\mathbf{Z}_s^i + \mathbf{D}^i\mathbf{\Gamma}_s^i$ are recovered feature representations of source data in the i-th domain obtained in previous iteration steps. The objective function in (4.7) consists of two terms. The first term is the reconstruction error of source data using dictionaries from the target domain while the second term is the sum of reconstruction error of recovered feature representations of source data in all intermediate domains. Similarly, we enforce both $\mathbf{X}_s^0$ and $\mathbf{X}_s^i$ to have the same sparse codes. After sparse coding in the k-th step, we recover the feature representations of source data in the k-th domain as follows: $\mathbf{X}_s^k = \mathbf{D}^C\mathbf{Z}_s^k + \mathbf{D}^k\mathbf{\Gamma}_s^k$.

We use the sparse codes obtained in the last iterative step to derive the new feature representations for source and target data. The new augmented feature representation of source and target data are $\tilde{\mathbf{X}}_s = [\tilde{\mathbf{X}}_s^0, ..., \tilde{\mathbf{X}}_s^N]$ and $\tilde{\mathbf{X}}_t = [\tilde{\mathbf{X}}_t^0, ..., \tilde{\mathbf{X}}_t^N]$ respectively, where $\tilde{\mathbf{X}}_s^i = \mathbf{D}^C\mathbf{Z}_s^N + \mathbf{D}^i\mathbf{\Gamma}_s^N$ and $\tilde{\mathbf{X}}_t^i = \mathbf{D}^C\mathbf{Z}_t^N + \mathbf{D}^i\mathbf{\Gamma}_t^N$ and $\mathbf{Z}_s^N$, $\mathbf{Z}_t^N$, $\mathbf{\Gamma}_s^N$, $\mathbf{\Gamma}_t^N$ are the sparse codes obtained in the last iterative step where $k = N$. The final stage of recognition across all the domains is performed using an SVM classifier trained on new feature vectors after dimension reduction via the Principal Component Analysis (PCA).

4.4 Optimization

In this section, we provide the details of the optimization algorithms.

4.4.1 Source and Target Domain-specific Dictionaries Learning

The objective functions for learning source and target domain-specific dictionaries in (4.2) could be divided into two subproblems: (a) computing sparse codes with fixed $\mathbf{D}^C$, $\mathbf{D}^0$ and $\mathbf{D}^t$; (b) updating $\mathbf{D}^0$ and $\mathbf{D}^t$ with fixed sparse codes and $\mathbf{D}^C$. Since the two objective functions in (4.2) are in the same form, we only describe the optimization of $\mathbf{D}^0$ as shown below:

$$\begin{aligned} &\min_{\mathbf{D}^0,\mathbf{Z}^0,\mathbf{\Gamma}^0} \|\mathbf{X}_s - \mathbf{D}^C\mathbf{Z}^0 - \mathbf{D}^0\mathbf{\Gamma}^0\|_F^2 + \lambda\|\mathbf{D}^0\mathbf{D}^{C^T}\|_F^2 \\ &s.t. \quad \forall i, \|z_i^0\|_0 + \|\alpha_i^0\|_0 \leq T \end{aligned} \tag{4.8}$$

(a) Given fixed $\mathbf{D}^C$, $\mathbf{D}^0$, (4.8) is reduced to:

$$\begin{aligned} &\min_{\mathbf{Z}^0,\mathbf{\Gamma}^0} \|\mathbf{X}_s - \begin{bmatrix}\mathbf{D}^C & \mathbf{D}^0\end{bmatrix} \begin{bmatrix} \mathbf{Z}^0 \\ \\ \mathbf{\Gamma}^0 \end{bmatrix} \|_F^2 \\ &s.t. \quad \forall i, \|z_i^0\|_0 + \|\alpha_i^0\|_0 \leq T \end{aligned} \tag{4.9}$$

The minimization of (4.9) is a LASSO problem and we compute $\mathbf{Z}^0, \mathbf{\Gamma}^0$ using the method proposed in [132].

(b) Given fixed sparse coefficients $\mathbf{Z}^0, \mathbf{\Gamma}^0$, (4.8) is reduced to:

$$\min_{\mathbf{D}^0} \|\mathbf{J}_s - \mathbf{D}^0\mathbf{\Gamma}^0\|_F^2 + \lambda\|\mathbf{D}^0\mathbf{D}^{C^T}\|_F^2 \tag{4.10}$$

where $\mathbf{J}_s = \mathbf{X}_s - \mathbf{D}^C\mathbf{Z}^0$ is a fixed matrix. Motivated by [104] and [220], we update $\mathbf{D}^0 = [d_1^0, ..., d_n^0]$ atom by atom, i.e. updating the j-th atom d_j^0 by fixing other atoms

in $\mathbf{D}^0$. Specifically, let $\hat{\mathbf{J}}_\mathbf{s} = \mathbf{J}_s - \sum_{j \neq k} d_j^0 \alpha_{(j)}^0$ where $\alpha_{(j)}^0$ corresponds to the j-th row of $\mathbf{\Gamma}^0$, then we solve the following problem for updating d_j^0 in $\mathbf{D}^0$:

$$d_j^0 = \arg\min_{d_j^0} f(d_j^0) = \|\hat{\mathbf{J}}_s - d_j^0 \alpha_{(j)}^0\|_F^2 + \lambda \|{d_k^0}^T \mathbf{D}^C\|_F^2 \tag{4.11}$$

Let the first-order derivative of (4.11) with respect to d_j^0 equal to zero, *i.e.* $\frac{\partial f(d_j^0)}{\partial d_j^0} = 0$, then the closed form solution of d_j^0 is obtained as:

$$d_j^0 = (\|\alpha_{(j)}^0\|_2^2 \, \mathbf{I} + \lambda \mathbf{D}^C \mathbf{D}^{C^T})^{-1} \hat{\mathbf{J}}_\mathbf{s} {\alpha_{(j)}^0}^T \tag{4.12}$$

Also note that as an atom of a dictionary, it should be normalized to unit vector, i.e. $\hat{d_k^0} = d_k^0 / \|d_k^0\|_2$. Along with this, the corresponding coefficient should be multiplied $\|d_k^0\|_2$, *i.e.* $\hat{\alpha_{(j)}^0} = \|d_j^0\|_2 \, \alpha_{(j)}^0$.

We alternate between sparse coding and dictionary updating steps until the objective function in (4.8) converges, yielding the source domain-specific dictionary $\mathbf{D}^0$ and corresponding sparse coefficients $\mathbf{Z}^0, \mathbf{\Gamma}^0$.

4.4.2 Computing Domain-adaptive Sparse Codes

In (4.3), given fixed $\mathbf{D}^C$ and domain-specific dictionaries $\mathbf{D}^k$, $k \in [0, K]$, the objective function (4.3) could be rewritten as follows:

$$\mathbf{Z}^k, \mathbf{\Gamma}^k = \arg\min_{\mathbf{Z}, \mathbf{\Gamma}} \|\tilde{\mathbf{X}} - \tilde{\mathbf{D}} \, [\, \mathbf{Z} \;\; \mathbf{\Gamma} \,]^T \|_F^2 \tag{4.13}$$

where $\tilde{\mathbf{X}} = \left[\mathbf{X}_t^T, \mathbf{X}_t^T, \mathbf{X}_t^{0^T}, ..., \mathbf{X}_t^{k-1^T}\right]^T$ and $\tilde{\mathbf{D}} = \begin{bmatrix} \mathbf{D}^{t^T}, \mathbf{D}^{k^T}, \mathbf{D}^{0^T}, ..., \mathbf{D}^{k-1^T} \\ \mathbf{D}^{c^T}, \mathbf{D}^{c^T}, \mathbf{D}^{c^T}, ..., \mathbf{D}^{c^T} \end{bmatrix}^T$.
We can solve (4.13) as a LASSO problem to compute the sparse codes as in [132].

4.5 Experiments

4.5.1 Evaluation on CMU-PIE Face Dataset

The CMU-PIE dataset [166] is a controlled face dataset of 68 subjects with a total of $41,368$ images. Each subject has 13 images under 9 different poses, 21 different illuminations and 4 different expressions. We first evaluated our approach on the task of face recognition across blur and illuminations. Then we carried out experiment of face recognition across pose variation.

4.5.1.1 Face Recognition Across Blur and Illuminations

We followed the protocol in [141] to construct source and target domains. Specifically, we choose 34 subjects under first 11 illumination conditions to compose the source domain. The target domain was formed by the remaining images with the other 10 illumination conditions. The images in the source domain were labeled, but not those in the target domain. We synthesized domain shift by applying two different types of blur kernels to the target data: 1) Gaussian blur kernel with different standard deviations from 2 to 8, and 2) motion blur kernel with different lengths from 3 to 19 along $\Theta = 135^o$. In summary, the domain shift consist of two components. The first is a change in illumination direction, 11 illumination directions in the source domain and other 10 different illuminations directions in the target domain. The second component is due to blur.

We compared our method with the following approaches: (1) K-SVD [1],

which directly decomposes the target data with dictionaries from the source domain and uses a nearest neighbor classifier on the resulting sparse codes for testing. (2) GFK [71], which introduces the geodesic flow kernel on Grassmann manifold to calculate the distance between the source and the target samples. (3) SIDL [141], which creates a set of intermediate dictionaries to model the domain shift. (4) TJM [125], which jointly performs feature matching and instance re-weighting across domains. As in [141], we also compared with two other methods proposed in [4, 16]. [4] introduced a blur insensitive descriptor which was called the Local Phase Quantization (LPQ) while [16] estimated an illumination robust albed map (Albedo) for matching.

Tables 4.1 and 4.2 show the classification accuracies of different methods for face recognition across Gaussian blur and motion blur respectively. The proposed method achieves the best performance. This shows the benefits of our method for bridging the domain shift by iterative domain dictionary learning. It is also interesting to note that the baseline K-SVD which cannot handle the domain shift between the training and testing sets performs poorly, while other DA methods improve upon it. In addition, since both illumination and blur variations exist in the domain shift, LPQ [4] which is only blur robust and albedo [16] which is only illumination insensitive are not able to handle all the domain changes. Moreover, our method outperforms the method most similar to ours [141], which also learned a set of dictionaries to model the domain shift. This is because [141] only regularizes two adjacent domains to have the same sparse codes and the learned dictionaries do not fully capture the domain changes. However, our method encodes the domain

σ	2	3	4	5	6	7	8
Ours	**88.9**	**85.3**	**84.8**	**82.7**	**81.2**	**80.5**	**80.7**
SIDL [141]	84.0	80.3	78.9	78.2	77.9	76.5	74.8
TJM [125]	67.4	65.6	65.3	64.4	63.8	63.8	63.5
GFK [71]	81.1	78.5	77.6	75.9	74.0	72.1	70.4
LPQ [4]	69.1	66.5	64.4	61.6	58.3	55.3	53.2
Albedo [16]	72.4	50.9	36.8	24.8	19.6	17.3	15.7
K-SVD [1]	49.1	41.2	36.8	34.6	32.7	29.2	28.0

Table 4.1: Recognition accuracies across different Gaussian blur kernels on the CMU-PIE dataset [166]. Each column corresponds to Gaussian kernels with the standard deviation $\sigma = 2, 3, 4, 5, 6, 7, 8$.

changes into domain-specific dictionaries well by encouraging feature representation of different domain data to have domain-invariant sparse codes.

Benefit of learning the common dictionary and domain-specific dictionaries separately: Here we illustrate the benefits of the separation of the common dictionary and domain-specific dictionaries. Our method uses both the common dictionary and domain-specific dictionaries that are incoherent to the common dictionary to represent each intermediate domain while [141] only uses a single dictionary to represent it. We want to compare the difference between the synthesized feature representations of the target data obtained by the above two methods. Therefore, we compute and visualize the synthesized faces of one target face in in-

L	3	7	9	11	13	15	17
Ours	**97.9**	**89.7**	**88.2**	**82.5**	**77.4**	**75.0**	**70.8**
SIDL [141]	95.6	86.5	85.9	81.2	75.7	72.3	63.2
TJM [125]	71.8	69.4	66.2	64.1	60.0	57.1	54.1
GFK [71]	91.3	84.9	82.4	77.6	70.7	66.9	59.8
LPQ [4]	81.8	77.4	73.8	62.6	54.5	47.1	43.4
Albedo [16]	82.3	70.7	60.9	45.9	35.1	26.4	18.9
K-SVD [1]	85.0	56.5	42.6	30.3	25.9	19.8	17.3

Table 4.2: Recognition accuracies across different motion blur kernels on the CMU-PIE dataset [166]. Each column corresponds to motion blur with the length $L = 3, 7, 9, 11, 13, 15, 17$.

termediate domains using our method and the method proposed in [141] as shown in Figure 4.2. In addition, our synthesized faces of the target face in the first two rows have two components corresponding to the common dictionary and domain-specific dictionaries respectively, we also visualize the two components in intermediate domains in the first two rows of Figure 4.2. We observed that the synthesized faces obtained by two methods gradually transit from clear images to blur images. Moreover, the components that correspond to the common dictionary in the synthesized faces are always clear images while the components that correspond to domain-specific dictionaries become more and more blurred. This shows that the common dictionary has the ability to exploit features shared by all the domains, and only

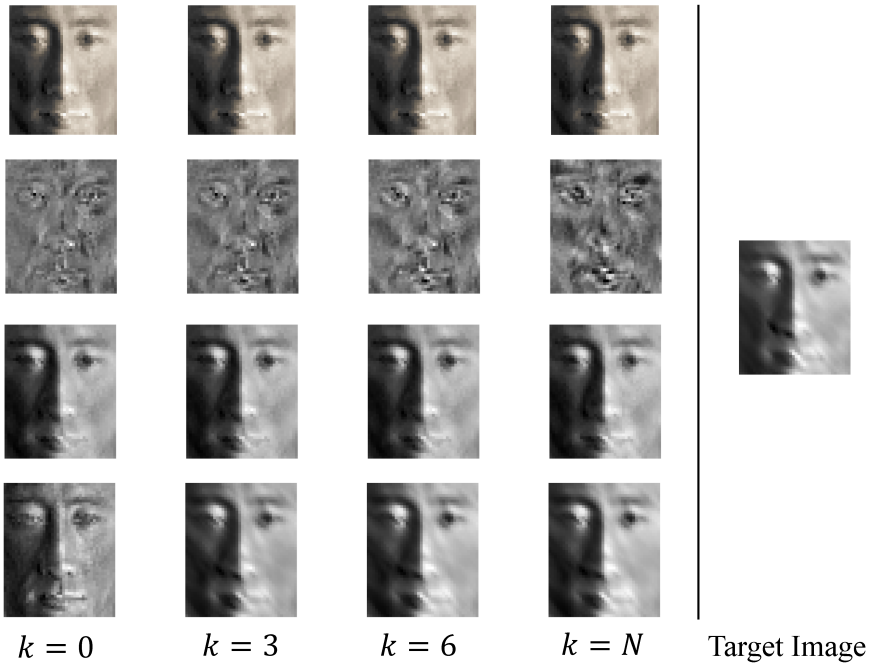


Figure 4.2: Synthesized face images of a target face along the intermediate domains. The image in the last column denotes the original target face, while the images in the first four columns are synthesized face images of the target face along four different intermediate domains. The images in row 1 and row 2 are components of our synthesized face images corresponding to the common dictionary and the domain-specific dictionaries respectively. The face images in row 3 and row 4 are synthesized face images generated by our approach and [141] respectively.

the domain-specific dictionaries are used to exploit the domain shift. In addition, it can also be seen that better reconstruction is achieved by our method, specifically for the region around the mouth where the motion blur is dominant. It further demonstrates that the separation of the common dictionary from domain-specific dictionaries enables us to learn more compact and discriminative representations for learning.

Methods	c11	c29	c05	c37	average
Ours	**86.7**	**98.5**	95.6	**89.7**	**92.6**
SIDL [141]	76.5	**98.5**	**98.5**	88.2	90.4
TJM [125]	83.8	**98.5**	95.6	82.4	90.1
GFK [71]	63.2	92.7	92.7	76.5	81.3
Eigen light-field [75]	78.0	91.0	93.0	89.0	87.8
K-SVD [1]	48.5	76.5	80.9	57.4	65.8

Table 4.3: Recognition accuracies across pose variation on the CMU-PIE dataset [166].

4.5.1.2 Face Recognition Across Pose Variation

The second experiment we carried out on the CMU-PIE data set is face recognition across pose variation. There are 5 different poses of face images ranging from frontal to $\pm 45^o$. The four non-frontal poses are denoted as *c05* (yaw about -22.5^o), *c29* (yaw about 22.5^o), *c11* (yaw about 45^o) and *c37* (yaw about -45^o). We selected the front-illuminated face images to be labeled source domain. Face images with the same illumination condition under four non-frontal poses formed faces in the target domain. The task is to classify the unlabeled face images in the target domain. We compared our method with the following methods: K-SVD [1], GFK [71], SIDL [141] and TJM [125]. In addition, we compare with Eigen light-field [75], which uses the appearance model to tackle pose variations in face recognition.

Figure 4.3: Recovered face images of a target face image along the intermediate domains. The first image is the original target face image, the second image is the component of the recovered face image corresponding to the common dictionary. The remaining six images are the components of the recovered face images corresponding to domain-specific dictionaries.

As shown in Table 4.3, our method outperforms its direct competitor [141] under all cases except the case where the target pose is $c05$. It is interesting to note that when the pose variations are large, [75] which relies on a generic training set to build pose model has higher average recognition accuracies than the unsupervised DA method proposed in [71]. However, our method demonstrates improved performances over both [75] and other domain adaptation approaches [71, 125, 141] when pose variations are large.

Benefit of learning the common dictionary and domain-specific dictionaries separately: In addition, we chose a target face with pose ID *c11* and synthesized feature representations of this face images in intermediate domains. Since synthesized face images have two components corresponding to the common dictionary and domain-specific dictionaries respectively, we visualize the two components of synthesized face images separately in Figure 4.3. It can be seen that the components corresponding to domain-specific dictionaries in intermediate domains gradually adapt from frontal face to non-frontal face. This demonstrates that

the domain-specific dictionaries have the ability to encode the domain shift due to different yaw angles.

4.5.2 Visual Object Recognition

In this section, we present the results generated by our approach for object recognition using the benchmark domain adaptation dataset introduced in [158]. This dataset contains visual objects across four different domains, *i.e.* Caltech, Amazon, DSLR, Webcam. Images from the first two domains Caltech [74] and Amazon are downloaded from Google and Amazon websites respectively. The last two domains include images captured by a digital SLR (DSLR) and a webcam (Webcam). Following [71], we selected 10 object classes common to all four domains with a total of 2533 images for our experiment. Image representation is based on SURF [13] features that are similar to those in [71, 158]. Specifically, all the images were resized to have the same width and converted to grayscale. The SURF detector [13] was then used to extract local scale-invariant interest points. Then a random subset of these interest point descriptors was quantized to 800 visual words by k-means clustering. Each image was represented by a 800-dimensional histogram.

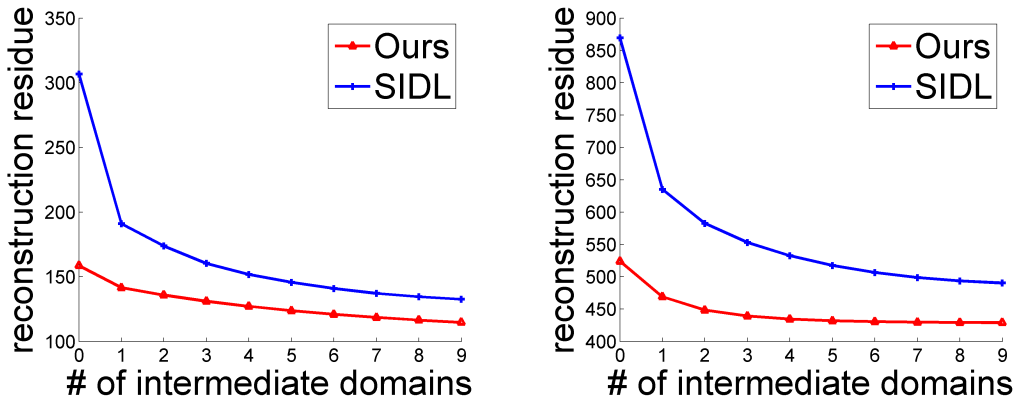
We evaluated our approach following the protocol introduced in [158]. In the source domain, we randomly selected 20 labeled images per category when Amazon, Webcam and Caltech are used as source domains, and 8 labeled images when DSLR is the source domain. We compared our method with K-SVD [1], GFK [71], SA [62], SIDL [141], TJM [125], DIP [8] and SIE [9]. We ran 20 different trials corresponding

Methods	C→A	C→D	C→W	A→C	A→W	A→D
K-SVD [1]	38.0	19.8	21.3	33.9	23.5	22.3
GFK [71]	40.4	41.1	40.7	37.9	35.7	36.3
SA [62]	39.0	39.6	23.9	35.3	38.6	38.8
SIDL [141]	43.3	42.3	36.3	40.4	37.9	33.3
TJM [125]	46.7	44.6	38.9	39.4	42.0	45.2
DIP [8]	50.0	49.0	47.6	43.3	46.7	42.8
SIE [9]	51.9	52.5	47.3	44.5	**48.6**	43.2
Ours	**54.7**	**53.7**	**48.1**	**45.3**	44.5	**45.8**
Methods	W→C	W→A	W→D	D→C	D→A	D→W
K-SVD [1]	17.1	16.7	46.5	22.6	14.3	46.8
GFK [71]	29.3	35.5	85.9	30.3	36.1	79.1
SA [62]	32.3	37.4	77.8	38.9	38.0	83.6
SIDL [141]	36.3	38.3	86.2	36.1	39.1	86.2
TJM [125]	30.2	30.0	89.2	31.4	32.8	85.4
DIP [8]	37.0	42.5	86.4	39.0	40.5	86.7
SIE [9]	39.9	**44.1**	89.3	38.9	39.1	88.6
Ours	**40.1**	41.8	**93.6**	**39.3**	**41.7**	**92.4**

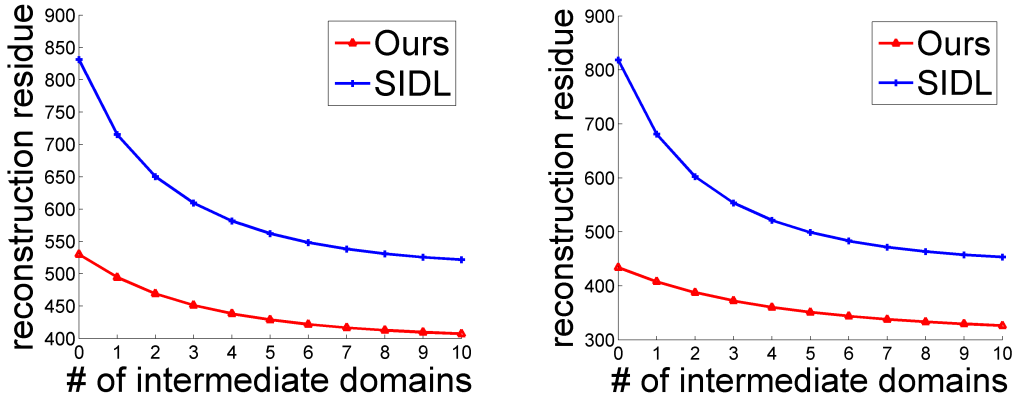
Table 4.4: Object classification accuracies of different approaches on the benchmark dataset [158]

to different selections of labeled source data and report the average recognition accuracy in Table 4.4. It can be seen that our method achieves the best performance for a majority of combinations of source and target domains. In particular, our method consistently outperforms SIDL [141] which is most similar to ours. This is because [141] only regularizes two adjacent domains to have the identical pairwise sparse codes and the learned dictionaries do not fully capture the domain changes. However, our method encodes the domain changes in the domain-specific dictionaries by encouraging feature representation of different domain data to have the same domain-adaptive sparse codes.

Decrease of reconstruction error along the transition path: We also show how the average reconstruction error of target data using both the common dictionary and domain-specific dictionaries changes along the transition path we have learned in Figure 4.4. First, we observe that reconstruction residues obtained by our method and [141] are gradually decreasing along the transition path, which provides empirical support to Proposition 1. Second, the proposed method achieves much lower reconstruction error. This demonstrates that our approach can learn more compact and more reconstructive dictionaries by learning the common and domain-specific dictionaries separately. Last but not the least, the proposed learning algorithm generally terminates within 8 to 10 steps, which demonstrates that the generated intermediate domains bridge the gap between source and target domains.

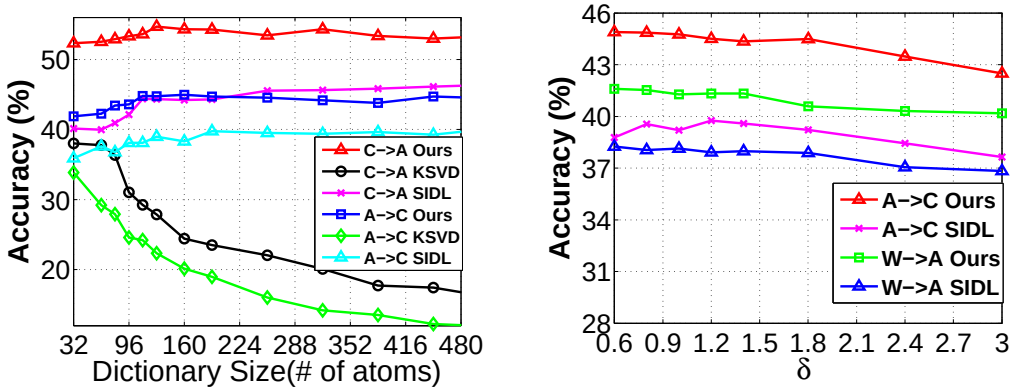


(a) clear v.s. motion blur frontal face ($L = 5$) (b) frontal face v.s. face images at pose $c05$



(c) Amazon v.s. Caltech (d) Webcam v.s. Amazon

Figure 4.4: Average reconstruction error of the target data decomposed using dictionaries along the intermediate domains. We compare with SIDL [141]. The source and target domains are: (a) clear frontal face v.s. motion blur frontal face ($L = 5$) (b) frontal face images v.s. face images at pose $c05$ (c) Amazon v.s. Caltech in Office dataset [158] (d) Webcam v.s. Amazon in Office dataset [158].



(a) dictionary size

(b) stopping threshold δ

Figure 4.5: The effects of dictionary size and stopping threshold δ on office datasets [158].

4.5.3 Parameter Sensitivity

In order to evaluate the effect of dictionary size on our approach, we choose two different combinations of source and target domains and plot the results in Figure 4.5(a). Our approach yields significant improvement over K-SVD [1] since we bridge the domain shift by generating intermediate domains. Our approach also outperforms SIDL [141] by a large margin of 4.5%. This is because we learn more compact and reconstructive dictionaries to represent target data, which leads to much lower reconstruction errors, as demonstrated in Figure 4.4. The dictionary size is set to be 128 or 256 based on the source sample size in all the experiments. We also evaluate our approach with varying values of stopping threshold δ as shown in Figure 4.5(b). It can be seen that both [141] and the proposed approach converge in fewer steps with increasing value of δ, thus generating fewer intermediate domains. In addition, our approach is insensitive to the regulization parameter η, which is

chosen from 1500 to 2500 throughout all the experiments. The final dimensionality after PCA is between 60 and 140.

4.6 Concluding Remarks

In this chapter, we presented a novel domain adaptive dictionary learning approach for unsupervised domain adaptation. We first learned a common dictionary to recover features shared by all domains. Then we acquired a set of domain-specific dictionaries, which generates a transition path from source to target domains. The common dictionary is essential for reconstruction while domain-specific dictionaries are able to bridge the domain shift. Final feature representations are recovered by utilizing both common and domain-specific dictionaries. We extensively evaluated our approach on two benchmark datasets and the experimental results clearly confirm the effectiveness of our approach.

Chapter 5: Deep Regionlets: Blended Learning of Regionlets and Deep Learning for Generic Object Detection

5.1 Introduction

In this chapter, we study another fundamental problem in computer vision: object detection. Object detection has been extensively studied in the computer vision over several decades [18,20,43,45,61,67,68,92,117,153,177,184–186,196,197,213,217] due to its appeal to both academic research explorations as well as commercial applications. While designing object detection algorithms, two key issues need to be carefully addressed: where the candidate locations are in images and how to discern whether they are the objects of interests. Because of these two issues, object detection has become one of the most valuable pattern recognition tasks, with great benefits in scene understanding [114], face recognition [149, 150, 175, 218], action recognition [96, 146], robotics and self-driving vehicles, etc. Although studied over several decades, accurate detection is highly challenging when generating bounding boxes for specific object categories, due to cluttered backgrounds, occlusions, variations in object scale, pose, viewpoint and even part deformations.

Prior works in object detection before the deep learning era addressed object

deformations with several strategies based on hand-crafted features (*i.e.* histogram of oriented gradients (HOG) [45], local binary pattern (LBP) [2], HOG-LBP [184], scale-invariant feature transform (SIFT) [126]). One of the earliest works, the classical Adaboost [178] detector deployed an ensemble classifier of fast features to model local variations, especially for the detection of faces or pedestrians. The deformable part model (DPM) [61] first proposed to model object deformations explicitly using latent variables, improved localization precision. However, these approaches usually involve exhaustive search for possible locations, scales and aspect ratios of the object, by using the sliding window approach. Furthermore, spatial pyramid matching of bag-of-words (BoW) models [50] in object recognition, has been adopted for object detection, providing robustness to large deformations. The computational cost has been alleviated by using thousands of object-independent candidate detection windows instead of millions of sliding windows, yet still inefficiently as it employed a large codebook to encode the features.

Owing to its ability to efficiently learn a descriptive and flexible object representation, Wang *et al.*'s regionlet-based detection framework [185] has gained a lot of attention. It provides the flexibility to deal with different scales and aspect ratios without performing exhaustive search. It first proposed the concept of ***regionlet*** by defining a three-level structural relationship: candidate bounding boxes (sliding windows), regions inside the bounding box and groups of regionlets (sub-regions inside each region). It operates by directly extracting features from regionlets in several selected regions within an arbitrary detection bounding box and performs (max) pooling among the regionlets. Such a feature extraction hierarchy is capa-

ble of dealing with variable aspect ratios and flexible feature sets, which leads to improved learning of robust feature representation of the object for region-based object detection.

Recently, deep learning has achieved significant success on many computer vision tasks such as image classification [87, 107], semantic segmentation [123] and object detection [67,68] Despite the superior performance of deep learning-based detection approaches, most network architectures [43,122,153] do not take advantage of successful conventional ideas such as DPM or regionlets. Those conventional methods have been shown to be effective for modeling object deformation, sub-categories and multiple aspect ratios. As deep convolutional neural networks [111] exhibit superior capability in learning hierarchical and discriminative features (deep features), it motivates us to think how to bridge the deep neural network and conventional object detection schemes.

Recent advances [44, 138, 142] have answered the question by combining the conventional DPM-based detectors with deep neural network architectures and achieving promising results. Yet few methods [186,226] have been explored for conventional regionlet detection schema. Zou *et al.* [226] made the preliminary attempt to utilize deep features instead of hand-crafted features. They introduced the dense neural pattern (DNP) to extract features from an image with arbitrary resolution using a well-trained deep convolutional neural network (*i.e* AlexNet [107]). Activations from same receptive fields but different feature maps can be extracted by back-tracking to exact coordinates in the original image. Though [226] presented effective performance boost with deep features, several limitations make DNP [226] not applicable

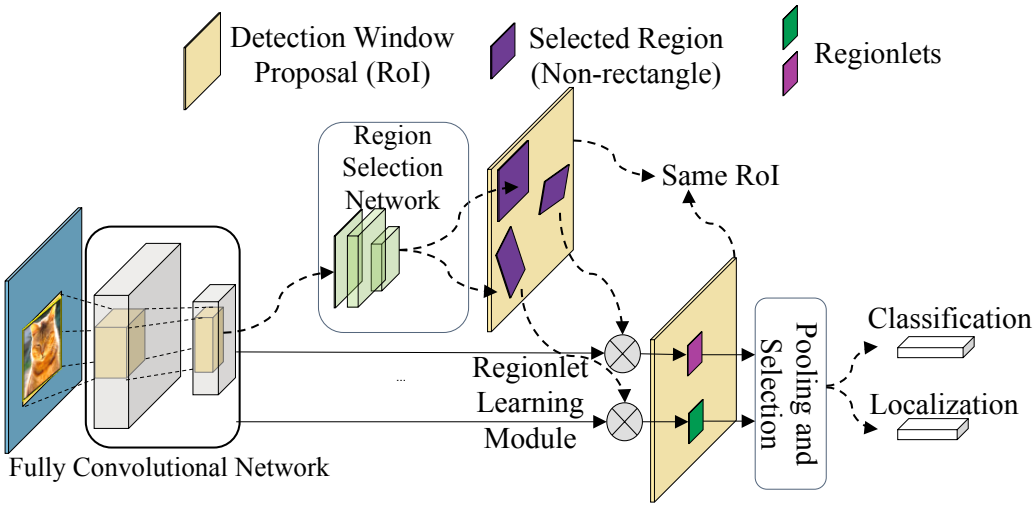


Figure 5.1: Architecture of the Deep Regionlets detection approach. It consists of a Region Selection Network (RSN) and a deep regionlet learning module. The region selection network performs *non-rectangular* region selection from the detection window proposal generated by the region proposal network. Deep regionlet learning module learns the regionlets through a spatial transformation and a gating network. The entire pipeline is end-to-end trainable. For better visualization, the region proposal network is not displayed here.

to deep neural networks developed recently (*i.e.* VGG [168], ResNet [87]). First, as DNP uses selective search to generate the region proposal, it would be extremely inefficient to extract the feature activations with more sophisticated deep neural networks. Second, end-to-end training is not feasible as DNP [226] directly extracted features on well-trained deep neural network for classification tasks.

These observations motivate us to establish a "real" bridge between deep convolutional neural networks and conventional *regionlet* object detection schema. In this chapter, we incorporate the conventional regionlet method into an *end-to-end* trainable deep learning framework. Despite being able to handle arbitrary bounding boxes, several drawbacks arise when directly integrating the regionlet methodology into the deep learning framework. First, both regionlet [185] and DNP [226] proposed to learn cascade object classifiers after hand-crafted/deep feature extraction in each regionlet, thus end-to-end learning is not feasible in both approaches. Second, regions in regionlet-based detection have to be rectangular, which does not effectively model object deformation. Moreover, both regions and regionlets are fixed after training is completed.

To this end, we propose a novel object detector named "**Deep Regionlets**" by blending deep learning and the traditional regionlet method [185, 226]. The proposed framework "Deep Regionlets" is able to address the limitations of both traditional regionlet and DNP extension, leading to significant precision improvement by exploiting the power of deep convolutional neural networks.

The overall design of the proposed detection system is illustrated in Figure 5.1. It consists of a region selection network and a deep regionlet learning module. The

region selection network (RSN) performs *non-rectangular* region selection from the detection window proposal[1] (RoI) to address the limitations of the traditional regionlet approach. We further design a deep regionlet learning module to learn the regionlets through a spatial transformation and a gating network. By using the proposed gating network, which is a soft regionlet selector, the final feature representation is more suitable for detection. The entire pipeline is end-to-end trainable using only the input images and ground truth bounding boxes.

We conduct a detailed analysis of our approach to understand its merits and properties. Extensive experiments on two detection benchmark datasets, PASCAL VOC [58] and Microsoft COCO [118] show that the proposed deep regionlet approach outperforms several competitors [43, 44, 138, 153, 213]. Even without segmentation labels, we outperform state-of-the-art algorithms Mask R-CNN [85] and RetinaNet [117].

To summarize, the major contributions of this chapter are four-fold:

- We propose a novel approach for object detection, "Deep Regionlets". Our work blends the traditional regionlet method and the deep learning framework. The system could be trained in a fully end-to-end manner.

- We design a region selection network, which **first** performs **non-rectangular** regions selection within the detection bounding box generated from a detection window proposal. It provides more flexibility in modeling objects with variable

[1]The detection window proposal is generated by a region proposal network [43, 68, 153]. It is also called region of interest (ROI)

shapes and deformable parts.

- We propose a deep regionlet learning module, including feature transformation and a gating network. The gating network serves as a soft regionlet selector and lets the network focus on features that benefit detection performance.

- We present empirical results on object detection benchmark datasets, which demonstrates the competitive performance over state-of-the-art.

5.2 Related Work

Generic object detection accuracy has improved over years. Such improvement is due to more effecive handling of multi-viewpoints, modeling deformations [61], and the success of deep learning techniques [67, 68, 87, 153, 168]. A complete survey of the object detection literature is beyond the scope of this dissertation.

Briefly speaking, many approaches based on traditional representations [61, 177, 185] and deep learning [20, 36, 43, 44, 64, 67, 68, 86, 89, 97, 97, 115, 122, 138, 151, 153, 190, 213, 217] have been proposed. Traditional approaches mainly used hand-crafted features (*i.e.* LBP [2], HOG) [45]) to design object detectors using the sliding window paradigm. One of the earliest works [177] used boosted cascaded detectors for face detection, which led to its wide adoption. Deformable part models [60] further extended the cascaded detectors to more general object categories. Due to the rapid development of deep learning techniques [87, 107, 168], the deep learning-based detectors have become dominant object detectors.

Deep learning-based detectors could be further categorized into two classes,

single-stage detectors and two-stage detectors, based on whether the detectors have proposal-driven mechanism or not. The single-stage detectors [64, 101, 105, 109, 116, 117, 122, 151, 160, 213, 217] apply regular, dense sampling windows over object locations, scales and aspect ratios. By exploiting multiple layers within a deep CNN network directly, the single-stage detectors achieved high speed but their detection accuracy was low compared to two-stage detectors.

Two-stage detectors [34–36, 43, 44, 68, 78, 85, 121, 138, 153, 169, 170, 225] first generate a sparse set of candidate proposals of detection bounding boxes by the region proposal network (RPN). After filtering out the majority of negative background boxes by RPN, the second stage classifies the detection bounding box proposals and performs regression to predict the object categories and their corresponding locations. The two-stage detectors consistently achieve higher accuracy than single-stage detectors and numerous extensions have been proposed [43, 44, 68, 85, 138, 153]. Our method follows the two-stage detector architecture by taking advantage of the region proposal network without the need for dense sampling of object locations, scales and aspect ratios.

5.3 Traditional Regionlets for Detection

In this section, we review traditional regionlet-based approach and its dense neural pattern extension as our work is directly motivated by the regionlet detection scheme. We incorporate *regionlet* into an end-to-end trainable deep learning framework. The proposed deep regionlets framework overcomes the limitations of

both traditional regionlet method [185] and the DNP [226], leading to significant improvement in detection performance.

5.3.1 Regionlets Definition

A *regionlet* is a base feature extraction region defined proportionally with respect to a sliding window or a detection bounding box) at arbitrary resolution (*i.e.* size and aspect ratio). Wang *et al.* [185, 186] first introduced the concept of regionlets, illustrated in Figure 5.2.

Figure 5.2 shows a three-level structure consisting of a detection bounding box, number of regions inside the bounding box and a group of regionlets (sub-regions inside each region). In Figure 5.2, the yellow box is a detection bounding box. A rectangular feature extraction region inside the detection bounding box is denoted as R (purple rectangle). Furthermore, within this region R, we spot some small sub-regions (small magenta rectangles) $r_{i\{i=1...N\}}$(*e.g.* r_1, r_2) and define them as a set of *regionlets*. One of the motivations behind the term *regionlet* is that the hand-crafted features extracted from these sub-regions will be aggregated into a single feature representation of R.

To summarize, one detection bounding box is represented by several regions, each of which consists of a small set of regionlets. By using the *relative* positions and sizes of regionlets and regions, the difficulty of the arbitrary detection bounding box has been well addressed. Therefore, the regionlet-based representation is able to model relative spatial layouts inside an object and can be efficiently applied to

arbitrary bounding boxes at different scales and aspect ratios. However, the initialization of regionlets possess randomness and both regions (R) and regionlets (*i.e.* r_1, r_2) are fixed after training is completed. Moreover, it is based on hand-crafted features (*i.e.* HOG [45] or LBP descriptors [2]) in each regionlet respectively. The proposed deep regionlet-based approach in Section 5.4 mitigates such limitations.

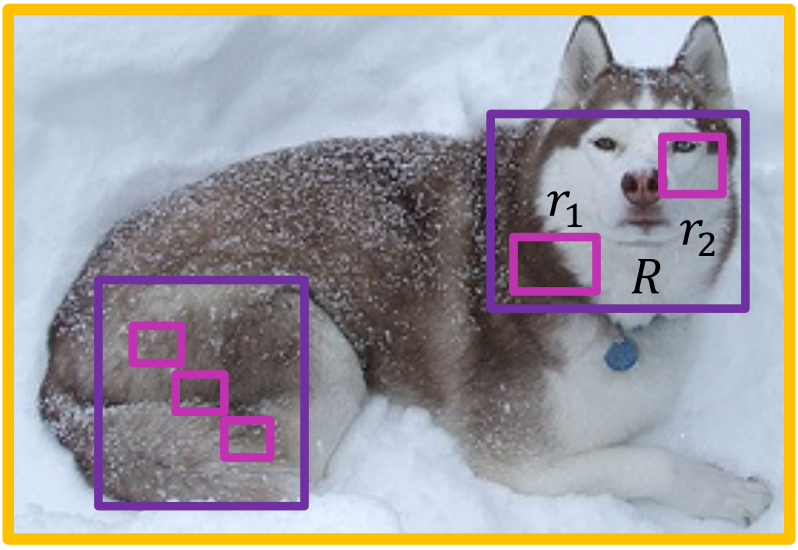


Figure 5.2: Illustration of structural relationships among the detection bounding box, feature extraction regions and regionlets. The yellow box is a detection bounding box and R is a feature extraction region shown as a purple rectangle with filled dots inside the bounding box. Inside R, two small sub-regions denoted as r_1 and r_2 are the *regionlets*.

5.3.2 Dense Neural Pattern Extension

Despite the success of the sophisticated regionlet detection method [185], the features employed are still hand-crafted representations such as LBP [2], HOG [45] or covariance-based on the gradients of the image.

Recently, the deep convolutional neural network has demonstrated promising performance on object detection [67, 68]. The dramatic improvements are due to hierarchically learning more complex features (deep features) from deep neural networks. One intuitive way to improve the traditional regionlet-based approach is to utilize deep features instead of hand-crafted features. Zou *et al.* [226] made the first attempt to incorporate the regionlet detection scheme in the deep neural network. [226] introduced DNP to extract features from an image with arbitrary resolution using a well trained classification-based deep convoluttional neural network (*i.e.* AlexNet [107]).

However, there are several limitations which make DNP [226] not applicable to recent deep neural networks (*i.e.* VGG [168], ResNet [87]). First, DNP [226] directly extracted features on well-trained deep neural network for classification task hence end-to-end training is not feasible. Second, it used the sliding window approach (*i.e.* selective search) to generate region proposals, which would become extremely inefficient with more sophisticated deep neural networks. The proposed deep regionlet-based approach overcomes both traditional regionlet method and DNP, leading to significant improvement in detection accuracy.

5.4 Deep Regionlets

In this section, We first present the overall design of the proposed deep regionlet approach with end-to-end training and then describe each module in detail.

5.4.1 System Architecture

Generally speaking, an object detection network performs a sequence of convolutional operations on an image of interest using a deep convolutional neural network. At some layer, the network bifurcates into two branches. One branch, RPN, generates a set of candidate bounding boxes[2] while the other branch performs classification and regression by pooling the convolutional features inside the proposed bounding box generated by RPN [43, 153]. Taking advantage of this detection network, we introduce the overall design of the proposed object detection framework, named "Deep Regionlets", as illustrated in Figure 5.1.

The general architecture consists of an RSN and a deep regionlet learning module. In particular, RSN is used to predict transformation parameters to select regions given a candidate bounding box, which is generated by the RPN. The regionlets are further learned within each selected region defined by the RSN. The system is designed to be trained in a fully end-to-end manner using only input images and the ground truth bounding box. The RSN as well as the regionlet learning module can be simultaneously learned over each selected region given the detection bounding box proposal.

5.4.2 Region Selection Network

We design the RSN to have the following properties:

- End-to-end trainable.

[2] [43, 68, 153] also called the detection bounding box as detection window proposal.

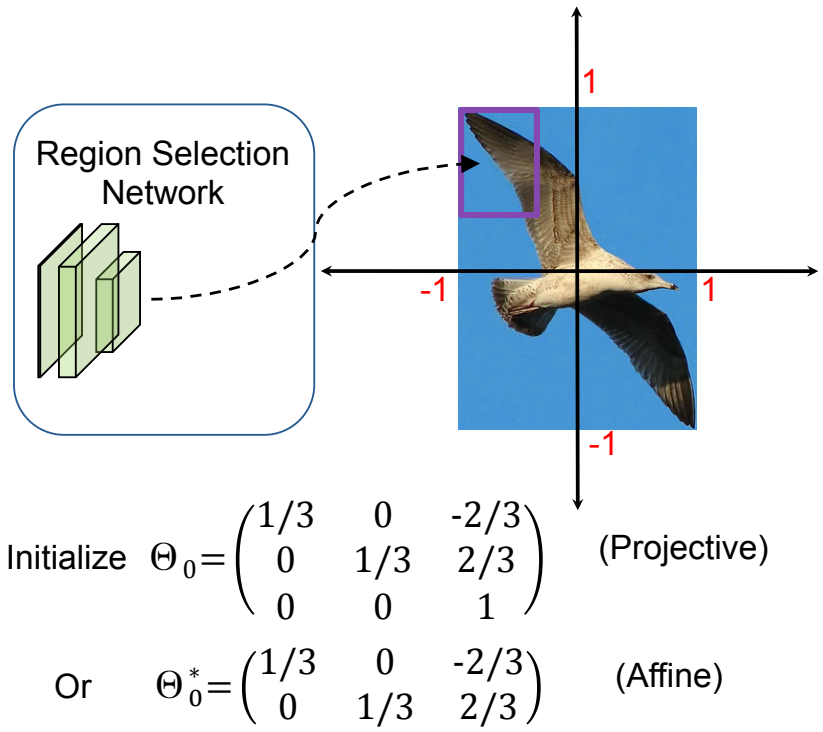


Figure 5.3: Initialization of one set of projective transformation parameters and affine transformation parameters. Normalized projective transformation parameters $\Theta_0 = [\frac{1}{3}, 0, -\frac{2}{3}; 0, \frac{1}{3}, \frac{2}{3}; 0, 0, 1]$ $(\theta_i \in [-1, 1])$ and affine transformation parameters $\Theta_0^* = [\frac{1}{3}, 0, -\frac{2}{3}; 0, \frac{1}{3}, \frac{2}{3}]$ $(\theta_i^* \in [-1, 1])$ selects the top-left region in the 3×3 evenly divided detection bounding box, shown as the purple rectangle.

- Possess simple structure without introducing too many parameters.
- Generate regions with arbitrary shapes.

Keeping these in mind, we design the RSN to predict a set of *projective* transformation parameters. By using these projective transformation parameters, as well as not requiring the regions to be rectangular, we have more flexibility in modeling an object with arbitrary shape and deformable parts.

Specifically, we design the RSN using a small neural network with three

fully connected layers. The first two fully connected layers have output size of 256, with ReLU activation. The last fully connected layer has the output size of nine, which is used to predict the set of projective transformation parameters $\Theta = [\theta_1, \theta_2, \theta_3; \theta_4, \theta_5, \theta_6; \theta_7, \theta_8, \theta_9]$. It is noted that in our preliminary work [196], RSN is only designed to predict the set of affine parameters $\Theta^* = [\theta_1, \theta_2, \theta_3; \theta_4, \theta_5, \theta_6]$

Note that the candidate detection bounding boxes proposed by RSN have arbitrary sizes and aspect ratios. In order to address this difficulty, we use the *relative* positions and sizes of the selected region within a detection bounding box. The candidate bounding box generated by the region proposal network is defined by the top-left point (w_0, h_0), width w and height h of the box. We normalize the coordinates by the width w and height h. As a result, we could use the normalized projective transformation parameters $\Theta = [\theta_1, \theta_2, \theta_3; \theta_4, \theta_5, \theta_6; \theta_7, \theta_8, \theta_9]$ ($\theta_i \in [-1, 1]$) to evaluate one selected region within one candidate detection window at different sizes and aspect ratios without scaling images into multiple resolutions or using multiple-components to enumerate possible aspect ratios, like anchors [64, 122, 153].

5.4.2.1 Initialization of Region Selection Network

Taking advantage of the *relative* and *normalized* coordinates, we initialize the RSN by equally dividing the whole detecting bounding box to several sub-regions, named as *cell*s, without any overlap among them.

Figure 5.3 shows an example of initialization from one projective transforma-

tion and one affine transformation in [196]. (*i.e.* 3×3). The first cell, which is the top-left bin in the whole region (detection bounding box) could be defined by initializing the corresponding projective transformation parameters $\Theta_0 = [\frac{1}{3}, 0, -\frac{2}{3}; 0, \frac{1}{3}, \frac{2}{3}; 0, 0, 1]$ or affine transformation parameters $\Theta_0^* = [\frac{1}{3}, 0, -\frac{2}{3}; 0, \frac{1}{3}, \frac{2}{3}]$. The other eight of 3×3 cells are initialized in a similar way.

5.4.3 Deep Regionlet Learning

After regions are selected by the RSN, regionlets are further learned from the selected region defined by the normalized projective (affine) transformation parameters. Note that our motivation is to design the network to be trained in a fully end-to-end manner using only input images and ground truth bounding boxes. Therefore, both the selected regions and regionlet learning should be able to be trained by deep neural networks. Moreover, we would like the regionlets extracted from the selected regions to better represent objects with variable shapes and deformable parts.

Inspired by the spatial transform network [95, 222], any parameterizable transformation including translation, scaling, rotation, affine or even projective transformation can be learned by a spatial transformer. In this section, we introduce our deep regionlet learning module to learn the regionlets in the selected region, which is defined by the projective transformation parameters. It is noted that affine transformation is the special case of projective transformation obtained by setting $\theta_7 = 0$, $\theta_8 = 0$, $\theta_9 = 1$ in Θ.

More specifically, we aim to learn regionlets from one selected region defined by one set of projective transformation Θ to better match the shapes of objects. This is done with a selected region R from RPN, transformation parameters $\Theta = [\theta_1, \theta_2, \theta_3; \theta_4, \theta_5, \theta_6; \theta_7, \theta_8, \theta_9]$ predicted by RSN and a set of feature maps $U = \{U_i, i = 1, \ldots, n\}$. Without loss of generality, let U_i be one of the feature map out of the n feature maps. A selected region R is of size $w \times h$ with the top-left corner (w_0, h_0). Inside the U_i feature map, we present the regionlet learning module as follows:

Let s denote the source and t denote target, we define (x_p^s, y_p^s) as the p-th spatial location in the original feature map U_i and (x_p^s, y_p^s) as the corresponding spatial location in the output feature maps after projective transformation. First, a grid generator [95] generates the source map coordinates $(x_p^s, y_p^s, 1)$ based on the transformation parameters, given the p-th spatial location $(x_p^t, y_p^t, 1)$ in target feature maps. The process is generated using (5.1) given below.

$$\begin{pmatrix} x_p^s \\ y_p^s \\ 1 \end{pmatrix} = \frac{1}{z_p^s} \begin{pmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_6 \\ \theta_7 & \theta_8 & 1 \end{pmatrix} \begin{pmatrix} x_p^t \\ y_p^t \\ 1 \end{pmatrix} \tag{5.1}$$

where $\theta_9 = 1$ and $z_p^s = \theta_7 x_p^t + \theta_8 y_p^t + 1$ to ensure the third dimension of $(x_p^s, y_p^s, 1)$ is 1.

Next, the sampler samples the input feature U at the generated source coordinates. Let U_{nm}^c be the value at location (n, m) in channel c of the input feature. The total output feature map V is of size $H \times W$. $V(x_p^t, y_p^t, c|\Theta, R)$ be the output feature value at location (x_p^t, y_p^t) ($x_p^t \in [0, H]$, $y_p^t \in [0, W]$) in channel c, which is

computed as

$$V(x_p^t, y_p^t, c|\Theta, R) = \sum_n^H \sum_m^M U_{nm}^c \max(0, 1 - |x_p^s - m|) \max(0, 1 - |y_p^s - n|) \quad (5.2)$$

5.4.3.1 Back Propagation through Spatial Transform

To allow back propagation of the loss through the regionlet learning module, we can define the gradients with respect to the feature maps and the region selection network. In this layer's `backward` function, we have partial derivative of the loss function with respect to the feature map variable U_{mn}^c and projective transform parameter $\Theta = [\theta_1, \theta_2, \theta_3; \theta_4, \theta_5, \theta_6; \theta_7, \theta_8, \theta_9]$. Motivated by [95], the partial derivative of the loss function with respect to the feature map is:

$$\frac{\partial V(x_p^t, y_p^t, c|\Theta, R)}{\partial U_{nm}^c} = \sum_n^H \sum_m^M \max(0, 1 - |x_p^s - m|) \times \max(0, 1 - |y_p^s - n|) \quad (5.3)$$

Moreover, during back propagation, we need to compute the gradients with respect to the projective transformation parameter vector $\Theta = [\theta_1, \theta_2, \theta_3; \theta_4, \theta_5, \theta_6; \theta_7, \theta_8, 1]$. Note that we set $\theta_9 = 1$ in (5.1) hence we only need to calculate the gradient of V with respect to eight projective parameters. In this way, RSN could also be updated to adjust the selected region. Although (5.2) may not be differentiable when $x_p^s = m$ or $y_p^s = n$, this seldom happens in practice because the possibility that the calculated x_p^s or y_p^s are integers is extremely low. We empirically set the gradients at these points to be 0 as their effect on the back propagation process is negligible.

We consider θ_1 and θ_7 as examples due to space limitations and similar derivative can be computed for other parameters $\theta_i (i = 2, \ldots, 6, 8)$ respectively (See Appendix for a complete derivation). Denote $V(x_p^t, y_p^t, c|\Theta, R)$ as V_p for simplicity, after applying the chain rule for the differentiable points:

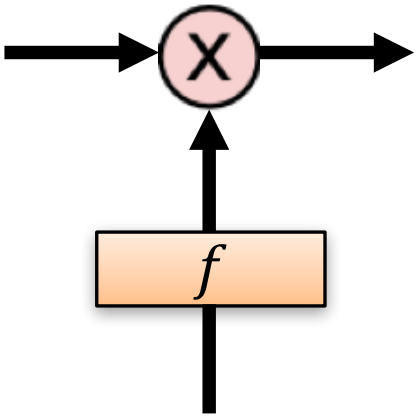


Figure 5.4: Design of the gating network. f denotes the non-negative gate function (*i.e.* sigmoid)

$$\begin{aligned}
\frac{\partial V_p}{\partial \theta_1} &= \frac{\partial V_p}{\partial x_p^s}\frac{\partial x_p^s}{\partial \theta_1} = \frac{x_p^t}{z_p^s}\frac{\partial V_p}{\partial x_p^s} \\
\frac{\partial V_p}{\partial \theta_7} &= \frac{\partial V_p}{\partial z_p^s}\frac{\partial z_p^s}{\partial \theta_7} = (\frac{\partial V_p}{\partial x_p^s}\frac{\partial x_p^s}{\partial z_p^s} + \frac{\partial V_p}{\partial y_p^s}\frac{\partial y_p^s}{\partial z_p^s})x_p^t \\
&= -\frac{x_p^t}{z_p^s}(\frac{\partial V_p}{\partial x_p^s}x_p^s + \frac{\partial V_p}{\partial y_p^s}y_p^s)
\end{aligned} \tag{5.4}$$

where

$$\begin{aligned}
\frac{\partial V_p}{\partial x_p^s} &= \sum_n^H \sum_m^M U_{nm}^c \max(0, 1 - |y_p^s - n|)\eta(x_p^s - m) \\
\eta(x_p^s - m) &= \begin{cases} 0, & \text{if } |m - x_p^s| \geq 1 \\ 1, & \text{if } m > x_p^s \\ -1, & \text{if } m < x_p^s \end{cases}
\end{aligned} \tag{5.5}$$

It is worth noting that (x_p^t, y_p^t) are normalized coordinates in range $[-1, 1]$ so that they can to be scaled with respect to w and h with start position (w_0, h_0).

5.4.3.2 Gating Network

The gating network, which serves as a soft regionlet selector, is used to assign regionlets with different weights and generate regionlet feature representation. We design a simple gating network using a fully connected layer with `sigmoid` activation. The output values of the gating network are in the range $[0, 1]$. Given the output feature maps $V(x_p^t, y_p^t, c|\Theta, R)$ described above, we use a fully connected layer to generate the same number of outputs as feature maps $V(x_p^t, y_p^t, c|\Theta, R)$, which is followed by an activation layer `sigmoid` to generate the corresponding weight respectively. The final feature representation is generated by the product of feature maps $V(x_p^t, y_p^t, c|\Theta, R)$ and their corresponding weights.

5.4.3.3 Regionlet Pool Construction

Object deformations may occur at different scales. For instance, deformation could be caused by different body parts in person detection. Same number of regionlets (size $H \times W$) learned from small selected region have higher extraction density, which may lead to non-compact regionlet representation. In order to learn a *compact* and *efficient* regionlet representation, we further perform the pooling (*i.e.* max/ave) operation over the feature maps $V(x_p^t, y_p^t, c|\Theta, R)$ of size $(H \times W)$.

We reap two benefits from the pool operation: (1) Regionlet representation is

compact (small size). (2) Regionlets learned from different size of selected regions are able to represent such regions in an efficient way, and handle object deformations at different scales.

5.5 Relations to Recent Works

We review the traditional regionlet-based approach and its DNP extension in Section 5.3. Besides this, our deep regionlet approach is related to some recent object detection works in different aspects. In this section, we discuss both similarities and differences in detail.

5.5.1 Spatial Transform Networks

Jaderberg *et al.* [95] first proposed the spatial transformer module to provide spatial transformation capabilities into a deep neural network. It only learns *one global parametric transformation* (scaling, rotations as well as affine transformation). Such learning is known to be difficult to apply on semi-dense vision tasks (*e.g.*, object detection) and the transformation is on the entire feature map, which means the transformation is applied identically across all the regions in the feature map.

The proposed RSN learns a set of projective transformation and each transformation can be considered as the localization network in [95]. However, regionlet learning is different from image sampling [95] method as it adopts a region-based spatial transformation and feature wrapping. By learning the transformation locally in the detection bounding box, our method provide the flexibility of learning a com-

pact, efficient feature representation of objects with variable shape and deformable parts.

5.5.2 Deformable Part Model and its deep learning extensions

Deformable part models [60] explicitly represent spatial deformations of object parts via latent variables. A root filter is learned to model the global appearance of the objects, while the part filters are designed to describe the local parts in the objects. However, DPM is a shallow model and the training process involves heuristic choices to select components and part sizes, making end-to-end training inefficient.

Both works [44, 138] extend the DPM with end-to-end training in deep CNNs. Motivated by DPM [61] to allow parts to slightly move around their reference position (partition of the initial regions), they share the similar idea of learning part offsets[3] to model the local element and pool the features at their corresponding locations after the shift. While [44, 138] show promising improvements over other deep learning-based object detectors [68, 153], it still lacks the flexibility of modeling non-rectangular objects with sharp shapes and deformable parts.

It is noticeable that the deep regionlet learning module in the proposed method on the selected region is a generalization of Deformable RoI Pooling in [44, 138]. First, we generalize the selected region to be non-rectangular by learning the projective transformation parameters. Such non-rectangular regions could provide the capabilities of *scaling*, *shifting* and *rotation* around the original reference region. If

[3] [44] uses term offset while [138] uses term displacement

we only enforce the RSN to learn the shift, our regionlet learning mechanism would degenerate to deformable RoI pooling as in [44, 138]

5.5.3 Spatial-based RoI Pooling

Traditional spatial pyramid pooling [110] performs pooling over hand crafted regions at different scales. With the help of deep CNNs, [86] proposes to use spatial pyramid pooling in deep learning-based object detection. However, as the pooling regions over image pyramid still need to be carefully designed to learn the spatial layout of the pooling regions, the end-to-end training strategy is not well facilitated. In contrast, the deep regionlet learning approach learns pooling regions end-to-end in deep CNNs. Moreover, the region selection step for learning regionlets accommodates different sizes of the regions. Hence, we are able to handle object deformations at different scales without generating the feature pyramid.

5.6 Experiments

In this section, we present comprehensive experimental results of the proposed approach on two challenging benchmark datasets: PASCAL VOC [58] and MS-COCO [118]. There are in total 20 categories of objects in PASCAL VOC [58] dataset, which includes rigid objects such as cars and deformable objects like cats. We follow the common settings used in [18, 43, 68, 153] to draw compelete comparsions. More specifically, we train our deep model on (1) VOC 2007 `trainval` and (2) union of VOC 2007 `trainval` and VOC 2012 `trainval` and evaluate on the VOC

2007 `test` set. We also report results on the VOC 2012 `test` set with the model trained on the VOC 2007 `trainvaltest` and VOC 2012 `trainval`. In addition, we report results on the VOC 2007 `test` split for ablation analysis. Mean average precision (mAP) is reported for all the experiments on PASCAL VOC.

MS-COCO [118] is a widely used challenging dataset, which contains 80 object categories. Following the official settings in COCO website[4], we use the COCO 2017 `trainval` split (union of 135k images from `train` split and 5k images from `val` split) for training. We report the COCO-style average precision (mmAP) on `test-dev` 2017 split, which requires evaluation from the MS-COCO server[5] for testing.

For the base network, we use both VGG-16 [168] and ResNet-101 [87] to demonstrate the generalization of our approach regardless of which network backbone we use. The *á trous* algorithm [123, 135] is adopted in stage 5 of ResNet-101. Following the suggested settings in [43, 44], we also set the pooling size to 7 by changing the conv5 stage's effective stride from 32 to 16 to increase the feature map resolution. In addition, the first convolution layer with stride 2 in the conv5 stage is modified to 1. Both backbone networks are intialized with the pre-trained ImageNet [87, 107] model.

In the following sections, we report the results of a series of ablation experiments to understand the behavior of the proposed deep regionlet approach. Furthermore, we present comparisons with state-of-the-art detectors [43, 44, 85, 116, 117, 153]

[4] `http://cocodataset.org/#detections-challenge2017`

[5] The updated settings (2017) are different from the previous settings (2016, 2015) in [18, 43, 44, 117, 117], as it includes different train/val sets.

Methods	Regionlet [185, 186]	DNP [226]	Ours-A (AlexNet)
mAP@0.5(%)	41.7	46.1	63.2
Methods	Ours-A (VGG16)	Ours-A (ResNet-50)	Ours-A (ResNet-101)
mAP@0.5(%)	73.0	74.2	75.3

Table 5.1: Ablation studies on the improvement of the proposed deep regionlets method over traditional *regionlets* [185, 186] and its extension DNP [226]. Results are reported on different network architecture backbones, *i.e* AlexNet [107], VGG16 [168] ResNet-50 [87] and ResNet-101 [87]. Ours-A denotes RSN predicting affine transformation parameters.

on both PASCAL VOC [58] and MS COCO [118] datasets.

5.6.1 Ablation Study on PASCAL VOC

In this section, we comprehensively evaluate the deep regionlets method on the PASCAL VOC [58] detection benchmark to understand its behavior. Unless otherwise stated, all ablation studies are performed with RSN predicting the affine transformation parameters in the proposed approach.

5.6.1.1 Comparison with the conventional Regionlets detection schema

We first evaluate how the deep regionlets approach improves over the conventional *Regionlets* detection schema [185,186] and its DNP extension [226]. It is noted that DNP [226] first attempted to utilize the deep features using AlexNet [107]. In order to draw a fair comparison, we train our model on VOC 2007 `trainval` and

Methods	Global RSN	Offset-only RSN [44, 138]	Non-gating	Ours-A
mAP@0.5(%)	30.27	78.5	81.3 (+2.8)	82.0 (+3.5)

Table 5.2: Ablation studies of each component in deep regionlet approach. Output size $H \times W$ is set to 4×4 for all the baselines. Ours-A denotes RSN predicting affine transformation parameters.

# of Regions \ Regionlets Density	2×2	3×3	4×4	5×5	6×6
$4(2 \times 2)$ regions	78.0	79.2	79.9	80.2	80.3
$9(3 \times 3)$ regions	79.6	80.3	80.9	81.5	81.3
$16(4 \times 4)$ regions	80.0	81.0	82.0	81.6	80.8

Table 5.3: Results of ablation studies when an RSN selects different number of regions and regionlets are learned at different level of density.

evaluate on the VOC 2007 `test` set using AlexNet [107]. The shorter side of image is set to be 600 pixels and training is performed for 60k iterations with single mini-batch size on 1 GPU. The learning rate is set at 10^{-3} for the first 40k iterations and 10^{-4} for the remaining 20k iterations.

Results of improvements over traditional *regionlets* [185, 186] and DNP [226] are shown in Table 5.1. First, although DNP [226] improved over traditional *regionlet* [185, 186] by almost 5% with the help of deep features, our approach provide huge improvement over both traditional regionlet [185, 186] and DNP [226] (more than **20**% in terms of mAP) with the power of end-to-end trainable framework. Second, the mAP can be significantly increased by using deeper and more powerful

networks like ResNet-50 and ResNet-101 [87]. All these observations support the effectiveness and success of the integration of traditional regionlet method into the end-to-end trainable deep learning framework.

5.6.1.2 Ablation study on each component

Next, we investigate the proposed approach to understand each component and its behavior. For a fair comparison, we adopt ResNet-101 as the backbone network for ablation studies. We train our model on the union set of VOC 2007 + 2012 `trainval` as well as their horizontal flip and evaluate on the VOC 2007 `test` set. The shorter side of image is set at 600 pixels, as suggested in [43, 44, 68, 153]. The training is performed for 60k iterations with an effective mini-batch size of 4 on 4 GPUs, where the learning rate is set at 10^{-3} for the first 40k iterations and at 10^{-4} for the rest 20k iterations. We investigate the proposed approach to understand each component (1) RSN, (2) Deep regionlet learning and (3) Soft regionlet selection by comparing it with several baselines:

1. Global RSN. RSN only selects one global region and it is initialized as identity affine transformation (*i.e.* $\Theta_0^* = [1, 0, 0; 0, 1, 0]$). This is equivalent to global regionlet learning within the RoI.

2. Offset-only RSN. We set the RSN to only learn the offset by enforcing $\theta_1, \theta_2, \theta_4, \theta_5$ (in affine parameters) to not change during the training process. In this way, the RSN only selects the rectangular region with offsets to the initialized region. This baseline is similar to the Deformable RoI Pooling in [44] and

[138].

3. Non-gating selection: deep regionlet without soft selection. No soft regionlet selection is performed after the regionlet learning. In this case, each regionlet learned has the same contribution to the final feature representation.

Results are shown in Table 5.2. First, when the RSN only selects one global region, the RSN reduces to the single localization network [95]. In this case, regionlets are extracted in a global manner. It is interesting to note that selecting only one region by the RSN is able to converge, which is different from [43, 153]. However, the performance is extremely poor. This is because no discriminative regionlets could be explicitly learned within the region. More importantly, the results clearly demonstrate that RSN is ***indispensable*** in the deep regionlet approach.

Moreover, offset-only RSN could be viewed as similar to deformable RoI pooling in [44,138]. These methods all learn the offset of the rectangle region with respect to its reference position, which lead to improvement over [153]. However, non-gating selection outperforms offset-only RSN by 2.8% with selecting non-rectangular region. The improvement demonstrates that non-rectangular region selection could provide more flexibility around the original reference region, thus could better model non-rectangular objects with sharp shapes and deformable parts. Last but not least, by using the gate function to perform soft regionlet selection, the performance can be further improved by 0.7%.

Next, we present ablation studies on the following questions in order to understand more deeply the RSN and regionlet learning modules. We report the results

where the RSN predicts the affine transformation parameters:

1. How many regions should we learn by RSN?

2. How many regionlets should we learn in one selected region (density is of size $H \times W$)?

5.6.1.3 How many regions should we learn by RSN?

We investigate how the detection performance varies when different number of regions are selected by the RSN. All the regions are initialized as described in Section 5.4.2 without any overlap between regions. Without loss of generality, we report results for $4(2 \times 2)$, $9(3 \times 3)$ and $16(4 \times 4)$ regions in Table 5.3. We observe that the mean AP increases when the number of selected regions is increased from $4(2 \times 2)$ to $9(3 \times 3)$ for fixed regionlets learning number, but gets saturated with $16(4 \times 4)$ selected regions.

5.6.1.4 How many regionlets should we learn in one selected region?

Next, we investigate how the detection performance varies when different number of regionlets are learned in one selected region by varying H and W. Without loss of generality, we set $H = W$ throughout our experiments and vary the H from 2 to 6. In Table 5.3, we report results when we set the number of regionlets at $4(2 \times 2)$, $9(3 \times 3)$, $16(4 \times 4)$, $25(5 \times 5)$, $36(6 \times 6)$ before the regionlet pooling construction.

First, it is observed that increasing the number of regionlets from $4(2 \times 2)$ to $25(5 \times 5)$ results in improved performance. As more regionlets are learned from

one region, more spatial and shape information from objects could be learned. The proposed approach could achieve the best performance when regionlets are extracted at $16(4 \times 4)$ or $25(5 \times 5)$ density level. It is also interesting to note that when the density increases from $25(5 \times 5)$ to $36(6 \times 6)$, the performance degrades slightly. When the regionlets are learned at a very high density level, some redundant spatial information may be learned without being useful for detection, thus affecting the region proposal-based decision to be made. Throughout all the experiments, we report the results from 16 selected regions from RSN and set output size $H \times W = 4 \times 4$.

5.6.2 Experiments on PASCAL VOC

In this section, we present experimental results on PASCAL VOC dataset and compare our results thoroughly with several methods described below:

- Traditional regionlet method [185] and DNP [226].
- Popular deep learning-based object detectors: Faster R-CNN [153], SSD [122], R-FCN [43], soft-NMS [18], DP-FCN [138] and DF-RCNN/D-RFCN [44].
- State-of-the-art deep learning-based object detectors: MLKP [179], RefineDet [213], PAD [219], DES [217] and STDN [223], RFB-Net [121], PFPNet-R [101], C-FRCNN [34], DFPN-R [105].

Methods	training data	mAP@0.5(%)	training data	mAP@0.5(%)
Regionlet [185]	07	41.7	07 + 12	N/A
DNP [226]	07	46.1	07 + 12	N/A
Faster R-CNN [153]	07	70.0	07 + 12	73.2
R-FCN [43]	07	69.6	07 + 12	76.6
SSD512 [122]	07	71.6	07 + 12	76.8
Soft-NMS [18]	07	71.1	07 + 12	76.8
Ours-A	07	73.0	07 + 12	79.2
Ours-P	07	73.3	07 + 12	79.5
Ours-A§	07	73.8	07 + 12	80.1
Ours-P§	07	73.9	07 + 12	80.3

Table 5.4: Detection results on PASCAL VOC2007 using VGG16 as backbone architecture. Training data: "07": VOC2007 `trainval`, "07 + 12": union set of VOC2007 and VOC2012 `trainval`. Ours-A(Ours-P)§ denotes applying the soft-NMS [18] in the test stage.

5.6.2.1 PASCAL VOC 2007

We follow the standard settings as in [18, 43, 44, 153] and report mAP scores using IoU thresholds at 0.5.

We first evaluate the proposed deep regionlet approach on the small training dataset VOC 2007 `trainval`. For the training stage, we set the learning rate at 10^{-3} for the first 40k iterations, then decrease it to 10^{-4} for the next 20k iterations with single GPU. Next, we evaluate our approach on a large training dataset,

Methods	mAP@0.5(%)	Year	Methods	mAP@0.5(%)	Year
Fast R-CNN [68]	70.0	2015	Faster R-CNN [153]	78.1	2016
OHEM [164]	74.6	2016	SSD* [122]	77.1	2016
ION [14]	79.4	2016	DP-FCN [138]	78.1	2017
DF-RCNN(ROI Pooling) [44]	78.3	2017	DF-RCNN [44]	79.3	2017
D-RFCN(ROI Pooling) [44]	81.2	2017	D-RFCN† [44]	82.6	2017
MR-CNN [65]	78.2	2015	LocNet [66]	78.4	2016
DSSD [64]	78.6	2017	PAD [219]	79.5	2018
MLKP [179]	80.6	2018	DES* [217]	81.7	2018
RefineDet [213]	80.0	2018	STDN [223]	80.9	2018
RFB-Net [121]	82.2	2018	PFPNet-R [101]	82.3	2018
C-FRCNN [34]	82.2	2018	DFPN-R [105]	82.4	2018
Ours-A	82.2		Ours-A§	83.1	
Ours-P	***82.5***		Ours-P§	**83.3**	

Table 5.5: Detection results on PASCAL VOC2007 test set. For a fair comparison, we only list the results of single model without multi-scale training/testing, ensemble, iterative bounding box regression or additional segmentation label. Training data: union set of VOC 2007 and 2012 `trainval`. *: the results are reported using new data augmentation trick. D-RFCN†: this entry is obtained from [44] using OHEM [164]. Ours-A(Ours-P)§ denotes we apply the soft-NMS [18] in the test stage.

created by merging VOC 2007 and VOC 2012 `trainval`. Due to using more training data, the number of iterations is increased. More specifically, we perform the same training process as described in Section 5.6.1. Moreover, we use 300 RoIs at test stage from a single-scale image testing with setting the image shorter side to

be 600. It is noted that for a fair comparison, we do not deploy the multi-scale training/testing, ensemble, iterative bounding box regression, online hard example mining(OHEM) [164], although it is shown in [18,44] that such enhancements could lead to additional performance boost. We report our results from RSN predicting both projective transformation parameters (Ours-P) and affine transformation parameters (Ours-A).

PASCAL VOC 2007 using VGG16 Backbone The results on VOC2007 `test` using VGG16 [168] backbone are summarized in Table 5.4. We first compare with traditional regionlet method [185], DNP [226] and several popular object detectors [18,122,153] when training using small size dataset (VOC2007 `trainval`). Next, we evaluate our method as we increase the training dataset (union set of VOC 2007 and 2012 `trainval`). With the power of deep CNNs, the deep regionlet approach has significantly improved the detection performance over the traditional regionlet method [185] and DNP [226]. We also observe that more data always helps. Moreover, it is encouraging that soft-NMS [18] is only applied in the test stage without modification in the training stage, which could directly improve over [153] by 1.1%. In summary, our method is consistently better than all the compared methods and the performance could be further improved if we replace NMS with soft-NMS [18].

PASCAL VOC 2007 using ResNet-101 Backbone Next, we change the network backbone from VGG16 [168] to ResNet-101 [87] and present corresponding results in Table 5.5. This is also the common settings for evaluating deep learning-based object detectors. Besides basic object detection framework, Faster R-CNN [153], SSD [122], R-FCN [43], soft-NMS [18], we also compare with our direct

competitors DF-RCNN/D-RFCN [44] and DP-FCN [138] as discussed in Section 5.5. In addition, we also compare with most recent object detectors, MLKP [179], RefineDet [213], PAD [219], DES [217] and STDN [223] [6].

First, compared to the performance in Table 5.4 using VGG16 [168] backbone architecture, the mAP can be significantly increased by using deeper networks like ResNet-101 [87]. Second, we outperform DP-FCN [138] and DF-RCNN [44] by 4.4% and 3.2% respectively. This provides the empirical support that our deep regionlet learning method could be treated as a *generalization* of Deformable RoI Pooling in [44, 138], as discussed in Section 5.5. Moreover, the results demonstrate that selecting *non-rectangular* regions from our method provide more capabilities including *scaling*, *shifting* and *rotation* to learn the feature representations.

Furthermore, we compare the proposed deep regionlet approach with the most recent published methods, PAD [219], MLKP [179], DES* [217], RefineDet [213] and STDN [223], RFB-Net [121], PFPNet-R [101], C-FRCNN [34], DFPN-R [105]. It can be seen that our method outperforms all the recent published methods including DES* [217] (0.8%), which used new data augmentation trick described in SSD* [122]. Such a trick is proven to boost the performance as shown in [64, 122, 217]. It is also noted that D-RFCN [44] reported 82.6% using OHEM [164] while we do not deploy OHEM. We achieve comparable result compared to D-RFCN [44] that uses OHEM. In summary, our method achieves state-of-the-art performance on object detection task when using ResNet-101 as backbone network. Note that all the other

[6] To the best of our knowledge, we report the results from original papers under the same settings. Some papers reported best results using different networks and settings.

results [14,34,65,66,101,105,121,179,213,219,223] are reported without using extra training data (*i.e.* COCO data), multi-scale training/testing [170], OHEM, ensemble or other post processing techniques.

Complete Object Detection Results We present the complete object detection results of the proposed deep regionlet method on the PASCAL VOC 2007 `test` set. Other results are reported from either the updated model [64, 122], the complete detection results reported in the paper [34, 105] or the official code provided by the authors with suggested settings [43,44,153,213] [7]. Note that we produce slightly lower performance 81.4% than 82.6% reported in [44]. The difference may come from sampling order of the images from the training set. Keeping this in mind, it is observed that our method achieves best average precision on majority on all the 20 classes in VOC 2007 `test` set.

5.6.2.2 PASCAL VOC 2012

We also present our results evaluated on VOC 2012 `test` set in Table 5.7. We follow the same experimental settings as in [43,64,122,138,153] and train our model using VOC"07++12", which consists of VOC 2007 `trainvaltest` and VOC 2012 `trainval` set. It can be seen that our method achieves state-of-the-art performance. In particular, we outperform DP-FCN [138], which further proves the generalization of our method over Deformable ROI Pooling in [138].

[7] We only list the methods, which either reported complete detection results on VOC 2007 or the code is public available

5.6.3 Experiments on MS COCO

In this section, we evaluate the proposed deep regionlet approach on the MS COCO [118] dataset and compare with other state-of-the-art object detectors: Faster R-CNN [153], SSD [122], R-FCN [43], Deformable F-RCNN/R-FCN [44], Mask R-CNN [85], RetinaNet [117].

We adopt ResNet-101 [87] as the backbone architecture of all the methods for a fair comparison. Following the settings in [43,44,85,117], we set the shorter edge of the image to 800 pixels. The training is performed for 280k iterations with effective mini-batch size 8 on 8 GPUs. We first train the model with 10^{-3} learning rate for the first 160k iterations, followed by learning rate 10^{-4} and 10^{-5} for another 80k iterations and the next 40k iterations. Five scales and three aspect ratios are used for anchors. We report results using either the released models or the code from the original authors. It is noted that we only deploy single-scale image training (no scale jitter) without the iterative bounding box average throughout all the experiments, although these enhancements could further boost performance.

Table 5.8 shows the results on 2017 `test-dev` set[8], which contains $20,288$ images. Compared with the baseline methods Faster R-CNN [153], R-FCN [43] and SSD [122], both Deformable F-RCNN/R-FCN [44] and our method provide huge improvements over [43,122,153] (+3.7% and +8.5%). Moreover, it can be seen that our method outperform Deformable F-RCNN/R-FCN [44] by wide margin(∼4%).

[8]MS COCO server does not accept 2016 and 2015 test-dev evaluation. As a result, we are not able to report results on 2016, 2015 test-dev set.

This observation further supports that our deep regionlet learning module could be treated as a *generalization* of [44, 138], as discussed in Section 5.5. It is also noted although the most recent state-of-the-art object detectors based on Mask R-CNN[9] [85] also utilized multi-task training with segmentation labels, we still improve over [85] by 1.1%. In addition, the main contribution focal loss in [117], which overcomes the obstacle caused by the imbalance of positive/negative samples, is complimentary to our method. We believe it can be applied in our method to further boost the performance. In summary, compared with Mask R-CNN [85], RetinaNet[10] [117] and other recent detectors [20, 89, 213, 225], our method achieves competitive performance compared to state-of-the-arts on MS COCO when using ResNet-101 as a backbone network.

5.6.4 Complexity Analysis: Parameters and Speed

In this section, we present the analysis on the speed and parameter of the proposed deep regionlets approach.

Runtime Speed: We evaluate the runtime of our approach and compare with other two-stage object detectors, Faster R-CNN [153], R-FCN [43] using the original Caffe implementation and ResNet-101 backbone with Batch Normalization(BN) layers for a fair comparison. The time is reported on single Nvidia TITAN X GPU

[9]Note [85] reported best result using ResNeXt-101-FPN [195]. We only compare the results in [85] using ResNet-101 [87] backbone for fair comparison.

[10] [117] reported best result using multi-scale training for 1.5× longer iterations. We only compare the results without scale jitter during training.

including image resizing, network forward and post-processing. On average, Faster R-CNN [153] takes 0.37s and R-FCN [43] takes 0.24s per image, while our method take 0.49s per image. In addition, we also report the runtime for DF-RCNN/D-RFCN [44] on the same machine configuration for reference purpose. DF-RCNN takes about 0.34s and D-RFCN takes 0.25s per image, note that DF-RCNN/D-RFCN [44] uses a different MXNet framework instead of Caffe and some python layers in RPN have been optimized with CUDA implementation.

Number of Parameters: The RSN has three fully connected layers (First two connected layer have output size of 256, last fully connected layer has the output size of 9), giving about 5.28M ($16 \times (1024 \times 256 + 256 \times 256 + 256 \times 9)$) parameters, while deep regionlet learning module and gating network do not introduce new parameters. According to [43, 89, 153, 219], in total, Faster R-CNN has about 58.3M parameters, R-FCN has about 56.4M parameters. Therefore, the total number of parameters is about 63.6M on top of Faster R-CNN framework. The increase in the number of parameters could be considered minor.

5.7 Concluding Remarks

In this chapter, we present a novel deep regionlet-based approach for object detection. The proposed region selection network can select *non-rectangular* region within the detection bounding box, by which an object with rigid shape and deformable parts can be better modeled. We also design the deep regionlet learning module so that both the selected regions and the regionlets can be learned simul-

taneously. Moreover, the proposed system can be trained in a fully end-to-end manner without additional efforts. Finally, we extensively evaluate our approach on two detection benchmarks for generic object detection. Experimental results show competitive performance over state-of-the-art.

Methods	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Faster R-CNN [153]	76.4	79.8	80.7	76.2	68.3	55.9	85.1	85.3	89.8	56.7	87.8	69.4	88.3	88.9	80.9	78.4	41.7	78.6	79.8	85.3	72.0
R-FCN [43]	80.5	79.9	87.2	81.5	72.0	69.8	86.8	88.5	89.8	67.0	88.1	74.5	89.8	**90.6**	79.9	81.2	53.7	81.8	81.5	85.9	79.9
SSD [122]	80.6	84.3	87.6	82.6	71.6	59.0	88.2	88.1	89.3	64.4	85.6	76.2	88.5	88.9	87.5	83.0	53.6	83.9	82.2	87.2	81.3
DSSD [64]	81.5	86.6	86.2	82.6	74.9	62.5	89.0	88.7	88.8	65.2	87.0	78.7	88.2	89.0	87.5	83.7	51.1	***86.3***	81.6	85.7	**83.7**
RefineDet [213]	80.0	83.9	85.4	81.4	75.5	60.2	86.4	88.1	89.1	62.7	83.9	77.0	85.4	87.1	86.7	82.6	55.3	82.7	78.5	***88.1***	79.4
DCN [44]	81.4	83.9	85.4	80.1	75.9	68.8	88.4	88.6	89.2	68.0	87.2	75.5	89.5	89.0	86.3	84.8	54.1	85.2	82.6	86.2	80.3
DFPN-R [105]	82.4	**92.0**	***88.2***	81.1	71.2	65.7	88.2	87.9	**92.2**	65.8	86.5	**79.4**	***90.3***	***90.4***	**89.3**	**88.6**	**59.4**	**88.4**	75.3	**89.2**	78.5
C-FRCNN [34]	82.2	84.7	***88.2***	***83.1***	76.2	71.1	87.9	88.7	89.5	68.7	**88.6**	78.2	89.5	88.7	84.8	86.2	55.4	84.7	82.0	86.0	81.7
Ours-A	82.2	83.9	87.6	79.7	76.3	***71.4***	88.9	89.3	89.9	69.1	87.6	78.3	90.1	88.9	87.0	85.8	54.3	84.7	82.2	85.9	82.1
Ours-P	***82.5***	86.1	87.3	80.2	76.6	70.9	***89.1***	***89.5***	89.9	69.8	87.7	***79.0***	**90.4**	89.0	86.4	86.0	54.9	84.8	83.0	87.4	***82.2***
Ours-A§	83.1	***87.6***	88.1	***83.1***	***77.1***	71.2	***89.1***	***89.5***	89.8	**70.9**	87.8	78.4	***90.3***	89.0	88.1	87.9	56.1	84.4	***83.1***	87.8	81.7
Ours-P§	**83.3**	87.0	**88.5**	**83.2**	**77.5**	**71.9**	**89.3**	**89.8**	***90.4***	***70.4***	***88.2***	78.7	**90.4**	89.4	***88.7***	***88.1***	***57.0***	85.4	**83.3**	87.4	81.9

Table 5.6: Complete Object Detection Results on PASCAL VOC 2007 `test` set for each object category. Ours-A(Ours-P)§ denotes we apply the soft-NMS [18] in the test stage.

Methods	FRCN [153]	YOLO9000 [152]	FRCN OHEM	DSSD [64]	SSD* [122]	ION [14]
mAP@0.5(%)	73.8	73.4	76.3	76.3	78.5	76.4
Methods	R-FCN [43]	DP-FCN [138]	Ours-A	Ours-P	Ours-A§	Ours-P§
mAP@0.5(%)	77.6	79.5	80.4	***80.6***	81.2	**81.3**

Table 5.7: Detection results on VOC2012 `test` set using training data "07++12": the union set of 2007 `trainvaltest` and 2012 `trainval`. SSD* denotes the new data augmentation. Ours-A(Ours-P)§ denotes we apply the soft-NMS [18] in the test stage.

Methods	Training Data	mmAP 0.5:0.95	mAP @0.5	mAP small	mAP medium	mAP large
Faster R-CNN [153]	trainval	24.4	45.7	7.9	26.6	37.2
SSD* [122]	trainval	31.2	50.4	10.2	34.5	49.8
DSSD* [64]	trainval	33.2	53.5	13.0	35.4	51.1
R-FCN [43]	trainval	30.8	52.6	11.8	33.9	44.8
DF-RCNN [44]	trainval	33.1	50.3	11.6	34.9	51.2
D-RFCN [44]	trainval	34.5	55.0	14.0	37.7	50.3
CoupleNet [225]	trainval	34.4	54.8	13.4	38.1	50.8
RefineDet512 [213]	trainval	36.4	57.5	16.6	39.9	51.4
RelationNet [89]	trainval	39.0	58.6	-	-	-
Cascade-RCNN [20]	trainval	42.7	62.1	23.7	45.5	55.2
Mask R-CNN [85]	trainval	38.2	59.6	19.8	40.2	48.8
RetinaNet800 [117]	trainval	39.1	59.1	21.8	42.7	50.2
Ours-A	trainval	39.3	59.8	21.7	43.7	50.9
Ours-P	trainval	39.9	61.7	22.9	44.1	51.7

Table 5.8: Object detection results on MS COCO 2017 `test-dev` using ResNet-101 [87] as backbone acchitecture. Training data: union set of 2017 `train` and 2017 `val` set. SSD*, DSSD* denote the new data augmentation

Chapter 6: Conclusions and Direction for Future Work

6.1 Summary

In Chapter 2, we presented a simple thresholded feature learning algorithm under spare support recovery. Motivated by support recovery theoretical guarantees, we proposed a novel approach to learn a dictionary which is optimized for applying the thresholded feature. The competitive performance and superior efficiency of the proposed approach are extensively studied using both synthetic simulations and real-data experiments.

In Chapter 3, we presented two novel sparse representation-based methods for face-based biometrics. (*i.e.* video-based face recognition and template-based face verification). In the first part, we presented a novel structured dictionary learning framework for video-based face recognition. We learned a structured dictionary with both discriminative and reconstructive properties for recognition purposes. Specifically, we encouraged our sub-dictionaries to better represent the corresponding subject face images, while also preserving the subspace structure by enforcing the representation to be low-rank. Moreover, inspired by [21, 119], we proposed an efficient alternating optimization algorithm that converges reasonable faster. Finally, we extensively evaluated our approach on three benchmark databases for video-

based face recognition. The experimental results clearly demonstrate the competitive performance over state-of-the-art.

In the second part, we presented a template regularized sparse coding approach for template-based face verification task. First, we adaptively learned a reference dictionary to adequately represent the training set. Then template adaptive dictionaries were generated by adapting the reference dictionary with the test template pair. Second, we performed template regularized sparse coding on all the dictionaries to derive the discriminative template sparse codes for verification purposes. Finally, both the reference score and template adaptive score were used to measure the similarity of the pair templates. We evaluated our approach on the benchmark IARPA IJB-A dataset for template-based face verification. The experimental results clearly demonstrate competitive performance over existing methods.

In Chapter 4, we addressed the problem of domain shift in computer vision applications. We presented a novel domain adaptive dictionary learning framework for domain adaptation problem. We first learned a common dictionary to recover features shared by all domains. Then we acquired a set of domain-specific dictionaries, which generates a transition path from source to target domain. The common dictionary is essential for reconstruction while domain-specific dictionaries are able to bridge the domain shift. Final feature representations were recovered by utilizing both common and domain-specific dictionaries. We extensively evaluated our approach on both face recognition and object classification and the experimental results clearly confirmed the superior performance of our approach.

Finally, in Chapter 5, we studied another fundamental problem in computer

vision: generic object detection. We presented a deep representation-based, named deep regionlets for object detection. It consists of a region selection network and a deep regionlet learning module. The proposed region selection network can select *non-rectangular* region within the detection bounding box, by which an object with rigid shape and deformable parts can be better modeled. We also designed the deep regionlet learning module so that both the selected regions and the regionlets can be learned simultaneously. Moreover, the proposed system can be trained in a fully end-to-end manner without additional efforts. Finally, we presented the results of ablation studies and extensively evaluated our approach on two detection benchmarks for generic object detection. The proposed algorithm outperforms several state-of-the-art algorithms, such as RetinaNet and Mask R-CNN, even without additional segmentation labels.

6.2 Directions for Future Work

In this section, we outline several promising future directions that could be further explored and some preliminary results where applicable.

6.2.1 Exploring a Deeper Potential

Sparse coding is a technique to learn an efficient and compact representation of data using a small number of basis vectors [112, 132]. It has been sufficiently studied to discover the high-level feature representations from unlabeled samples. Recently, deep learning has gained great attentions in feature learning problems.

One of the advantages of deep learning lies in its composition of multiple non-linear transformations, which result in more descriptive and discriminative feature representations. However, there has been little work [82, 188] on the deep sparse architectures. [188] proposed Deep l_0 Encoders, to model l_0 sparse approximation as feed-forward neural networks. [82] proposed Deep Sparse-coded Network (DSN), a deep architecture for sparse coding as a principled extension of its single-layer counterpart. The results are very preliminary and only reported on the small dataset like CIFAR-10.

The proposed DLTF approach in Chapter 2 is mainly for unsupervised learning task, while recognizing a potential link between DLTF and (supervised) deep models. Eqn. (2.1) can be viewed as a fully-connected layer followed by a non-linearity function, one of the standard building blocks in existing deep models. Here $\max_k$ could be thought either as a generalization of max pooling without reducing dimension, or a locality-aware neuron. Thus, DLTF can also be viewed as a variant of AE, where $\max_k$ acts as the nonlinear function to promote sparsity in hidden layer activations. Its objective is to recover the sparse support besides maintaining a faithful reconstruction. Then a natural question arises: will DLTF be a competitive alternative in layer-wise unsupervised pre-training of deep models?

We conducted preliminary experiments on comparing AE and DLTF in pre-training neural networks for classifying the MNIST dataset. More specifically, we constructed two fully connected network models for the 10-class MNIST classification benchmark (note the different setting with clustering experiments). Model 1 takes 784-dimensional inputs, followed by a 1000-dimensional hidden layer and

Pre-Training	Random (No)	AE	DLTF
Model 1	1.62%	1.08%	1.04%
Model 2	1.26%	0.89%	0.81%

Table 6.1: The error rate comparison for MNIST classification, using different pre-training strategies.

optimized using softmax loss. Model 2 has one additional 1000-dimensional hidden layer appended before the loss function. For Model 1, we performed DLTF with $n = 784, m = 1000$ and choose $k = 100$, to learn the dictionary as the initialization for its hidden layer. For Model 2, we performed DLTF with $n = 784, m = 1,000, k = 200$ for the first hidden layer; we then fix the first layer, and perform DLTF on the first layer outputs with $n = 1,000, m = 1,000, k = 100$, for the second layer. We compared DLTF pre-training with random initializations (no pre-training), and classical AE pre-training. All models are then tuned from end to end, and dropout with a ratio of 0.5 is applied to all fully-connected layers during fine-tuning. Table 6.1 compares the error rates, where DLTF shows an advantage.

Table 6.1 potentially implies that identifying the correct parameter subspace is a more promising goal for layer-wise pre-training, than minimizing the MSE. This hypothesis, if validated further, could lead to new insights on pre-training or even training deep models. We would like to conduct more experiments to verify if DLTF pre-training can benefit the training of more general deep models.

Despite being preliminary, the results suggest that identifying the correct

(sparse) parameter subspace (i.e., the nonzero support) may be a more promising goal for layer-wise pre-training, than solely minimizing the reconstruction error. It is interesting to further explore the potential link between DLTF and (supervised) deep models.

6.2.2 Domain Generalization for Face Recognition

Domain adaptation tackles the problem where data from training set (source domain) and test set (target domain) have different underlying distributions. It adapts the classifier trained using the source domain to perform well on target samples by taking advantage of the unlabeled target data. However, as a related research problem, domain generalization differs from domain adaptation by assuming the target domain samples are not available during the training process. In other words, domain generalization aims to better classify testing data from any unseen target domain.

One of the applications for domain generalization techniques is template adaptation. In this case, the source domain may be a well-learned metric to measure the similarity for the pair of template for face verification. The target domain is the set of template pairs from never before seen subject. [40] investigated the template adaptation by using one-vs-rest linear SVMs. More specifically, it learned two linear SVMs classifiers, where each of them is designed using the positive features from one template in the pair to the large negative features from the training set. Then the final similarity score is calculated by fusing the two SVM margins evaluated on the

other mated template. Although it achieved encouraging performance, it treated each template as one positive feature without considering the intra-class variation. Motivated by [203], one promising direction is exploiting the low-rank structure from the samples within one template.

Another applications for the domain generalization is template-based face verification using the external training set. Given the model trained from the training set and a pair of test template, the assumption that test pair is sampled from the same data distribution as the source domain is usually violated. It is more likely that the test pair is sampled from the unseen target domain. Therefore, developing an adaptive domain generalization method to adapt the model trained from the training set to facilitate the verification performance is necessary and of great importance. One promising directions to address this problem is to use recently proposed meta learning algorithm which involves hierarchical levels of learning in statistics.

6.2.3 Measurement of Domain Shifts

Although the domain adaptation problem has been extensively studied, the sub-problem of characterizing the domain shift is still an open question. Taking pose variations as example, different yaw angles correspond to geometric domain shifts. The illumination variations corresponds to the appearance shifts. One promising direction is to develop statistical models to represent the domain shifts due to pose and illumination variations. Some public face datasets such as CMU MultiPIE [166]

could be used to explore the effectiveness of the statistical model. Another promising direction is to develop principled methods to measure such domain shift and to predict the adaptation ability of the statistical model so that the appropriate adaptation techniques can be employed.

6.2.4 Fast and Accurate Object Detection

Two key issues need to be carefully addressed when designing object detection algorithms,: where the candidate locations are in images and how to discern whether they are the objects of interest. Although studied over several decades, accurate and fast detection is highly challenging when generating bounding boxes for specific object categories, due to cluttered backgrounds, occlusions, variations in object scale, pose, viewpoint and even part deformations. In order to push the boundary of fast and accurate object detection, more fundamental and radically different approaches are needed to achieve a quantum leap in detection performance. Several promising directions are: optimization of deep neural networks, as back-propagation often results in a local optimal, re-designing of the backbone network architectures, which could accelerate the computing time. In addition, pre-training is indispensable to achieve good performance on visual recognition tasks and training without sufficient data could lead to a local optimum. How to efficiently search through the weight space in the deep neural network through alternate optimization techniques will make great difference in computer vision applications including object detection.

Appendix A: Proof of Theorem 1: The Weak Recovery Guarantee

Proof. Denote $\boldsymbol{y} = W^\top \boldsymbol{x} = W^\top W \boldsymbol{z}$. Then for $0 \leq i \leq k$:

$$\begin{aligned} |y_i| &= |z_i + \sum_{j=1, j\neq i}^{k} \langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle z_j| \\ &\geq |z_i| - \mu_W \sum_{j=1, j\neq i}^{k} |z_j| \geq |z_k| - k\mu_W |z_1| \end{aligned} \tag{A.1}$$

On the other hand, for $i > k$:

$$\max_{i>k} \{|y_i|\} = \max_{i>k} \left\{ |\sum_{j=1}^{k} \langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle z_j| \right\} \leq k\mu_W |z_1| \tag{A.2}$$

So if $2k\mu_W |z_1| \leq |z_k|$, the first k entries of $\boldsymbol{y}$ are guaranteed to have greater magnitudes than the rest, and thus will be correctly identified. □

Theorem 1 reveals a sufficient condition on the required number of samples to guarantee the selection consistency, and a similar conclusion could be found in [134]. Following a similar analysis, additional results with regard to the decay rate of nonzeros may be obtained. Moreover, based on the random matrix theory [56] and that $\mu_W \leq 1$ due to the normalization of W, μ_W will decay to 0 with the rate $\frac{\log(n)}{m}$ if W is a sub-Gaussian random matrix.

Appendix B: Proof of Theorem 2: The Strong Recovery Guarantee

Proof. Define $\Omega_z = \text{supp}(\boldsymbol{z}), \Omega_{\bar{z}} = \text{supp}(\bar{\boldsymbol{z}})$, $S = \Omega_z \cup \Omega_{\bar{z}}$. Let $[\boldsymbol{z}]_S$ denotes the subvector indexed by the set S. First we have

$$\begin{aligned} \|\bar{\boldsymbol{z}} - W^\top \boldsymbol{x}\|^2 =& \|\bar{\boldsymbol{z}} - \boldsymbol{z}\|^2 + \|\boldsymbol{z} - W^\top \boldsymbol{x}\|^2 \\ &+ 2\langle \bar{\boldsymbol{z}} - \boldsymbol{z}, \boldsymbol{z} - W^\top \boldsymbol{x}\rangle. \end{aligned} \tag{B.1}$$

Since $\|\bar{\boldsymbol{z}} - W^\top \boldsymbol{x}\|^2 \leq \|\boldsymbol{z} - W^\top \boldsymbol{x}\|^2$ due to the projection property, we have

$$\begin{aligned} \|\bar{\boldsymbol{z}} - \boldsymbol{z}\|^2 &\leq 2\langle \bar{\boldsymbol{z}} - \boldsymbol{z}, W^\top \boldsymbol{x} - \boldsymbol{z}\rangle \\ &= 2\langle \bar{\boldsymbol{z}} - \boldsymbol{z}, [W^\top \boldsymbol{x} - \boldsymbol{z}]_S\rangle \\ &\leq 2\|\bar{\boldsymbol{z}} - \boldsymbol{z}\|\|[W^\top \boldsymbol{x} - \boldsymbol{z}]_S\| \end{aligned} \tag{B.2}$$

It follows

$$\begin{aligned} &\|\bar{\boldsymbol{z}} - \boldsymbol{z}\| \leq 2\|[W^\top \boldsymbol{x} - \boldsymbol{z}]_S\| \\ =&2\|[W^\top \boldsymbol{x} - \boldsymbol{z} + W^\top(W\boldsymbol{z} - \boldsymbol{x}) - W^\top(W\boldsymbol{z} - \boldsymbol{x})]_S\| \\ \leq&2\|[W^\top \boldsymbol{x} - \boldsymbol{z} + W^\top(W\boldsymbol{z} - \boldsymbol{x})]_S\| + 2\|[W^\top(W\boldsymbol{z} - \boldsymbol{x})]_S\| \\ =&2\|\boldsymbol{z} - [W^\top \boldsymbol{x} + W^\top(W\boldsymbol{z} - \boldsymbol{x})]_S\| + 2\|[W^\top(W\boldsymbol{z} - \boldsymbol{x})]_S\| \\ =&2\|\boldsymbol{z} - [W^\top W\boldsymbol{z}]_S\| + 2\|[W^\top \boldsymbol{e}]_S\|. \end{aligned}$$

Using Assumption 1, we have

$$\begin{aligned}
&\|\boldsymbol{z} - [W^\top W\boldsymbol{z}]_S\|^2 \\
=&\|\boldsymbol{z}\|^2 + \|[W^\top W\boldsymbol{z}]_S\|^2 - 2\langle \boldsymbol{z}, [W^\top W\boldsymbol{z}]_S\rangle \\
\leq&\|\boldsymbol{z}\|^2 + ((1+\delta_W) - 2)\langle \boldsymbol{z}, [W^\top W\boldsymbol{z}]_S\rangle \\
\leq&\|\boldsymbol{z}\|^2 - (1-\delta_W)^2\|\boldsymbol{z}\|^2 = (2\delta_W - \delta_W^2)\|\boldsymbol{z}\|^2
\end{aligned}$$

$\delta_W \in (0, 1-\frac{\sqrt{3}}{2})$ is required to ensure $2\delta_W - \delta_W^2 > 0$. Also considering $|S| \leq 2k$, we have

$$\|\bar{\boldsymbol{z}} - \boldsymbol{z}\| \leq 2\sqrt{2\delta_W - \delta_W^2}\|\boldsymbol{z}\| + 2\left\|\max_{2k}(W^\top \boldsymbol{e})\right\| \tag{B.3}$$

Therefore, if the smallest nonzero element z_k of $\boldsymbol{z}$ is no less than the right side of (B.3), $\bar{\boldsymbol{z}}$ and $\boldsymbol{z}$ must have the same support set. It completes the proof. □

The Restricted Isometry Property (RIP) has been a fundamental concept in sparse recovery [22, 204]. Both RIP and mutual coherence require W to behave like an orthonormal system. A recent result [187] reveals both to be special forms of a more generalized sufficient condition for sparse recovery.

Appendix C: Proof of Theorem 3

To use Algorithm 1 to solve the proximal mapping problem (2) for any $\boldsymbol{c} \in \mathbb{R}^m$, we first turn the input $\boldsymbol{c}$ into positive vector $\boldsymbol{c}'$ by taking its absolute values (i.e. $\boldsymbol{c}' = |\boldsymbol{c}|$). The solution to the original problem can be obtained by using the sign of $\boldsymbol{c}$ because we know that the sign of the proximal mapping $prox_{\gamma}^{k',2}(\boldsymbol{c})$ is the same as of $\boldsymbol{c}$. Secondly, we sort the elements in $\boldsymbol{c}'$, which will meet the input requirement of Algorithm 1. Then we can use Algorithm 1 to get the proximal mapping. The sorting procedure costs $O(m \log m)$ time, and the Algorithm 1 costs $O(m)$ time, so the total time complexity is $O(m \log m)$.

When the elements of $\boldsymbol{c}$ are in increasing order, Lemma 5 converts the problem (11) to:

$$\min_{\boldsymbol{q}} \|\boldsymbol{q} - \boldsymbol{c}\|^2 + \gamma \sum_{i=m-k'+1}^{m} q_i^2, \tag{C.1}$$

$$s.t.\ q_1 \leq q_2 \leq ... \leq q_m$$

Dropping some constant terms, the objective can be further written as:

$$\min_{\boldsymbol{q}} \sum_{j=1}^{m-k'} (q_j - c_j)^2 + \sum_{j=m-k'+1}^{m} (1+\gamma)(q_j - \frac{1}{\lambda+1} c_j)^2 \tag{C.2}$$

$$s.t.\ q_1 \leq q_2 \leq ... \leq q_m$$

which can be solved by Algorithm 1.

Appendix D: Proof of Lemma 4

Proof. We prove by contradiction. Suppose that $c_i < c_j$ and $\boldsymbol{q}_i^* > \boldsymbol{q}_j^*$, we will show that we can get a better solution $\mathbf{v}'$ by swapping $\boldsymbol{q}*_i$ and $\boldsymbol{q}_j^*$, which makes a contradiction.

Firstly, we have $q_i^* \leq c_i, q_j^* \leq c_j$ since $\boldsymbol{q}^*$ is optimal. Let the objective value be o_1 before we swap q_i^* and q_j^*, and be o_2 after swap. From the objective definition, the objective value will change:

$$o_1 - o_2 = (c_i - q_i^*)^2 + (c_j - q_j^*)^2 - (c_i - q_j^*)^2 - (c_j - q_i^*)^2 \tag{D.1}$$

Let $a = (c_i - q_i^*), b = (c_i - q_j^*), d = c_i - q_i^* + c_j - q_j^*$, then

$$\begin{aligned} & o_1 - o_2 \\ =& a^2 + (d-a)^2 - b^2 - (d-b)^2 \\ =& 2(a^2 - b^2 - d(a-b)) \\ =& 2(a-b)(a+b-d) \end{aligned}$$

where $a - b = q_j^* - q_i^* < 0$, and $a + b - d = c_i - c_j < 0$, so $o_1 > o_2$, which contradicts the assumption that o_1 is optimal. It completes the proof. □

Appendix E: Proof of Lemma 5 and Lemma 6

Proof. We first prove Lemma 6 by contradiction. Assume $\mathbf{u}_j > \mathbf{u}_{j+1}$ and $\mathbf{x}_j^* \neq \mathbf{x}_{j+1}^*$, then we have $\mathbf{x}_j^* < \mathbf{x}_{j+1}^*$ because we have the constraint (11). Next the contradiction will be shown below by illuminating all six scenarios. We essentially will show there always exists a solution pair $(\mathbf{x}'_j, \mathbf{x}'_{j+1})$ satisfying $\mathbf{x}'_j = \mathbf{x}'_{j+1} \in [\mathbf{x}_j^*, \mathbf{x}_{j+1}^*]$ which gives a better solution than $(\mathbf{x}_j^*, \mathbf{x}_{j+1}^*)$ in every scenario.

- $\mathbf{x}_j^* \leq \mathbf{u}_{j+1} < \mathbf{u}_j \leq \mathbf{x}_{j+1}^*$: We can choose $\mathbf{x}'_j = \mathbf{x}'_{j+1} = \mathbf{u}_j$;
- $\mathbf{u}_{j+1} \leq \mathbf{x}_j^* < \mathbf{x}_{j+1}^* \leq \mathbf{u}_j$: We can choose $\mathbf{x}'_j = \mathbf{x}'_{j+1} = \mathbf{x}_j^*$;
- $\mathbf{x}_j^* < \mathbf{u}_{j+1} \leq \mathbf{x}_{j+1}^* \leq \mathbf{u}_j$: We can choose $\mathbf{x}'_j = \mathbf{x}'_{j+1} = \mathbf{x}_{j+1}^*$;
- $\mathbf{u}_{j+1} \leq \mathbf{x}_j^* < \mathbf{u}_j \leq \mathbf{x}_{j+1}^*$: We can choose $\mathbf{x}'_j = \mathbf{x}'_{j+1} = \mathbf{u}_j$;
- $\mathbf{u}_{j+1} < \mathbf{u}_j \leq \mathbf{x}_j^* < \mathbf{x}_{j+1}^*$: We can choose $\mathbf{x}'_j = \mathbf{x}'_{j+1} = \mathbf{x}_j^*$;
- $\mathbf{x}_j^* < \mathbf{x}_{j+1}^* \leq \mathbf{u}_{j+1} < \mathbf{u}_j$: We can choose $\mathbf{x}'_j = \mathbf{x}'_{j+1} = \mathbf{x}_{j+1}^*$.

This completes the proof of Lemma 6.

Now let us merge two successive variables $\mathbf{x}_j$ and $\mathbf{x}_{j+1}$ if we find $\mathbf{u}_j > \mathbf{u}_{j+1}$. From Lemma 6, we know $\mathbf{x}_j$ should be equal to $\mathbf{x}_{j+1}$. Introduce a new variable $\mathbf{x}_{j \vee j+1}$ to denote the value of $\mathbf{x}_j$ and $\mathbf{x}_{j+1}$. It follows that the original problem in

Lemma 5 is equivalent to solving

$$\min_{\mathbf{x}_1 \le \cdots \le \mathbf{x}_{j-1} \le \mathbf{x}_j = \mathbf{x}_{j+1} \le \mathbf{x}_{j+2} \le \cdots \le \mathbf{x}_J} \quad \sum_{j=1}^{J} \mathbf{t}_j (\mathbf{x}_j - \mathbf{u}_j)^2 \tag{E.1}$$

$$\Leftrightarrow \quad \min_{\mathbf{x}_1 \le \cdots \le \mathbf{x}_{j-1} \le \mathbf{x}_{j \vee j+1} \le \mathbf{x}_{j+2} \le \cdots \le \mathbf{x}_J} \mathbf{t}_j + \mathbf{t}_{j+1} \left(\mathbf{x}_{j \vee j+1} - \frac{\mathbf{t}_j \mathbf{u}_j + \mathbf{t}_{j+1} \mathbf{u}_{j+1}}{\mathbf{t}_j + \mathbf{t}_{j+1}} \right)^2 + \sum_{i \notin \{j, j+1\}} \mathbf{t}_i (\mathbf{x}_i - \mathbf{u}_i)^2$$

$$\Leftrightarrow \quad \min_{\mathbf{x}_1 \le \cdots \le \mathbf{x}_{j-1} \le \mathbf{x}_{j \vee j+1} \le \mathbf{x}_{j+2} \le \cdots \le \mathbf{x}_J} \mathbf{t}_{j \vee j+1} \left(\mathbf{x}_{j \vee j+1} - \mathbf{u}_{j \vee j+1} \right)^2 + \sum_{i \notin \{j, j+1\}} \mathbf{t}_i (\mathbf{x}_i - \mathbf{u}_i)^2 \tag{E.2}$$

where in the last line

$$\mathbf{t}_{j \vee j+1} = \mathbf{t}_j + \mathbf{t}_{j+1} \quad \mathbf{u}_{j \vee j+1} = \frac{\mathbf{t}_j \mathbf{u}_j + \mathbf{t}_{j+1} \mathbf{u}_{j+1}}{\mathbf{t}_j + \mathbf{t}_{j+1}}.$$

Therefore, if we define

$$\mathbf{u}' = [\mathbf{u}_1, \cdots, \mathbf{u}_{j-1}, \mathbf{u}_{j \vee j+1}, \mathbf{u}_{j+2}, \cdots, \mathbf{u}_J]^\top$$

$$\mathbf{t}' = [\mathbf{t}_1, \cdots, \mathbf{t}_{j-1}, \mathbf{t}_{j \vee j+1}, \mathbf{t}_{j+2}, \cdots, \mathbf{t}_J]^\top,$$

then we reduce the problem dimension to $J - 1$. It should be noticed that we do not need to solve the reduced problem from the beginning, since $\mathbf{u}'_{1:j-1} = \mathbf{u}_{1:j-1}$. By this way, Algorithm 1 can be completed in linear time $O(m)$.

Then the solution to the original one (E.1) can be recovered by extending the jth element of Reduce$(\mathbf{u}', \mathbf{t}', j-1)$ to the $(j+1)$th element. We can keep checking if there are any two successive components in $\mathbf{u}'$ disobeying the nondecreasing monotonicity. As long as we find one pair, we can reduce the original problem by

one dimension. Therefore, this problem can be recursively solved by the subroutine "Reduce" in Algorithm 1. This completes the proof of Lemma 5. □

Appendix F: Proof of Proposition 1

.

Proof. Define the Singular Value Decomposition (SVD) of $\mathbf{\Gamma}^k \in \mathbb{R}^{n\times N_t}$ as $\mathbf{\Gamma}^k = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$, where $\mathbf{U}\mathbf{U}^T = \mathbf{I} \in \mathbb{R}^{n\times n}$ and $\mathbf{V}\mathbf{V}^T = \mathbf{I} \in \mathbb{R}^{N_t\times N_t}$, and $\mathbf{\Lambda} = [\tilde{\mathbf{\Lambda}}, \mathbf{0}] \in \mathbb{R}^{n\times N_t}$ is a rectangular diagonal matrix, with $\tilde{\mathbf{\Lambda}} = diag(\lambda_1, ..., \lambda_n)$ being a diagonal matrix. We decompose $\mathbf{V} = [\mathbf{V}_1, \mathbf{V}_2]$ where $\mathbf{V}_1 \in \mathbb{R}^{N_t\times n}$ and evaluate the following term below:

$$
\begin{aligned}
\mathbf{\Gamma}^{kT}(\eta\mathbf{I} + \mathbf{\Gamma}^k\mathbf{\Gamma}^{kT})^{-1}\mathbf{\Gamma}^k =& \mathbf{V}\mathbf{\Lambda}^T\mathbf{U}^T(\eta\mathbf{I} + \mathbf{U}\mathbf{\Lambda}\mathbf{\Lambda}^T\mathbf{U}^T)^{-1}\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \\
=& [\mathbf{V}_1, \mathbf{V}_2]\mathbf{\Lambda}^T\mathbf{U}^T(\eta\mathbf{I} + \mathbf{U}\tilde{\mathbf{\Lambda}}^2\mathbf{U}^T)^{-1}\mathbf{U}\mathbf{\Lambda}[\mathbf{V}_1, \mathbf{V}_2]^T \\
=& [\mathbf{V}_1, \mathbf{V}_2]\mathbf{\Lambda}^T\mathbf{U}^T(\mathbf{U}(\eta\mathbf{I})\mathbf{U}^T + \mathbf{U}\tilde{\mathbf{\Lambda}}^2\mathbf{U}^T)^{-1}\mathbf{U}\mathbf{\Lambda}[\mathbf{V}_1, \mathbf{V}_2]^T \\
=& \mathbf{V}_1\tilde{\mathbf{\Lambda}}(\eta\mathbf{I} + \tilde{\mathbf{\Lambda}}^2)^{-1}\tilde{\mathbf{\Lambda}}\mathbf{V}_1^T \qquad \text{(F.1)} \\
=& \mathbf{V}_1\mathbf{\Theta}\mathbf{V}_1^T
\end{aligned}
$$

where $\mathbf{\Theta} = diag(\overrightarrow{\theta}) = diag(\frac{\lambda_1^2}{\lambda_1^2+\eta}, ..., \frac{\lambda_n^2}{\lambda_n^2+\eta})$.

Next, we replace $\Delta\mathbf{D}^k$ with (4.6) and have

$$\begin{aligned}
&\|\mathbf{J}^k - \Delta\mathbf{D}^k\mathbf{\Gamma}^k\|_F^2 - \|\mathbf{J}^k\|_F^2 \\
=&\|\mathbf{J}^k - \mathbf{J}^k\mathbf{\Gamma}^{k^T}(\eta\mathbf{I} + \mathbf{\Gamma}^k\mathbf{\Gamma}^{k^T})^{-1}\mathbf{\Gamma}^k\|_F^2 - \|\mathbf{J}^k\|_F^2 \\
=&tr(\mathbf{J}^k\mathbf{J}^{k^T}) - tr(2\mathbf{\Gamma}^{k^T}(\eta\mathbf{I} + \mathbf{\Gamma}^k\mathbf{\Gamma}^{k^T})^{-1}\mathbf{\Gamma}^k\mathbf{J}^{k^T}\mathbf{J}^k) \\
+&tr(\mathbf{\Gamma}^{k^T}(\eta\mathbf{I} + \mathbf{\Gamma}^k\mathbf{\Gamma}^{k^T})^{-1}\mathbf{\Gamma}^k\mathbf{J}^{k^T}\mathbf{J}^k\mathbf{\Gamma}^{k^T}(\eta\mathbf{I} + \mathbf{\Gamma}^{k^T}\mathbf{\Gamma}^k)^{-1}\mathbf{\Gamma}^k) - tr(\mathbf{J}^k\mathbf{J}^{k^T}) \\
=&tr(\mathbf{\Gamma}^{k^T}(\eta\mathbf{I} + \mathbf{\Gamma}^k\mathbf{\Gamma}^{k^T})^{-1}\mathbf{\Gamma}^k\mathbf{J}^{k^T}\mathbf{J}^k\mathbf{\Gamma}^{k^T}(\eta\mathbf{I} + \mathbf{\Gamma}^{k^T}\mathbf{\Gamma}^k)^{-1}\mathbf{\Gamma}^k) \\
&- tr(2\mathbf{\Gamma}^{k^T}(\eta\mathbf{I} + \mathbf{\Gamma}^k\mathbf{\Gamma}^{k^T})^{-1}\mathbf{\Gamma}^k\mathbf{J}^{k^T}\mathbf{J}^k)
\end{aligned} \tag{F.2}$$

We plug (F.1) in (F.2) and obtain:

$$\begin{aligned}
&\|\mathbf{J}^k - \Delta\mathbf{D}^k\mathbf{\Gamma}^k\|_F^2 - \|\mathbf{J}^k\|_F^2 \\
=&tr(\mathbf{V}_1\mathbf{\Theta}\mathbf{V}_1^T\mathbf{J}^{k^T}\mathbf{J}^k\mathbf{V}_1\mathbf{\Theta}\mathbf{V}_1^T) - tr(2\mathbf{V}_1\mathbf{\Theta}\mathbf{V}_1^T\mathbf{J}^{k^T}\mathbf{J}^k) \\
=&-tr((2\mathbf{\Theta} - \mathbf{\Theta}^2)\mathbf{V}_1^T\mathbf{J}^{k^T}\mathbf{J}^k\mathbf{V}_1) \\
=&-tr(\mathbf{Q}\mathbf{V}_1^T\mathbf{J}_k^T\mathbf{J}_k\mathbf{V}_1\mathbf{Q}) \\
=&-\|\mathbf{J}_k\mathbf{V}_1\mathbf{Q}\|_F^2 \leq 0
\end{aligned} \tag{F.3}$$

where $\mathbf{Q} = diag(\overrightarrow{q}) = diag(\frac{\sqrt{\lambda_1^4+2\eta\lambda_1^2}}{\lambda_1^2+\eta}, ..., \frac{\sqrt{\lambda_n^4+2\eta\lambda_n^2}}{\lambda_n^2+\eta})$

□

Appendix G: Derivatives of the loss function with respect to the projective transformation parameters

We present complete derivatives of the loss function with respect to the projective transformation parameters Θ. Denote $V(x_p^t, y_p^t, c|\Theta, R)$ as V_p:

$$
\begin{aligned}
\frac{\partial V_p}{\partial \theta_1} &= \frac{\partial V_p}{\partial x_p^s}\frac{\partial x_p^s}{\partial \theta_1} = \frac{x_p^t}{z_p^s}\frac{\partial V_p}{\partial x_p^s} \\
\frac{\partial V_p}{\partial \theta_2} &= \frac{\partial V_p}{\partial x_p^s}\frac{\partial x_p^s}{\partial \theta_2} = \frac{y_p^t}{z_p^s}\frac{\partial V_p}{\partial x_p^s} \\
\frac{\partial V_p}{\partial \theta_3} &= \frac{\partial V_p}{\partial x_p^s}\frac{\partial x_p^s}{\partial \theta_3} = \frac{\partial V_p}{\partial x_p^s} \\
\frac{\partial V_p}{\partial \theta_4} &= \frac{\partial V_p}{\partial y_p^s}\frac{\partial y_p^s}{\partial \theta_4} = \frac{x_p^t}{z_p^s}\frac{\partial V_p}{\partial y_p^s} \\
\frac{\partial V_p}{\partial \theta_5} &= \frac{\partial V_p}{\partial y_p^s}\frac{\partial y_p^s}{\partial \theta_5} = \frac{y_p^t}{z_p^s}\frac{\partial V_p}{\partial y_p^s} \\
\frac{\partial V_p}{\partial \theta_6} &= \frac{\partial V_p}{\partial y_p^s}\frac{\partial y_p^s}{\partial \theta_6} = \frac{\partial V_p}{\partial y_p^s} \\
\frac{\partial V_p}{\partial \theta_7} &= \frac{\partial V_p}{\partial z_p^s}\frac{\partial z_p^s}{\partial \theta_7} = (\frac{\partial V_p}{\partial x_p^s}\frac{\partial x_p^s}{\partial z_p^s} + \frac{\partial V_p}{\partial y_p^s}\frac{\partial y_p^s}{\partial z_p^s})x_p^t \\
&= -x_p^t(\frac{\theta_1 x_p^s + \theta_2 y_p^s + \theta_3}{z_p^{s2}}\frac{\partial V_p}{\partial x_p^s} + \frac{\theta_4 x_p^s + \theta_5 y_p^s + \theta_6}{z_p^{s2}}\frac{\partial V_p}{\partial y_p^s}) \\
&= -\frac{x_p^t}{z_p^s}(\frac{\partial V_p}{\partial x_p^s}x_p^s + \frac{\partial V_p}{\partial y_p^s}y_p^s) \\
\frac{\partial V_p}{\partial \theta_8} &= \frac{\partial V_p}{\partial z_p^s}\frac{\partial z_p^s}{\partial \theta_8} = (\frac{\partial V_p}{\partial x_p^s}\frac{\partial x_p^s}{\partial z_p^s} + \frac{\partial V_p}{\partial y_p^s}\frac{\partial y_p^s}{\partial z_p^s})y_p^t \\
&= -y_p^t(\frac{\theta_1 x_p^s + \theta_2 y_p^s + \theta_3}{z_p^{s2}}\frac{\partial V_p}{\partial x_p^s} + \frac{\theta_4 x_p^s + \theta_5 y_p^s + \theta_6}{z_p^{s2}}\frac{\partial V_p}{\partial y_p^s}) \\
&= -\frac{y_p^t}{z_p^s}(\frac{\partial V_p}{\partial x_p^s}x_p^s + \frac{\partial V_p}{\partial y_p^s}y_p^s)
\end{aligned}
\tag{G.1}
$$

where

$$\frac{\partial V_p}{\partial x_p^s} = \sum_n^H \sum_m^M U_{nm}^c \max(0, 1 - |y_p^s - n|)\eta(x_p^s - m)$$
$$\frac{\partial V_p}{\partial y_p^s} = \sum_n^H \sum_m^M U_{nm}^c \max(0, 1 - |x_p^s - m|)\eta(y_p^s - n)$$
$$\eta(x_p^s - m) = \begin{cases} 0, & \text{if } |m - x_p^s| \geq 1 \\ 1, & \text{if } m > x_p^s \\ -1, & \text{if } m < x_p^s \end{cases} \tag{G.2}$$
$$\eta(y_p^s - n) = \begin{cases} 0, & \text{if } |n - y_p^s| \geq 1 \\ 1, & \text{if } n > y_p^s \\ -1, & \text{if } n < y_p^s \end{cases}$$

Bibliography

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD : An algorithm for designing of overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[2] T. Ahonen, A. Hadid, and M. Pietikäinen. Face recognition with local binary patterns. In *European Conference on Computer Vision (ECCV)*, pages 469–481, 2004.

[3] T. Ahonen, A. Hadid, and M. Pietikäinen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, 2006.

[4] T. Ahonen, E. Rahtu, V. Ojansivu, and J. Heikkilä. Recognition of blurred faces using local phase quantization. In *International Conference on Pattern Recognition (ICPR)*, pages 1–4, 2008.

[5] L. T. Alessandro Bergamo. Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In *Advances in neural information processing systems (NIPS)*, 2010.

[6] O. Arandjelovic, G. Shakhnarovich, J. Fisher, R. Cipolla, and T. Darrell. Face recognition with image sets using manifold density divergence. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 581–588, 2005.

[7] Y. Aytar and A. Zisserman. Tabula rasa: Model transfer for object category detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2252–2259, 2011.

[8] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann. Unsupervised domain adaptation by domain invariant projection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 769–776, 2013.

[9] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann. Domain adaptation on the statistical manifold. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2481–2488, 2014.

[10] N. Bansal, X. Chen, and Z. Wang. Can we gain more from orthogonality regularizations in training deep networks? In *Advances in Neural Information Processing Systems, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 4266–4276, 2018.

[11] R. G. Baraniuk. Compressive sensing [lecture notes]. *IEEE signal processing magazine*, 24(4):118–121, 2007.

[12] Q. Barthelemy, A. Larue, A. Mayoue, D. Mercier, and J. I. Mars. Shift & 2d rotation invariant sparse coding for multivariate signals. *IEEE Trans. Signal Processing*, 60(4):1597–1611, 2012.

[13] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding (CVIU*, 110(3):346–359, June 2008.

[14] S. Bell, C. L. Zitnick, K. Bala, and R. B. Girshick. Inside-Outside Net: Detecting objects in context with skip pooling and recurrent neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2874–2883, 2016.

[15] R. Bhatia. Matrix analysis, graduate texts in mathematics, 1997.

[16] S. Biswas, G. Aggarwal, and R. Chellappa. Robust estimation of albedo for illumination-invariant matching and shape recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):884–899, May 2009.

[17] T. Blumensath and M. E. Davies. Iterative thresholding for sparse approximations. *Journal of Fourier analysis and Applications*, 14(5):629–654, 2008.

[18] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. Soft-NMS - improving object detection with one line of code. In *IEEE International Conference on Computer Vision (ICCV)*, pages 5562–5570, 2017.

[19] X. Cai, C. Wang, B. Xiao, X. Chen, and J. Zhou. Deep nonlinear metric learning with independent subspace analysis for face verification. In *Proceedings of the 20th ACM Multimedia Conference (MM)*, pages 749–752, 2012.

[20] Z. Cai and N. Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[21] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *J. ACM*, 58(3):11, 2011.

[22] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[23] X. Cao, D. P. Wipf, F. Wen, G. Duan, and J. Sun. A practical transfer learning algorithm for face verification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3208–3215, 2013.

[24] H. Cevikalp and B. Triggs. Face recognition based on image sets. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2567–2573, 2010.

[25] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Symposium on Theory of computing*, pages 380–388. ACM, 2002.

[26] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun. Bayesian face revisited: A joint formulation. In *European Conference on Computer Vision (ECCV)*, pages 566–579, 2012.

[27] J. Chen, V. M. Patel, and R. Chellappa. Unconstrained face verification using deep CNN features. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016.

[28] L. Chen. Dual linear regression based classification for face cluster recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2673–2680, 2014.

[29] S. Chen, C. Sanderson, M. T. Harandi, and B. C. Lovell. Improved image set classification via joint sparse approximated nearest subspaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 452–459, 2013.

[30] S. Chen, A. Wiliem, C. Sanderson, and B. C. Lovell. Matching image sets via adaptive multi convex hull. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1074–1081, 2014.

[31] X. Chen, J. Liu, Z. Wang, and W. Yin. Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds. In *Advances in Neural Information Processing Systems (NIPS)*, pages 9079–9089, 2018.

[32] Y. Chen, V. M. Patel, P. J. Phillips, and R. Chellappa. Dictionary-based face recognition from video. In *European Conference on Computer Vision (ECCV)*, pages 766–779, 2012.

[33] Y. Chen, V. M. Patel, S. Shekhar, R. Chellappa, and P. J. Phillips. Video-based face recognition via joint sparse representation. In *International Conference on Automatic Face and Gesture Recognition, FG*, pages 1–8, 2013.

[34] Z. Chen, S. Huang, and D. Tao. Context refinement for object detection. In *European Conference on Computer Vision (ECCV)*, pages 74–89, 2018.

[35] B. Cheng, Y. Wei, H. Shi, R. S. Feris, J. Xiong, and T. S. Huang. Decoupled classification refinement: Hard false positive suppression for object detection. *CoRR*, abs/1810.04002, 2018.

[36] B. Cheng, Y. Wei, H. Shi, R. S. Feris, J. Xiong, and T. S. Huang. Revisiting RCNN: on awakening the classification power of faster RCNN. In *European Conference on Computer Vision (ECCV)*, pages 473–490, 2018.

[37] A. Cherian, V. Morellas, and N. Papanikolopoulos. Robust sparse hashing. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 2417–2420. IEEE, 2012.

[38] A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. *AISTATS*, 1001(48109):2, 2010.

[39] A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 921–928, 2011.

[40] N. Crosswhite, J. Byrne, O. M. Parkhi, C. Stauffer, Q. Cao, and A. Zisserman. Template Adaptation for Face Verification and Identification. *CoRR*, abs/1603.03958, 2016.

[41] Z. Cui, W. Li, D. Xu, S. Shan, and X. Chen. Fusing robust face region descriptors via multiple metric learning for face recognition in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3554–3561, 2013.

[42] Z. Cui, S. Shan, H. Zhang, S. Lao, and X. Chen. Image sets alignment for video-based face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2626–2633, 2012.

[43] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 379–387, 2016.

[44] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 764–773, 2017.

[45] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893, 2005.

[46] H. Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, June 2007.

[47] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 209–216, 2007.

[48] W. Deng, J. Hu, and J. Guo. Extended src: Undersampled face recognition via intraclass variant dictionary. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1864–1870, 2012.

[49] M. Denil and N. de Freitas. Recklessly approximate sparse coding. *arXiv preprint arXiv:1208.0959*, 2012.

[50] P. Dollár, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection. In *European Conference on Computer Vision (ECCV)*, pages 645–659, 2012.

[51] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[52] C. Duan, C. Chiang, and S. Lai. Face verification with local sparse representation. *IEEE Signal Processing Letters*, 20(2):177–180, 2013.

[53] L. Duan, I. W. Tsang, D. Xu, and S. J. Maybank. Domain transfer svm for video concept detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1375–1381, 2009.

[54] L. Duan, D. Xu, I. W. Tsang, and J. Luo. Visual event recognition in videos by learning from web data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1959–1966, 2010.

[55] J. M. Duarte-Carvajalino and G. Sapiro. Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization. *IEEE Trans. Image Processing*, 18(7):1395–1408, 2009.

[56] A. Edelman and N. R. Rao. Random matrix theory. *Acta Numerica*, 14:233–297, 2005.

[57] K. Engan, S. O. Aase, and J. H. Husøy. Multi-frame compression: theory and design. *Signal Processing*, 80(10):2121–2140, 2000.

[58] M. Everingham, L. J. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[59] A. Fawzi, M. Davies, and P. Frossard. Dictionary learning for fast classification based on soft-thresholding. *International Journal of Computer Vision*, 114(2-3):306–321, 2015.

[60] P. F. Felzenszwalb, R. B. Girshick, and D. A. McAllester. Cascade object detection with deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2241–2248, 2010.

[61] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[62] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[63] S. Foucart and H. Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Birkhäuser Basel, 2013.

[64] C. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD : Deconvolutional single shot detector. *CoRR*, abs/1701.06659, 2017.

[65] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware CNN model. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1134–1142, 2015.

[66] S. Gidaris and N. Komodakis. LocNet: Improving localization accuracy for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 789–798, 2016.

[67] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[68] R. B. Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[69] M. Golbabaee and P. Vandergheynst. Average case analysis of sparse recovery with thresholding : New bounds based on average dictionary coherence. In *ICASSP*, pages 3877–3880. IEEE, 2008.

[70] B. Gong, K. Grauman, and F. Sha. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 28, pages 222–230, 2013.

[71] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2066–2073, 2012.

[72] R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *IEEE International Conference on Computer Vision (ICCV)*, pages 999–1006, 2011.

[73] R. Gribonval, H. Rauhut, K. Schnass, and P. Vandergheynst. Atoms of all channels, unite! average case analysis of multi-channel sparse recovery using greedy algorithms. *Journal of Fourier Analysis and Applications*, 14(5):655–687, 2008.

[74] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, Caltech, 2007.

[75] R. Gross, I. Matthews, and S. Baker. Appearance-based face recognition and light-fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4):449–465, April 2004.

[76] R. Gross and J. Shi. The cmu motion of body (mobo) database. Technical Report CMU-RI-TR-01-18, Robotics Institute, June 2001.

[77] P. Grother and M. Ngan. Face recognition vendor test (FRVT): Performance of face identification algorithms. *NIST Interagency Report 8009*, May 2014.

[78] J. Gu, H. Hu, L. Wang, Y. Wei, and J. Dai. Learning region features for object detection. In *European Conference on Computer Vision (ECCV)*, pages 392–406, 2018.

[79] M. Guillaumin, J. J. Verbeek, and C. Schmid. Is that you? metric learning approaches for face identification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 498–505, 2009.

[80] H. Guo, Z. Jiang, and L. S. Davis. Discriminative dictionary learning with pairwise constraints. In *Asian Conference on Computer Vision (ACCV)*, pages 328–342, 2012.

[81] H. Guo, R. Wang, J. Choi, and L. S. Davis. Face verification using sparse representations. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 37–44, 2012.

[82] Y. Gwon, M. Cha, and H. T. Kung. Deep sparse-coded network (DSN). In *International Conference on Pattern Recognition (ICPR)*, 2016.

[83] M. T. Harandi, C. Sanderson, S. A. Shirazi, and B. C. Lovell. Graph embedding discriminant analysis on grassmannian manifolds for improved image set matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2705–2712, 2011.

[84] M. Hayat, M. Bennamoun, and S. An. Learning non-linear reconstruction models for image set classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1915–1922, 2014.

[85] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[86] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision (ECCV)*, pages 346–361, 2014.

[87] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[88] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2957–2964. IEEE, 2012.

[89] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[90] Y. Hu, A. S. Mian, and R. A. Owens. Sparse approximated nearest points for image set classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 121–128, 2011.

[91] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[92] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3296–3297, 2017.

[93] Z. Huang, R. Wang, S. Shan, and X. Chen. Projection metric learning on grassmann manifold with application to video based face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 140–149, 2015.

[94] M. M. Hyder and K. Mahata. An improved smoothed l^0 approximation algorithm for sparse representation. *IEEE Transactions on Signal Processing*, 58(4):2194–2205, April 2010.

[95] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2017–2025, 2015.

[96] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3192–3199, 2013.

[97] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. Acquisition of localization confidence for accurate object detection. In *European Conference on Computer Vision (ECCV)*, pages 816–832, 2018.

[98] Z. Jiang, Z. Lin, and L. S. Davis. Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1697–1704, 2011.

[99] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba. Undoing the damage of dataset bias. In *European Conference on Computer Vision (ECCV)*, pages 158–171, 2012.

[100] M. Kim, S. Kumar, V. Pavlovic, and H. A. Rowley. Face tracking and recognition with visual constraints in real-world videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[101] S. Kim, H. Kook, J. Sun, M. Kang, and S. Ko. Parallel feature pyramid network for object detection. In *European Conference on Computer Vision (ECCV)*, 2018.

[102] T. Kim, J. Kittler, and R. Cipolla. Learning discriminative canonical correlations for object recognition with image sets. In *European Conference on Computer Vision (ECCV)*, pages 251–262, 2006.

[103] B. F. Klare, B. Klein, E. Taborsky, A. Blanton, J. Cheney, K. Allen, P. Grother, A. Mah, and A. K. Jain. Pushing the frontiers of unconstrained face detection and recognition: IARPA Janus Benchmark A. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1931–1939, 2015.

[104] S. Kong and D. Wang. A dictionary learning approach for classification: Separating the particularity and the commonality. In *European Conference on Computer Vision (ECCV)*, pages 186–199, 2012.

[105] T. Kong, F. Sun, W. Huang, and H. Liu. Deep feature pyramid reconfiguration for object detection. In *European Conference on Computer Vision (ECCV)*, pages 172–188, 2018.

[106] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[107] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105. 2012.

[108] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1785–1792, 2011.

[109] H. Law and J. Deng. CornerNet: Detecting objects as paired keypoints. In *European Conference on Computer Vision (ECCV)*, pages 765–781, 2018.

[110] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2169–2178, 2006.

[111] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[112] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems (NIPS)*, pages 801–808. 2007.

[113] K. Lee, J. Ho, M. Yang, and D. J. Kriegman. Video-based face recognition using probabilistic appearance manifolds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 313–320, 2003.

[114] L. Li, R. Socher, and F. Li. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2036–2043, 2009.

[115] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun. DetNet: Design backbone for object detection. In *European Conference on Computer Vision (ECCV)*, pages 339–354, 2018.

[116] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017.

[117] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[118] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[119] Z. Lin, R. Liu, and Z. Su. Linearized alternating direction method with adaptive penalty for low-rank representation. In *Advances in neural information processing systems (NIPS)*, pages 612–620, 2011.

[120] Z. Lin, C. Lu, and H. Li. Optimized projections for compressed sensing via direct mutual coherence minimization. *arXiv preprint arXiv:1508.03117*, 2015.

[121] S. Liu, D. Huang, and Y. Wang. Receptive field block net for accurate and fast object detection. In *European Conference on Computer Vision (ECCV)*, pages 404–419, 2018.

[122] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.

[123] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[124] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu. Transfer feature learning with joint distribution adaptation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2200–2207, 2013.

[125] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu. Transfer joint matching for unsupervised domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1410–1417, 2014.

[126] D. G. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1150–1157, 1999.

[127] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[128] J. Lu, G. Wang, W. Deng, and P. Moulin. Simultaneous feature and dictionary learning for image set based face recognition. In *European Conference on Computer Vision (ECCV)*, pages 265–280, 2014.

[129] J. Lu, G. Wang, W. Deng, P. Moulin, and J. Zhou. Multi-manifold deep metric learning for image set classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1137–1145, 2015.

[130] J. Lu, G. Wang, and P. Moulin. Image set classification using holistic multiple order statistics features and localized multi-kernel metric learning. In *IEEE International Conference on Computer Vision (ICCV)*, pages 329–336, 2013.

[131] L. Ma, C. Wang, B. Xiao, and W. Zhou. Sparse representation for face recognition based on discriminative low-rank dictionary learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2586–2593, 2012.

[132] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, January 2010.

[133] J. Mairal, F. R. Bach, and J. Ponce. Task-driven dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):791–804, 2012.

[134] A. Makhzani and B. Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

[135] S. Mallat. *A Wavelet Tour of Signal Processing, 2nd Edition*. Academic Press, 1999.

[136] M. Mathilde, P. Dominique, and T. Karine. Learning out of leaders. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3):475513.

[137] A. Mignon and F. Jurie. PCCA: A new approach for distance learning from sparse pairwise constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2666–2672, 2012.

[138] T. Mordan, N. Thome, M. Cord, and G. Henaff. Deformable part-based fully convolutional network for object detection. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[139] Y. Naderahmadian, S. Beheshti, and M. A. Tinati. Correlation based online dictionary learning algorithm. *IEEE Trans. Signal Processing*, 64(3):592–602, 2016.

[140] H. V. Nguyen and L. Bai. Cosine similarity metric learning for face verification. In *Asian Conference on Computer Vision (ACCV)*, volume 6493, pages 709–720, 2010.

[141] J. Ni, Q. Qiu, and R. Chellappa. Subspace interpolation via dictionary learning for unsupervised domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 692–699, 2013.

[142] W. Ouyang, X. Zeng, X. Wang, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, H. Li, K. Wang, J. Yan, C. C. Loy, and X. Tang. DeepID-Net: Object detection with deformable part based convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1320–1334, 2017.

[143] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1187–1192, 2009.

[144] H. Park and C. Jun. A simple and fast algorithm for k-medoids clustering. *Expert Syst. Appl.*, 36(2):3336–3341, 2009.

[145] V. M. Patel, R. Gopalan, and R. Chellappa. Visual domain adaptation: An overview of recent advances. *IEEE Signal Processing Magazine*, July 2014.

[146] R. Poppe. A survey on vision-based human action recognition. *Image Vision Comput.*, 28(6):976–990, 2010.

[147] Q. Qiu, V. Patel, P. Turage, and R. Chellappa. Domain adaptive dictionary learning. In *European Conference on Computer Vision (ECCV)*, pages 631–645, 2012.

[148] I. Ramírez and G. Sapiro. An MDL framework for sparse coding and dictionary learning. *IEEE Trans. Signal Processing*, 60(6):2913–2927, 2012.

[149] R. Ranjan, A. Bansal, H. Xu, S. Sankaranarayanan, J. Chen, C. D. Castillo, and R. Chellappa. Crystal loss and quality pooling for unconstrained face verification and recognition. *CoRR*, abs/1804.01159, 2018.

[150] R. Ranjan, A. Bansal, J. Zheng, H. Xu, J. Gleason, B. Lu, A. Nanduri, J. Chen, C. D. Castillo, and R. Chellappa. A fast and accurate system for face detection, identification, and verification. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 2019.

[151] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[152] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[153] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[154] R. Rubinstein, A. M. Bruckstein, and M. Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057, 2010.

[155] R. Rubinstein and M. Elad. Dictionary learning for analysis-synthesis thresholding. *IEEE Trans. Signal Processing*, 62(22):5962–5972, 2014.

[156] R. Rubinstein, M. Zibulevsky, and M. Elad. Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. *Cs Technion*, 40(8):1–15, 2008.

[157] R. Rubinstein, M. Zibulevsky, and M. Elad. Double sparsity: learning sparse dictionaries for sparse signal approximation. *IEEE Trans. Signal Processing*, 58(3):1553–1564, 2010.

[158] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *European Conference on Computer Vision (ECCV)*, pages 213–226, 2010.

[159] S. Sankaranarayanan, A. Alavi, and R. Chellappa. Triplet similarity embedding for face verification. *CoRR*, abs/1602.03418, 2016.

[160] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[161] G. Shakhnarovich, J. W. F. III, and T. Darrell. Face recognition from long-term observations. In *European Conference on Computer Vision (ECCV)*, pages 851–868, 2002.

[162] S. Shekhar, V. M. Patel, H. V. Nguyen, and R. Chellappa. Generalized domain-adaptive dictionaries. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 361–368, 2013.

[163] Y. Shi and F. Sha. Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012.

[164] A. Shrivastava, A. Gupta, and R. B. Girshick. Training region-based object detectors with online hard example mining. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 761–769, 2016.

[165] A. Shrivastava, S. Shekhar, and V. M. Patel. Unsupervised domain adaptation using parallel transport on grassmann manifold. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 277–284, 2014.

[166] T. Sim, S. Baker, and M. Bsat. The cmu pose, illumination, and expression database. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1615–1618, 2003.

[167] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman. Fisher vector faces in the wild. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2013.

[168] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[169] B. Singh and L. S. Davis. An analysis of scale invariance in object detection SNIP. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3578–3587, 2018.

[170] B. Singh, M. Najibi, and L. S. Davis. SNIPER: efficient multi-scale training. *CoRR*, abs/1805.09300, 2018.

[171] P. Sprechmann, R. Litman, T. B. Yakar, A. M. Bronstein, and G. Sapiro. Supervised sparse analysis and synthesis operators. In *Advances in Neural Information Processing Systems (NIPS)*, pages 908–916, 2013.

[172] K. Sun, Z. Wang, D. Liu, and R. Liu. L_p-norm constrained coding with frank-wolfe network. *CoRR*, abs/1802.10252, 2018.

[173] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1701–1708, 2014.

[174] K. Tono, A. Takeda, and J.-y. Gotoh. Efficient DC algorithm for constrained sparse optimization. *arXiv preprint arXiv:1701.08498*, 2017.

[175] P. K. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea. Machine recognition of human activities: A survey. *IEEE Trans. Circuits Syst. Video Techn.*, 18(11):1473–1488, 2008.

[176] D. Vázquez, A. M. López, J. Marín, D. Ponsa, and D. G. Gomez. Virtual and real world adaptation for pedestrian detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):797–809, 2014.

[177] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 511–518, 2001.

[178] P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[179] H. Wang, Q. Wang, M. Gao, P. Li, and W. Zuo. Multi-scale location-aware kernel representation for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[180] R. Wang and X. Chen. Manifold discriminant analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 429–436, 2009.

[181] R. Wang, H. Guo, L. S. Davis, and Q. Dai. Covariance discriminative learning: A natural and efficient approach to image set classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2496–2503, 2012.

[182] R. Wang, S. Shan, X. Chen, and W. Gao. Manifold-manifold distance with application to face recognition based on image set. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[183] W. Wang, R. Wang, Z. Huang, S. Shan, and X. Chen. Discriminant analysis on Riemannian manifold of Gaussian distributions for face recognition with image sets. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2048–2057, 2015.

[184] X. Wang, T. X. Han, and S. Yan. An HOG-LBP human detector with partial occlusion handling. In *IEEE International Conference on Computer Vision (ICCV)*, 29 2009.

[185] X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for generic object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 17–24, 2013.

[186] X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for generic object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(10):2071–2084, 2015.

[187] Y. Wang, J. Zeng, Z. Peng, X. Chang, and Z. Xu. Linear convergence of adaptively iterative thresholding algorithms for compressed sensing. *IEEE Transactions on Signal Processing*, 63(11):2957–2971, 2015.

[188] Z. Wang, Q. Ling, and T. S. Huang. Learning deep l_0 encoders. In *Proceedings of the Thirtieth Conference on Artificial Intelligence (AAAI)*, pages 2194–2200, 2016.

[189] Z. Wang, J. Yang, H. Zhang, Z. Wang, Y. Yang, D. Liu, and T. S. Huang. *Sparse Coding and its Applications in Computer Vision*. World Scientific, 2016.

[190] Y. Wei, X. Pan, H. Qin, W. Ouyang, and J. Yan. Quantization mimic: Towards very tiny CNN for object detection. In *European Conference on Computer Vision (ECCV)*, pages 274–290, 2018.

[191] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1753–1760, 2009.

[192] Z. Wen and W. Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013.

[193] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 529–534, 2011.

[194] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, pages 478–487, 2016.

[195] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2017.

[196] H. Xu, X. Lv, X. Wang, Z. Ren, N. Bodla, and R. Chellappa. Deep regionlets for object detection. In *European Conference on Computer Vision (ECCV)*, pages 827–844, 2018.

[197] H. Xu, X. Lv, X. Wang, Z. Ren, and R. Chellappa. Deep regionlets: Blended representation and deep learning for generic object detection. *CoRR*, abs/1811.11318, 2018.

[198] H. Xu, Z. Wang, H. Yang, D. Liu, and J. Liu. Learning simple thresholded features with sparse support recovery. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[199] H. Xu, J. Zheng, A. Alavi, and R. Chellappa. Learning a structured dictionary for video-based face recognition. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016.

[200] H. Xu, J. Zheng, A. Alavi, and R. Chellappa. Template regularized sparse coding for face verification. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, 2016*, pages 1448–1454, 2016.

[201] H. Xu, J. Zheng, A. Alavi, and R. Chellappa. Cross-domain visual recognition via domain adaptive dictionary learning. *arXiv preprint arXiv:1804.04687*, 2018.

[202] H. Xu, J. Zheng, and R. Chellappa. Bridging the domain shift by domain adaptive dictionary learning. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 96.1–96.12, 2015.

[203] Z. Xu, W. Li, L. Niu, and D. Xu. Exploiting low-rank structure from latent domains for domain generalization. In *European Conference on Computer Vision (ECCV)*, pages 628–643, 2014.

[204] H. Yang, Y. Huang, L. Tran, J. Liu, and S. Huang. On benefits of selection diversity via bilevel exclusive sparsity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5945–5954, 2016.

[205] J. Yang, R. Yan, and A. G. Hauptmann. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM Multimedia Conference (MM)*, pages 188–197, 2007.

[206] M. Yang, W. Liu, and L. Shen. Joint regularized nearest points for image set based face recognition. In *International Conference on Automatic Face and Gesture Recognition, FG*, pages 1–7, 2015.

[207] M. Yang, L. Zhang, X. Feng, and D. Zhang. Fisher discrimination dictionary learning for sparse representation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 543–550, 2011.

[208] M. Yang, P. Zhu, L. J. V. Gool, and L. Zhang. Face recognition based on regularized nearest points between image sets. In *International Conference on Automatic Face and Gesture Recognition, FG*, pages 1–7, 2013.

[209] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *CoRR*, abs/1411.7923, 2014.

[210] X. Yuan, P. Li, and T. Zhang. Exact recovery of hard thresholding pursuit. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3558–3566, 2016.

[211] G. Zhang, R. He, and L. S. Davis. Jointly learning dictionaries and subspace structure for video-based face recognition. In *Asian Conference on Computer Vision (ACCV)*, pages 97–111, 2014.

[212] Q. Zhang and B. Li. Discriminative K-SVD for dictionary learning in face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2691–2698, 2010.

[213] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li. Single-shot refinement neural network for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[214] T. Zhang. Sparse recovery with orthogonal matching pursuit under rip. *IEEE Transactions on Information Theory*, 57(9):6215–6221, 2011.

[215] Y. Zhang, Z. Jiang, and L. S. Davis. Learning structured low-rank representations for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 676–683, 2013.

[216] Z. Zhang, F. Li, T. W. S. Chow, L. Zhang, and S. Yan. Sparse codes auto-extractor for classification: A joint embedding and dictionary learning framework for representation. *IEEE Trans. Signal Processing*, 64(14):3790–3805, 2016.

[217] Z. Zhang, S. Qiao, C. Xie, W. Shen, B. Wang, and A. L. Yuille. Single-shot object detection with enriched semantics. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[218] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Comput. Surv.*, 35(4):399–458, 2003.

[219] X. Zhao, S. Liang, and Y. Wei. Pseudo mask augmented object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[220] J. Zheng and Z. Jiang. Learning view-invariant sparse representations for cross-view action recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3176–3183, 2013.

[221] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai. Graph regularized sparse coding for image representation. *Image Processing, IEEE Transactions on*, 20(5):1327–1336, 2011.

[222] Y. Zhong, J. Chen, and B. Huang. Toward end-to-end face recognition through alignment learning. *IEEE Signal Process. Lett.*, 24(8):1213–1217, 2017.

[223] P. Zhou, B. Ni, C. Geng, J. Hu, and Y. Xu. Scale-transferrable object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[224] X. Zhu, Z. Huang, H. Cheng, J. Cui, and H. T. Shen. Sparse hashing for fast multimedia search. *ACM Transactions on Information Systems (TOIS)*, 31(2):9, 2013.

[225] Y. Zhu, C. Zhao, J. Wang, X. Zhao, Y. Wu, and H. Lu. CoupleNet: Coupling global structure with local parts for object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4146–4154, 2017.

[226] W. Y. Zou, X. Wang, M. Sun, and Y. Lin. Generic object detection with dense neural patterns and regionlets. In *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2014.