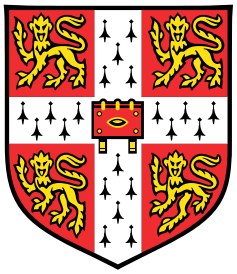
Application of Prior Information to Discriminative Feature Learning



Yang Liu

Computer Laboratory
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Lucy Cavendish College

November 2018

Application of Prior Information to Discriminative Feature Learning

Yang Liu

Learning discriminative feature representations has attracted a great deal of attention since it is a critical step to facilitate the subsequent classification, retrieval and recommendation tasks. In this dissertation, besides incorporating prior knowledge about image labels into the image classification as most prevalent feature learning methods currently do, we also explore some other general-purpose priors and verify their effectiveness in the discriminant feature learning. As a more powerful representation can be learned by implementing such general priors, our approaches achieve state-of-the-art results on challenging benchmarks. We elaborate on these general-purpose priors and highlight where we have made novel contributions.

We apply sparsity and hierarchical priors to the explanatory factors that describe the data, in order to better discover the data structure. More specifically, in the first approach we propose that we only incorporate sparse priors into the feature learning. To this end, we present a support discrimination dictionary learning method, which finds a dictionary under which the feature representation of images from the same class have a common sparse structure while the size of the overlapped signal support of different classes is minimised. Then we incorporate sparse priors and hierarchical priors into a unified framework, that is capable of controlling the sparsity of the neuron activation in deep neural networks. Our proposed approach automatically selects the most useful low-level features and effectively combines them into more powerful and discriminative features for our specific image classification problem.

We also explore priors on the relationships between multiple factors. When multiple independent factors exist in the image generation process and only some of them are of interest to us, we propose a novel multi-task adversarial network to learn a disentangled feature which is optimized with respect to the factor of interest to us, while being distraction factors agnostic. When common factors exist in multiple tasks, leveraging common factors cannot only make the learned feature representation more robust, but also enable the model to generalise from very few labelled samples. More specifically, we address the domain adaptation problem and propose the re-weighted adversarial adaptation network to reduce the feature distribution divergence and adapt the classifier from source to target domains.

Declaration

This dissertation is submitted for the degree of Doctor of Philosophy. As required by the University Statues, I hereby declare that this dissertation is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text and Acknowledgements.

This dissertation contains less than the limits of 150 figures and 60,000 words.

Yang Liu
November 2018

Acknowledgements

First of all, I would like to express my deepest thanks to my supervisor, Dr Ian Wassell, for his kind support and patience all the way during my PhD study. He has offered me the freedom to pursue my own research interest and he has always been helpful to give me valuable advice when difficulties occurred. His professional spirits, patience, and expertise have played important roles in my life.

My heartfelt thanks are also to my coauthor Qingchao Chen at University College London for all the insightful discussions and detailed suggestions. He has always been kind and supportive whenever I asked for help. Thanks for his time and efforts helping me with my coding and mathematical difficulties.

I would also thank Zhaowen Wang and Hailin Jin at Adobe Research, for many insightful discussion and professional collaborations that helped me improve the value of my research from both academic and industrial perspectives. I am very grateful to Adobe Research for giving me the freedom to carry on my research during my visit.

I appreciate the kindness from the members of the Digital Technology Group and the support from the staff of the Computer Laboratory. My particular thanks to Mrs Lise Gough for offering the kindest and warmest advises whenever needed.

I would also like to thank the Cambridge Trust scholarship, Lucy Cavendish College, the Computer Laboratory and Cambridge Philosophical Society for their generous financial support.

The last paragraph is reserved for my parents, for their cares, understanding, great support and so much more besides.

Yang Liu

November 2018

Table of contents

List of figures

List of tables

List of Acronyms

AAE	Adversarial Auto-encoder
ADMM	Alternating direction method of multipliers
AMP	Approximately message passing
AWGN	Additive white Gaussian noise
BP	Basis pursuit
BPDN	Basis pursuit de-noise
CNN	Convolutional neural network
DA	Domain Adaptation
DFT	Discrete Fourier Transform
DL	Dictionary learning
DLL	Dictionary learning layer
DNN	Deep Neural Networks
ELU	Exponential linear unit
EMD	Earth Mover's Distance
FCL	Fully connected layer
FISTA	Fast Iterative Shrinkage-Thresholding Algorithm
FOCUSS	Focal underdetermined system solver
GAN	Generative adversarial networks
GPU	Graphics processing unit
IRLS	Iteratively reweighted least squares minimization
JS	Jenson-Shannon
LASSO	Least absolute shrinkage and selection operator
MMD	Maximum Mean Discrepancy
MMV	Multiple measurement vector
MOD	Method of optimal direction
MSE	Mean-Squared Error
OT	Optimal transport
ReLu	Rectified Linear Unit

ResNet	Residual net
RIP	Restricted isometry property
SGD	Stochastic gradient descent
SRC	Sparse representation based classification
UDA	Unsupervised domain adaptation
WGAN	Wasserstein GAN

List of Symbols

x	Scalar
$\boldsymbol{x}$	Vector
$\boldsymbol{X}$	Matrix
$f(\cdot), F(\cdot)$	Function
$\delta(\cdot)$	The Dirac delta function
$\mathbb{E}(\cdot)$	Expectation
$\mathrm{P}(\cdot)$	Probability
$\|\boldsymbol{x}\|_p$	ℓ_p norm
$\|\boldsymbol{X}\|_{p,q}$	$\ell_{p,q}$ norm
$\|\boldsymbol{X}\|_F$	Frobenius norm
$(\boldsymbol{X})^T$	Matrix transpose
$(\boldsymbol{X})^{-1}$	Matrix inverse
$\mathbf{1}$	Array with all ones
$\boldsymbol{I}$	Identity matrix
x_i	The i^{th} element of the vector $\boldsymbol{x}$
$x_{(i,j)}$	The element at the i^{th} row , the j^{th} column of $\boldsymbol{X}$
$\boldsymbol{x}_i$	The i^{th} column of the matrix $\boldsymbol{X}$
$\boldsymbol{X}_i$	The sub-matrix of the matrix $\boldsymbol{X}$ for the i^{th} class
$x^{(k)}$, $\boldsymbol{x}^{(k)}$, $\boldsymbol{X}^{(k)}$	Scalar, vector and matrix at the k^{th} iteration
$\mathrm{diag}(\boldsymbol{x})$	Diagonal matrix with diagonal elements given by vector $\boldsymbol{x}$
$\boldsymbol{x}^{\odot 2}$	Element by element square of vector $\boldsymbol{x}$
$\max(\cdot)$	Operator of taking the maximum value
$\min(\cdot)$	Operator of taking the minimum value
$\circ$	Operator of Hadamard element-wise product
$*$	Operator of Convolution
$\frac{\partial L}{\partial \theta}$	Gradient of the loss L with respect to parameters θ
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate normal distribution with mean vector

Chapter 1

Introduction

1.1 Feature Learning

Computer vision has been successfully used in real-world recognition problems, where state-of-the-art recognition algorithms focus on training the classifier or the regressor from large training sets. Learning discriminative feature representations has attracted a great deal of attention since it is a critical step to facilitate the subsequent classification, retrieval and recommendation tasks. A discriminative feature representation of the data can make it easier to extract useful information when building classifiers or other predictors [9].

However, classical computer vision approaches depended on hand-crafted features that often rely on expert knowledge and are time consuming to design, thus limiting the scalability and effectiveness of the approach. This motivates the design of efficient feature learning techniques to automatically learn the intrinsic structure of data and to discover valuable information from raw data. As real-world data such as images, video, and sensor measurement are usually complex, redundant and highly variable, how to discover useful feature representation in a scalable and efficient manner from raw data is still a challenging open question.

In the world of big data, where priors are not usually available, the progress of feature learning is limited by the growth of computational power and the growth of data. Incorporating prior knowledge about the representation has long been known to influence profoundly the effectiveness of feature learning [9]. In the image classification task, the primary prior information comes from the labels, which inform the model that a set of images contains similar patterns. Most of the current prevalent feature learning methods rely upon a significant amount of labelled data (label priors), which is expensive and laborious to collect, and may be infeasible to produce in some cases. Hence techniques to automatically learn feature representation from limited labelled data are desirable. In this dissertation, besides

label priors, we explore different types of general-purpose prior knowledge and employ data-driven models for learning a discriminative and compact image representation, that is ideal for robust image classification. It is worth noting that these general-purpose priors require some knowledge of the domain to permit design but, unlike hand-designed features, are higher level and have relatively broad applicability.

1.2 Contribution

The fundamental aim of this dissertation is to explore feature representations that have a better discriminative and generalisation capability for robust image classification, and can be applied to a broad range of scenarios. Besides incorporating label priors into the image classification as most prevalent feature learning methods currently do, we also explore some other general-purpose priors and verify their effectiveness in the image classification task. More specifically, we apply different general priors, particularly to *explanatory factors* that describe the data, in order to better discover the data structure. As a more powerful representation can be learned by implementing such general priors, our approaches achieve state-of-the-art results on challenging benchmarks. We elaborate on these general-purpose priors and highlight where we have made novel contributions.

1. **Sparsity:** For any given image, only a small fraction of the possible factors are relevant to describe it. In terms of feature representation, this could be achieved by incorporating certain forms of sparse priors on the representation. In Chapter 3, we first derive a new discriminative metric (which defines the similarity between objects) that uses the sparse structure of the feature representation rather than the magnitude of the feature representation directly in order to enhance the discrimination capability and consequently providing robust image classification.

2. **A hierarchical organisation of explanatory factors:** These factors (or concepts) are useful to describe the data in a hierarchical fashion, where the more abstract concepts that are higher in the hierarchy are defined in terms of less abstract ones. In Chapter 4, we provide a unified framework that can potentially take advantage of sparse and hierarchical priors. More specifically, by incorporating dictionary learning (sparse prior) and deep learning (hierarchical prior), we propose an approach to control the sparsity of neuron activation in the deep neural networks, that automatically select the most reuseful low level features and effectively combine them into more powerful and discriminative features for our specific image classification problem.

3. **Multiple independent explanatory factors:** Multiple independent factors exist in the image generation process, however, in most applications only some factors are of interest to us. A good feature representation should disentangle the underlying factors so that what one learns about one factor could generalise to many configurations of the other factors. In chapter 5, instead of learning a shared representation to predict all the factors as is performed in conventional multi-task learning, we propose a novel multi-task adversarial network to learn a disentangled feature which is optimised with respect to the factor of interest to us, while being distraction factors agnostic.

4. **Common factors across tasks:** When common factors exist in multiple tasks, leveraging common factors cannot only make the learned feature representation more robust, but also enable the model to generalise from very few labelled samples. By incorporating the prior knowledge about common factors across classification tasks in two domains, in chapter 6 we propose a new unsupervised domain adaptation model, i.e., the re-weighted adversarial adaptation network, that reduces the feature distribution divergence and adapts the classifier from source (labelled) to target (unlabelled) domains. Consequently, the image classification performance can be enhanced when the feature learned from training data can perform well on test data that have never been seen in training or have even been sampled from a different domain, i.e., has a better generalisation capability.

1.3 Organization

This dissertation is organised as follows:

Chapter 2 gives the necessary background on dictionary learning and neural networks, particularly concerning sparse coding and deep convolutional neural networks. We also provide a discussion on contemporary deep learning architectures with a particular emphasis on image classification, in order to better understand our work and motivation.

Chapter 3 is the first chapter where we begin to present our research contributions. By incorporating the sparse prior on feature representation, we propose a new discriminative metric that uses the sparse structure, i.e., support, to measure the similarity between the pairs of feature representations, rather than using the Euclidean distance directly, which is an approach that is widely adopted in the prior art. Here the support of the representation denotes the indices of the non-zero elements of the sparse image representation under some dictionary. More specifically, we incorporate a structured sparse prior in feature learning and propose a support discrimination dictionary learning method, which finds a dictionary under which the feature representation of images from the same class have a common sparse

structure while the size of the overlapped signal support of different classes is minimised. The superior performance of the proposed approach in comparison to the state-of-art is demonstrated using face, object and scene datasets.

Chapter 4 presents the work conducted concerning the use of both sparse and hierarchical priors on feature representation to achieve more powerful and discriminative features. In our approach, we propose a unified framework, that combines the strength of both a conventional convolutional neural network (hierarchical prior) and dictionary learning (sparse prior), to automatically select the most useful low level features and effectively combine them into a more abstract, discriminant and robust feature representation to better describe categorical concepts in image understanding. Moreover, we take advantage of the structure of the dictionary and design a new label discriminative regressor to further improve the discriminative capability and avoid overfitting. Our proposed approach is evaluated using various challenging scene datasets and shows superior performance to many state-of-the-art approaches.

Chapter 5 address the problem of image feature learning where multiple independent factors exist in the image generation process and only some of the factors are of interest to us. We present a novel multi-task adversarial network based on an encoder-discriminator-generator architecture. The encoder extracts a disentangled feature representation for the factors of interest. The discriminators classify each of the factors as individual tasks. The encoder and the discriminators are trained cooperatively on the factors of interest, but in an adversarial way on the factors of distraction. The generator provides further regularisation on the learned feature by reconstructing images with shared factors as the input image. We design a new optimisation scheme to stabilise the adversarial optimisation process when multiple distributions need to be aligned. The experiments conducted on face recognition and font recognition tasks show that our method outperforms the state-of-the-art methods in terms of both recognising the factors of interest and generalisation to images with unseen variations.

Chapter 6 address the domain adaptation problem by using the prior knowledge about the common explanatory factors that exist between classification tasks in two tasks/domains. We propose a new re-weighted adversarial adaptation network that can jointly learn adaptive classifiers and transferable features from labelled data in the source domain and unlabeled data in the target domain. More specifically, to alleviate the requirement for many common supports in matching the feature distribution between the source and target domains, we choose to minimise the optimal transport based Earth-Mover distance and reformulate it to a minimax objective function. The overall network can be trained in an end-to-end and adversarial manner. To further adapt the classifier, we propose to match the label distribution

and embed it into the adversarial training. Empirical evidence shows that the new approach outperforms state of the art methods on standard domain adaptation benchmarks, particularly when the domain shifts are disparate.

Chapter 7 summarises the contributions made in this dissertation, reviews the results, and looks at the research questions that arise from work presented in this dissertation. Proposals are made for future possible research directions.

1.4 Publication List

These contributions have led to the following publications:

1. **Yang Liu**, Ian J. Wassell. A New Face Recognition Algorithm based on Dictionary Learning for a Single Training Sample per Person. British Machine Vision Conference (BMVC 2015), Sep 2015.

2. **Yang Liu**, Wei Chen, Qingchao Chen, Ian J. Wassell, Support Discrimination Dictionary Learning for Image Classification. European Conference on Computer Vision (ECCV 2016), Oct 2016. [Chapter 3]

3. Wei Chen, David Wipf, Yu Wang, **Yang Liu**, Ian J. Wassell, Simultaneous Bayesian Sparse Approximation With Structured Sparse Models, IEEE Transactions on Signal Processing 2016 (TSP 2016).

4. Qingchao Chen, Mat Ritchie, **Yang Liu**, Kevin Chetty, Karl Woodbridge, Joint fall and aspect angle recognition using fine-grained micro-Doppler classification, IEEE Radar Conference, May 2017.

5. **Yang Liu**, Qingchao Chen, Wei Chen, Ian J. Wassell, Discriminant Dictionary Learning meets CNN in Scene Recognition. Conference on Artificial Intelligence (AAAI 2018), Feb 2018. [Chapter 4]

6. **Yang Liu**, Zhaowen Wang, Hailin Jin, Ian J. Wassell, Multi-Task Adversarial Network for Disentangled Feature Learning. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018), June 2018. [Chapter 5]

7. **Yang Liu***[1], Qingchao Chen*, Zhaowen Wang, Ian J. Wassell, Kevin Chetty, Reweighted Adversarial Adaptation Network for Unsupervised Domain Adaptation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018), June 2018. [Chapter 6]

[1] * indicates equal contributions

Chapter 2

Preliminaries

In this chapter, we give the necessary background regarding sparse representation, dictionary learning and neural networks with particular emphasis on image classification, in order to better understand our work and its motivation.

2.1 Sparse Representation and Dictionary Learning

Theoretical developments of sparse representation [31][176][183] have received much attention in the past decade and it has made a significant impact in the disciplines of computer vision and other machine learning applications. Based on sparse representation theory, a signal can be decomposed into a linear combination of a few basic signals which is capable of representing most of the information conveyed by the target signal. In this section, we review the theories and algorithms that form the basis of sparse representation and dictionary learning. We also discuss their potential for use in discriminative feature learning tasks.

2.1.1 Overview

Given the fact that most natural signals have a concise representation when expressed in some certain basis, we can use only a few nonzero entries in $\boldsymbol{\alpha}$ to reflect the features of seemingly dense data $\boldsymbol{y}$. Mathematically, finding the maximally sparse representation $\boldsymbol{\alpha} \in \mathbb{R}^N$ from its dense representation $\boldsymbol{y} \in \mathbb{R}^M$ is an ill-posed inverse problem, which can be represented as:

$$\min \|\boldsymbol{\alpha}\|_0 \ \ s.t. \ \ \boldsymbol{y} = \boldsymbol{D}\boldsymbol{\alpha}, \tag{2.1}$$

where $\boldsymbol{D} \in \mathbb{R}^{M \times N}$ is a known dictionary, $\|\boldsymbol{\alpha}\|_0$ denotes the ℓ_0 norm that counts the number of nonzero values in the vector $\boldsymbol{\alpha}$. If the objective allows for some tolerance on the fitting error:

$$\boldsymbol{y} = \boldsymbol{D}\boldsymbol{\alpha} + \boldsymbol{n}, \tag{2.2}$$

where $\boldsymbol{n}$ is the noise term, we instead consider a ***denoised*** version of optimization:

$$\min \|\boldsymbol{\alpha}\|_0 \;\; s.t. \;\; \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{\alpha}\|_2^2 \leq \varepsilon, \tag{2.3}$$

where ε upper bounds the energy of the fitting error $\boldsymbol{n}$. In the context of dictionary learning, the dictionary $\boldsymbol{D}$ is not fixed, and the task involves learning the dictionary from the data in a supervised learning step.

However, the problem expressed in Eq. (2.1) is computation intractable, which is known to be NP-hard as the search for the exact solution of $\boldsymbol{\alpha}$ is nontrivial when the data dimension becomes large. Therefore, Eq. (2.1) is often relaxed into an equivalent tractable optimisation problem. We now introduce some existing algorithms in the literature that play essential roles in calculating the maximally sparse representation, i.e., sparse coding.

2.1.2 Sparse Coding

Many approaches have been proposed in the literature to calculate the sparse representation, most of which can be classified into two categories, i.e., convex optimisation, and concave optimisation. We will briefly review these algorithms in this section. For other types of algorithms, e.g., greedy methods, Bayesian approaches, we refer the readers to the following literature [26] [51] [91] [128] [161] [173] [192].

Convex Optimization

The ℓ_1 penalty has been traditionally considered as the tightest convex approximation of the ℓ_0 problem in Eq. (2.1). Basis Pursuit (BP) [31] refers to the problem of relaxing the ℓ_0 to the ℓ_1 penalty. The convexity guarantees that the optimal solution of the problem is always the unique global solution without falling into any local minimum traps. In its noiseless form, the objective in Eq. (2.1) changes to:

$$\min \|\boldsymbol{\alpha}\|_1 \;\; s.t. \;\; \boldsymbol{y} = \boldsymbol{D}\boldsymbol{\alpha}. \tag{2.4}$$

Its denoised version, which is called Basis Pursuit De-Noising (BPDN), is correspondingly expressed as:

$$\min \|\boldsymbol{\alpha}\|_1 \;\; s.t. \;\; \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{\alpha}\|_2^2 \leq \varepsilon. \tag{2.5}$$

It is worth noting that the equivalence between Eq. (2.1) and (2.4) does not always hold unless both the signal and the dictionary meet some certain conditions, specifically the restricted isometry property (RIP) [24][25][26]. Generally, if the columns of dictionary $\boldsymbol{D}$ are highly incoherent and the concise representation $\boldsymbol{\alpha}$ is sufficiently sparse, we are confident that we can obtain the exact sparsest solution from the dense observation $\boldsymbol{y}$.

In addition to the problem represented in (2.4), some equivalent formulations also exist. For example, the denoised version of recovery problem can also be modeled as :

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{\alpha}\|_2^2 \;\; s.t. \;\; \|\boldsymbol{\alpha}\|_1 \leq \eta, \tag{2.6}$$

where $\eta \geq 0$. This formulation is called the least absolute shrinkage and selection operator (LASSO) [101]. For some choice of parameter $\lambda \geq 0$, the LASSO can be written as an unconstrained problem:

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1. \tag{2.7}$$

In principle, there is a one-to-one correspondence between λ and the sparsity bound η. Some discussion about the choice of λ is given in [55].

Convex optimisation aims to search for the globally optimal solution, and many approaches have been proposed for this purpose. Interior-point-methods are first developed for sparse coding in [31] [147], which are then improved for the compressive sensing problem in [23][95]. However, the computational complexity of Interior-point-methods are often too high for large-scale high-dimensional image data. In the light of a large number of real-world vision applications, many new efficient gradient-related algorithms have been proposed over the past decade. These efficient ℓ_1 minimization solvers include, but are not limited to, IST [42], SpaRSA[176], TwIST [16], GPSR[60], Approximately Message Passing (AMP) algorithms[50] and the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)[5].

Concave Optimization

Although the convexity of ℓ_1 minimisation avoids difficulties related to the local minimum issue, the associated global minimum of Eq. (2.4) may be biased away from the global minimum of Eq. (2.1), if dictionary coherence or the sparsity level breaks the RIP [25] requirement. An alternate solution is hence to consider replacing the convex penalty by some other non-convex penalty that not only has the same global optimum of Eq. (2.1), but

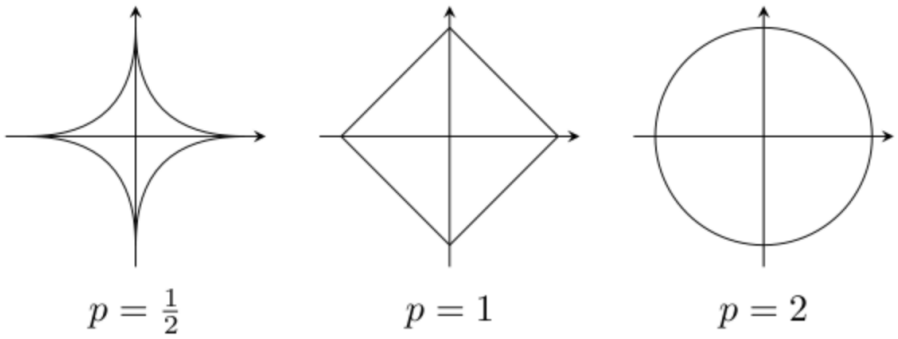


Fig. 2.1 ℓ_p norm unit ball.

also smooths away most of its local minima and also has tractable updates for the feasible solution.

It is well-known that concave, non-decreasing functions of the representation magnitudes promote sparse solutions [141][142][173]. Without giving much detail of the proof of this statement, we could instead get some basic understanding from its geometric interpretation with the help of ℓ_p norm unit ball. In Fig. 2.1, we use the ℓ_p norm penalty in a 2D case as an example. We can see that the concave penalty ($0 < p < 1$) is highly peaked at the sparse solution compared with its convex counterpart ($1 \leq p$). More specifically, we are seeking the solution that has the minimum objective function value $\|\boldsymbol{\alpha}\|_p$ that is also a feasible solution of constraint set, i.e., $\boldsymbol{y} = \boldsymbol{D}\boldsymbol{\alpha}$. From the view of Fig. 2.1, the unit ball with different norms grow from the origin point $(0,0)$ until the first tangent point in the constraint set is reached. Clearly, the concave penalty ($0 < p < 1$) is highly peaked at the sparse solution (located along the axes). In contrast, the convex penalty, e.g., ℓ_2 norm ends up with a locus that is a ball, in which the tangent point is not necessary located along the axes; thus no sparse solution is encouraged.

In this section, we consider approaches that relax the zero-norm constraint to a ℓ_p minimization problem. The ℓ_p norm used in the Focal Undetermined System Solver (FOCUSS) algorithm [75] which are defined according to $\|\boldsymbol{\alpha}\|_p = \sum_i |\alpha_i|^p$ are known to promote sparsity when $(0 < p \leq 1)$. We hence replace the constraints in the loss function representation in Eq. (2.3) with an appropriate regularization term, yielding the FOCUSS loss function:

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_p \ . \tag{2.8}$$

FOCUSS is also known by another name, specifically Iterative Reweighted Least Square (IRLS) [75]. The underlying principle used in this approach is as shown:

$$\|\boldsymbol{\alpha}\|_p = \sum_i |\alpha_i|^p = \sum_i |\alpha_i|^2 |\alpha_i|^{p-2} = \sum_i w_i |\alpha_i|^2 = \boldsymbol{\alpha}^T \boldsymbol{W}(\boldsymbol{\alpha})\boldsymbol{\alpha}, \tag{2.9}$$

where $\boldsymbol{W}(\boldsymbol{\alpha}) = diag(|\alpha_i|^{p-2})$. Replacing $\boldsymbol{\alpha}$ in $\boldsymbol{W}(\boldsymbol{\alpha})$ with a current estimate $\hat{\boldsymbol{\alpha}}$ essentially approximates the ℓ_p norm by a weighted ℓ_2 norm. IRLS optimizes a non-convex objective and therefore is only locally convergent to some sub-optimal result. Chartrand and Yin [30] introduced a regularizer ε for augmenting the IRLS, that varies from large to small as the number of iterations increases, in doing so smoothing the objective function and consequently avoiding most of local minima. Candes et al. [27] introduced an iteratively reweighted ℓ_1 minimization method, which repeatedly solves the ℓ_1 minimization problem to further boost sparsity. Wipf provided an extensive analysis on ℓ_2 and ℓ_1 reweighting schemes and made a distinction between separable and non-separable concave penalty designs [172]. However, owing to the concavity inherent in these approaches, one cannot expect the algorithm to always end up at a global minimum.

2.1.3 Sparse Representation based Classification

Sparse representation has proven to be a potent tool for classification tasks. This success is mainly because sparse representation has natural discrimination. It always chooses the most compact representation while rejecting all other possible but less compact representations. Wright et al. [175] proposed the sparse representation based classification (SRC) method for robust face recognition as illustrated in Fig. 2.2. In the SRC, the training samples themselves are used as the dictionary. Denoting the dictionary with n images from C classes as $\boldsymbol{D} = [\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_n] = [\boldsymbol{D}_1, \boldsymbol{D}_2, ..., \boldsymbol{D}_C] \in \mathbb{R}^{m \times n}$, where $\boldsymbol{D}_c$ includes n_c training images of class c. The test sample $\boldsymbol{y}$ can be represented as the linear combination of training data from the class to which it belongs. Since we do not know which class it belongs to in advance, we can only represent each test image as the linear combination of all the training samples. More specifically, we represent the test samples in an over-complete dictionary in which base elements or atoms are the training images themselves. Since there are limited numbers of images of each class, which accounts for only a small portion of the whole training set, most of the weights for different training images should be zero. Thus, the resulting weight satisfies the sparsity requirement, which means the sparsest coding vector can be found by using the efficient ℓ_1 minimisation technique or by some other sparse coding methods. The basic steps of SRC are summarised as follows:

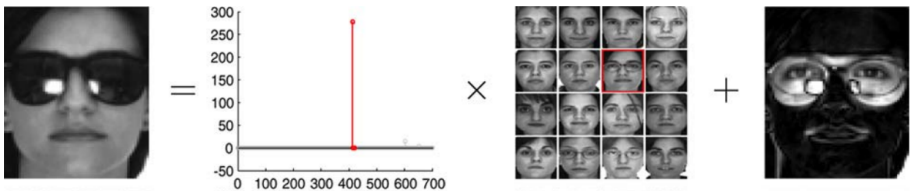


Fig. 2.2 Overview of sparse representation based classification (SRC) approach [175]. The method represents a test image (left), which is potentially occluded, as a sparse linear combination of all the training images (middle) plus sparse errors (right) due to occlusion. The red (darker) coefficients correspond to the training images of the correct individual. The SRC algorithm determines the true identity (indicated with a red box on the second row and third column) from 700 training images of 100 individuals (seven each) in the standard AR face database.

1. **Coding**: The sparse coding of the test sample $\boldsymbol{y}$ can be obtained by solving the ℓ_1 minimisation problem :

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1 \tag{2.10}$$

2. **Classification**: The test sample $\boldsymbol{y}$ is assigned to the class with the smallest residual error:

$$identity(y) = \arg\min_{c} \|\boldsymbol{y} - \boldsymbol{D}\delta_c(\hat{\boldsymbol{\alpha}})\|_2, \tag{2.11}$$

where $\delta_c(\hat{\boldsymbol{\alpha}})$ is a function that selects the coefficients associated with the c^{th} class.

Although the SRC approach has shown promising results in face recognition, the complexity of SRC can be very high owing to the use of all the training samples as the dictionary. However, the discriminative information in the training samples is not sufficiently exploited by constructing the dictionary in such a naive way. These problems can be addressed by learning an appropriate discriminative dictionary from the massive quanitity of training samples.

2.1.4 Dictionary Learning

The dictionary plays a critical role in the success of sparse representation and its application in image classification. There are two ways to build the dictionary: 1) using an off the shelf basis, e.g., wavelets, Curvelets and Fourier transform, or 2) learning a faithful and discriminative dictionary from the training data. Although an off the shelf basis [183] is universal and requires no training, the learned dictionary outperforms an off-the-shelf basis

in various computer vision applications, such as image classification [184], image denoising [2] [54].

Current prevailing dictionary learning approaches can be divided into two categories: unsupervised and supervised. The unsupervised dictionary learning methods do not take advantage of the class label information of the training samples, and their goal is to minimise the reconstruction error. The basic model of the unsupervised dictionary learning model is given as:

$$< \hat{\boldsymbol{D}}, \hat{\boldsymbol{A}} >= \arg\min_{\boldsymbol{D},\boldsymbol{A}} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{A}\|_F^2 \;\; s.t. \;\; \|\boldsymbol{\alpha}_i\|_1 \leq \eta \;\; \forall i \tag{2.12}$$

where $\boldsymbol{Y}$ is the training set, $\boldsymbol{D}$ is the dictionary to be learned, $\boldsymbol{\alpha}_i$ represents a column of the coding coefficient matrix $\boldsymbol{A}$ and $\|.\|_F$ is the Frobenius norm. Usually, the norm of each column $\boldsymbol{d}_n$ is required to satisfy $\|\boldsymbol{d}_n\|_2^2 \leq 1$. A common approach to minimise the objective function (2.12) is to update the two variables alternately, i.e., minimising w.r.t either dictionary or coefficient matrix while keeping the other fixed until a stopping criterion is met. Representative unsupervised dictionary learning methods include the Method of Optimal direction (MOD) [58] and KSVD [2]. Although these methods provide promising results in image restoration, they are not advantageous for image classification, since no label information concerning the training samples is exploited in the objective function (2.12).

If the class labels of the training samples are available, supervised dictionary learning methods can achieve a better classification performance by taking advantages of discrimination information in the dictionary learning process. In supervised dictionary learning methods, usually additional priors on the dictionary structure or the representation coefficient are leveraged in the learning procedure. The general model of supervised dictionary learning is shown as:

$$< \hat{\boldsymbol{D}}, \hat{\boldsymbol{A}} >= \arg\min_{\boldsymbol{D},\boldsymbol{A}} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{A}\|_F^2 \;\; s.t. \;\; \begin{cases} Prior(\boldsymbol{D}) \\ Prior(\boldsymbol{A}) \\ \|\boldsymbol{\alpha}_i\|_1 \leq \eta \;\; \forall i \end{cases} \tag{2.13}$$

where the constraint $Prior(\boldsymbol{D})$ makes the class-specific representation residual discriminative, the constraint $Prior(\boldsymbol{A})$ makes the representation themselves discriminative. In other words, discrimination can be exploited from the sparse representation or dictionary or both.

In the methods considering $Prior(\boldsymbol{A})$, a shared dictionary and a classifier over the representation are always learned simultaneously. All the training samples are encoded with the same shared dictionary, however, the shared dictionary loses the correspondence between the dictionary atom and the class labels. Representative approaches exploying $Prior(\boldsymbol{A})$ include for example, discriminative KSVD (D-KSVD)[194], label consistency KSVD (LC-KSVD)[92], support vector guided dictionary learning (SVGDL)[22]. The general objective

function for learning the dictionary and classifier is defined as:

$$< \hat{\boldsymbol{D}}, \hat{\boldsymbol{W}}, \hat{\boldsymbol{A}} >= \arg\min_{\boldsymbol{D},\boldsymbol{W},\boldsymbol{A}} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{A}\|_F^2 + \lambda_1 \mathcal{L}(\boldsymbol{H}, \boldsymbol{W}, \boldsymbol{A}) + \lambda_2 f(\boldsymbol{A}) + \lambda_3 f(\boldsymbol{W}) \tag{2.14}$$

where $\boldsymbol{H}$ represents the labels of the training samples, $\boldsymbol{W}$ is the parameter of the classifier, $\mathcal{L}$ is the classification loss function, $f(\boldsymbol{A})$ and $f(\boldsymbol{W})$ are the regularisers on the coefficients matrix and the classifier respectively. λ_1, λ_2 and λ_3 are scalars controlling the relative contributions of the corresponding terms.

In the methods considering $Prior(\boldsymbol{D})$, each dictionary atom is predefined to a single class label so that multiple sub-dictionaries are learned. Usually, the atoms of such class-specific dictionary should be able to well give reconstruction of images from the same class, while having poor representation of the samples from other classes. In this case, the representation residual associated with each class could be used in the classification, however, the representation coefficients are not enforced to be discriminative so that cannot be used in the classification efficiently. Mairal et al. [120] first introduced a discriminative reconstruction penalty term in the KSVD model for texture segmentation and scene analysis. To encourage the dictionaries representing different classes to be as independent as possible, Ramirez et al. [140] proposed the use of an incoherence promoting term to the dictionary learning model . Wang et al. [167] extended the class-specific dictionary learning algorithm for action recognition. Although the promising performance of class-specific dictionary representation has been reported in various of classification tasks, these dictionary learning methods might not be scalable to problems having a large number of classes.

By considering discrimination from both reconstruction residual and coding coefficients, Yang et al. [184] proposed a Fisher discrimination dictionary learning (FDDL) method, where class-specific sub dictionaries are adopted; meanwhile, the Fisher discrimination criterion is imposed on the coding vectors to boost the discrimination capability. Recently, the hybrid dictionary structure is proposed, in which the dictionary includes a shared common dictionary and a set of class-specific sub-dictionaries. Representative approaches using the hybrid dictionary structure include [98][139][198]. Although the shared dictionary could make the learned hybrid dictionary more compact, balancing the shared and the class-specific parts is not a trivial task.

2.2 Neural Networks

Neural networks encompass a broad range of statistical models, that essentially comprise a stack of linear and non-linear operations with learnable parameters. Deep neural networks have now come to dominate a diverse set of fields including computer vision, natural language understanding and speech recognition. In this section, instead of covering the long history of neural networks, we only provide an overview of the basic deep neural network architectures and some common strategies employed in the training process. For a comprehensive, in-depth overview of deep neural networks, we refer readers to the excellent references [17] [72].

2.2.1 Convolutional Neural Network

A typical convolutional neural network (CNN) is a sequenced stack of layers. The output of the previous layer is served as the input of the next layer. In this section, we will briefly introduce three main types of layers commonly used in CNN architectures: Fully-Connected Layer, Convolutional Layer and Pooling Layer. After that, we also discuss some widely used activation functions, that can be inserted between any two layers to boost representation capability.

Fully connected layers

One of the basic linear operators in a CNN is a matrix-vector multiplication, namely the fully connected (FC) layer. Mathematically, the output of the FC layer is:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} \tag{2.15}$$

where $\boldsymbol{x}$ is the input vector, $\boldsymbol{W}$ is the weight matrix and $\boldsymbol{b}$ is the bias vector. $\boldsymbol{W}$ and $\boldsymbol{b}$ are the parameters to be learned within the FC layer. FC layers are often used to combine the low-level features and act as linear classifiers at the end of a CNN. Each output dimension y_i is also called a unit, which is computed from all of the input dimensions in vector $\boldsymbol{x}$.

Convolution layers

When dealing with high-dimensional inputs such as images, it is always impractical to connect neurons in the current layer to all neurons in the previous layer (or volume). Denoting the input layer as the image that has the following dimensions: [width $H\times$ height $H\times$ depth c_1], the design of convolutional layers is motivated by only connecting each neuron to a local region of the input volume. As shown in Fig. 2.3, each convolution layer consist of c_2 filters,

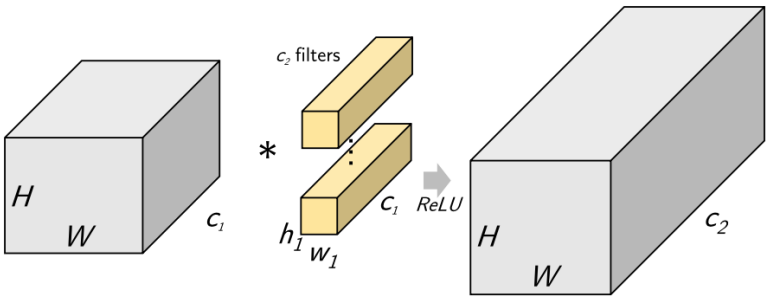


Fig. 2.3 c_2 Convolution filters with shape $h_1 \times w_1 \times c_1$.

which can be learned during training. Each filter has a small receptive field spatially (along width w_1 and height h_1), but always full along the entire depth of the input volume c_1 (e.g., all colour channels (RGB) in the first layer). In the forward pass, we slide and convolve each filter across the input volume and compute the dot product between the filter and the input at any location. In this way, given any filter, we can produce a 2-dimension activation map that gives the response of that filter at every spatial location. Intuitively, after training, each filter serves as a pattern detector that finds a certain pattern on the input image. It is worth noting that in Fig. 2.3, we assume padding appropriate to preserve the spatial dimensions $H \times W$. In one convolution layer, we stack the activation maps for different filters along the depth dimension and produce the output volume for this layer. The number of output channels is the same as the number of convolutional filters in the layer, which is c_2 in Fig. 2.3. Mathematically, the general form of 2D convolution operation [41] can be represented as:

$$(f * g)(m,n) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(i,j)g(m-i,n-j), \tag{2.16}$$

where $*$ is the convolution operator, $f(m,n)$ is an image and $g(m,n)$ is some filter function.

Pooling layers

The pooling layer, a form of non-linear spatial sub-sampling operation, is commonly periodically inserted between consecutive convolution layers. Its function is to add translation invariance to CNNs by making the network less sensitive to a small local change in the spatial location. The pooling layer also reduces the spatial size of the feature map thus reducing the computation and increasing the size of the receptive field of the following convolution layers. It is worth noting that the pooling layer operates independently on different depth slices of the input, sub-sampling using Max or Average operation [105]. Some recent CNN designs discard the pooling layer and use convolution layers with a larger stride instead, in

order to reduce the size of the feature map. The stride in this context means the step size of the convolution operation. Discarding pooling layers [85] [151] has also been shown to be important in training good generative models.

Activation Function

The activation function plays a critical role in a multi-layer neural network. Without a non-linear activation function, performing linear operations consecutively is useless since the cascade of multiple linear functions is itself equivalent to a linear function. In order to boost the representation capability of the CNN, non-linear activation functions are often inserted between every two layers. The *sigmoid* and *tanh* functions were among the first to be chosen as the activation functions. These two nonlinear functions have a nice property that, compresses the output value to a fixed range and also they have well-defined gradients. A problem with these functions is that the gradients are very small over a large portion of the domain of the function, which is also known as the saturated neuron problem. For this reason, and owing to improved empirical results, the Rectified Linear Unit (ReLu) activation function [67][99] is chosen for use in most modern neural networks. The ReLu operator has a piece-wise non-linear activation function as:

$$f(a) = \max(0, a), \tag{2.17}$$

where a is the input activation volume. ReLu does not suffer from the saturated neuron problem. In practice, Relu is very computationally efficient, and it can make the CNN training converge much faster than does *sigmoid/tanh activation*. Recently, some other advanced activation functions has been proposed, for example Leaky ReLu [84], Maxout [74] and the Exponential linear unit (ELU) [35]. A comparison of commonly used activation functions used is shown in Fig 2.4.

Another widely used non-linear activation function is *softmax*, which maps a vector $\boldsymbol{x} \in \mathbb{R}^C$ to a multinomial distribution:

$$y_i = \text{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{C} \exp(x_j)}. \tag{2.18}$$

A nice property of *softmax* activation is the output $y_i > 0$ and $\sum_{i=1}^{C} y_i = 1$. It is often adopted as the end of a CNN for image classification. It can cast the network output to a categorical distribution over C predefined classes.

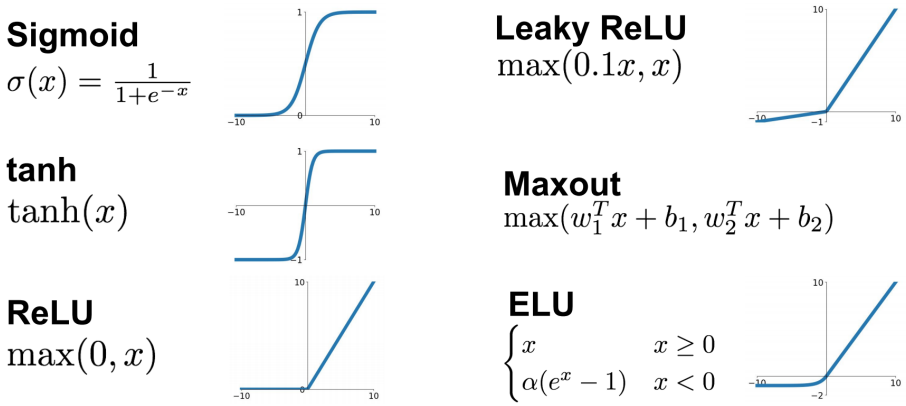


Fig. 2.4 Comparison of common activation functions.

2.2.2 Loss functions

There are two common objective functions used in CNNs under the supervised learning regime: classification and regression. For example, predicting a person's identity is a classification task, while predicting a person's age (a continuous value) is a regression task.

For classification, the target label is encoded as an one-hot vector $\boldsymbol{y} \in \mathbb{R}^C$, where C is the number of the class. The element $y_i = 1$ if the target label is the class i and all the other dimensions in $\boldsymbol{y}$ are zero. The CNN usually outputs a random variable $\hat{\boldsymbol{y}} \in \mathbb{R}^C$ as the prediction of the input. The cross-entropy [97] loss function is often used for the supervised classification task, which indicates the discrepancy between the variable $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$:

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{i=1}^{C} y_i \log \hat{y}_i. \tag{2.19}$$

For regression, the target and predicted variables can be denoted as $\boldsymbol{y} \in \mathbb{R}^d$ and $\hat{\boldsymbol{y}} \in \mathbb{R}^d$ respectively. Different distance metrics could be used as the loss function, but the ℓ_2 distance is the most widely used:

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2. \tag{2.20}$$

One important regression problem in the computer vision field is image reconstruction. Zhao er al. proposed an alternative ℓ_1 loss to make the image less blurred [196]. Using the ℓ_1 loss also improves robustness to the outliers [66].

2.2.3 Backpropagation

Due to the non-linear operations in the CNN, the objective function in Eq. (2.19) and Eq. (2.20) usually do not have a closed-form global optimum solution. Commonly, we

use stochastic gradient descent (SGD) to minimise the objective loss function instead [18]. During each iteration, we first randomly choose a subset of M training samples, also called a mini-batch, $\{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}$, where $i = 1, 2, ..., M$. Then we compute the gradient of the loss function with respect to the parameters θ in the CNN:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{1}{M} \sum_{i=1}^{M} \frac{\partial \mathcal{L}\left(f_\theta(\boldsymbol{x}^{(i)}), \boldsymbol{y}^{(i)}\right)}{\partial \theta}, \tag{2.21}$$

where $f_\theta(\boldsymbol{x}^{(i)})$ is the prediction output from the CNN. Then we can update the parameters taking a step along the direction of steepest descent:

$$\theta \leftarrow \theta - \gamma \frac{\partial \mathcal{L}}{\partial \theta}, \tag{2.22}$$

where γ is the learning rate.

As a conventional CNN contains a stack of layers, we can represent the function of the whole network f as a compound of the function of each layer f_i:

$$f(\boldsymbol{x}_0) = f_N(f_{N-1}(\cdots f_2(f_1(\boldsymbol{x}_0))\cdots)), \tag{2.23}$$

where $\boldsymbol{x}_0$ is the input of the network, N represents the number of layers in the network, and $f_i(.)$ is the function of the i^{th} layer, $i = 1, 2, ..., N$. The function $f_i(.)$, which is parameterized by θ_i takes the $\boldsymbol{x}_{i-1}$ as the layer input and outputs $\boldsymbol{x}_i$. Backpropagation is proposed by Rumelhart [146] and Lecun [104], which is an efficient approach to compute the partial gradient for the compound function based on the chain rule. We can calculate the gradients w.r.t the parameter in each layer as:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \left(\frac{\partial \boldsymbol{x}_i}{\partial \boldsymbol{\theta}_i}\right)^T \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_i}, \tag{2.24}$$

where $\frac{\partial \boldsymbol{x}_i}{\partial \boldsymbol{\theta}_i}$ is informally called the Jacobian matrix of $\boldsymbol{x}_i$ to $\boldsymbol{\theta}_i$, and $\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_i}$ can be calculated recursively,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_i} = \boldsymbol{J}_{i+1}^T \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_{i+1}}, \tag{2.25}$$

where $\boldsymbol{J}_{i+1}^T$ is the Jacobian matrix of $\boldsymbol{x}_{i+1}$ with respect to $\boldsymbol{x}_i$. In practice, we first forward pass a mini-batch from the first layer to the last layer and save all the intermediate activations in the temporary buffer. Then we can compute the gradient with respect to the θ_i for each layer and update them for one step.

2.2.4 Strategies in Training Neural Networks

In this section, we outline and summarise some strategies for training the neural network, including network initialisation, batch normalisation and dropout. These techniques are essential for accelerating the training phase and simplifying the optimisation process.

Network Initialization

Without carefully network initialization, gradients can either explode or vanish entirely. For instance, consider a network of N layers as described by Eq. (2.23). Denoting $\boldsymbol{x}$ as the input, if each layer $f_i(.)$ has identical mapping $f_i(\boldsymbol{x}) = \beta\boldsymbol{x}$, the output of the deep network will be:

$$f(\boldsymbol{x}_0) = f_N(f_{N-1}(\cdots f_2(f_1(\boldsymbol{x}_0))\cdots)) = \boldsymbol{x}\prod_i^N \beta^N. \tag{2.26}$$

There will be two different outcomes depends on the selection of scaling factor β in the weight initialization:

$$\lim_{N\to\infty} f(\boldsymbol{x}) = \begin{cases} \infty & if \ \ \beta > 1 \\ 0 & if \ \ \beta < 1 \end{cases} \tag{2.27}$$

In other words, when $\beta > 1$, the loss is divergent thus make the gradient explode, while when $\beta < 1$, the loss is stalled therefore making the gradient vanish. Intuitively, we usually initialize the weights from a Gaussian distribution with a standard variance σ such that the expected value $E(\beta) = 1$ in order to avoid gradient exploding or vanishing problems.

In practice, denoting n_{in},n_{out} as the number of units in the previous layer and the next layer respectively, for *tanh* activation, Glorot et al. proposed to use Xavier initialization [67] by setting

$$\sigma = \sqrt{\frac{1}{n_{in}}}. \tag{2.28}$$

When considering both the number of units in the previous layer and the next layer, a recommended setting is

$$\sigma = \sqrt{\frac{2}{n_{in} + n_{out}}}. \tag{2.29}$$

For the *ReLu* activation, He et al. [84] reaching the conclusion that the standard variance of weights in the network should be

$$\sigma = \sqrt{\frac{2}{n_{in}}}. \tag{2.30}$$

Batch Normalization

A technique developed recently by Ioffe et al., known as Batch Normalization [89], alleviates a lot of the difficulties in properly initialising the parameters in the network. It provides a direct and neat approach to maintaining the desired activation distribution, i.e., a zero mean, unit variance Gaussian distribution. The batch Normalization, which is a simple differential operation, uses the batch statistic to whiten the layer response. We first calculate the mean and variance of the mini-batch and prevent gradient explosion/vanishing problem by normalising the response/gradient based on the batch statistic. Given M samples in a mini-batch and each denoted by a vector $\boldsymbol{x}_i \in \mathbb{R}$, $i = 1,2,...,M$. The batch normalisation process can be described as:

$$\boldsymbol{\mu}_b = \frac{1}{M}\sum_{i=1}^{M}\boldsymbol{x}_i, \tag{2.31}$$

$$\boldsymbol{\sigma}_b^2 = \frac{1}{M}\sum_{i=1}^{M}(\boldsymbol{x}_i - \boldsymbol{\mu}_b)^2 \tag{2.32}$$

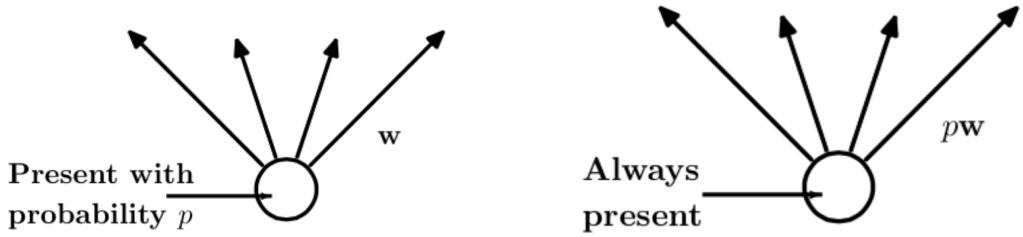
$$\hat{\boldsymbol{x}}_i = \frac{\boldsymbol{x}_i - \boldsymbol{\mu}_b}{\sqrt{\boldsymbol{\sigma}_b^2 + \varepsilon}}, \tag{2.33}$$

where for mini-batch b, the mean is $\boldsymbol{\mu}$, the variance is $\boldsymbol{\sigma}^2$, and ε is a parameter chosen to address numerical stability issues. For a given layer response $\boldsymbol{x}_i$, the normalized output response is $\hat{\boldsymbol{x}}_i$. Finally batch normalization scales and shifts the normalized response according to:

$$\boldsymbol{y}_i = \gamma\hat{\boldsymbol{x}}_i + \beta, \tag{2.34}$$

where $\gamma, \beta \in \mathbb{R}^d$ are learnable parameters.

In practice, we always insert the batch normalisation after fully connected or convolution layers, and before nonlinear activations. Batch normalisation can improve the gradient flow through the network and allow for higher learning rates. Empirically, it has been found that using batch normalisation significantly improves robustness to poor initialisation and sometimes can even speed up the training process. It is worth noting that the batch normalisation works differently in the test stage. The statistics (mean and variance) are not computed based on the batch. Instead, the single fixed empirical statistics of the activations achieved during training is used.



(a) Dropout in the training stage (b) Dropout in the testing stage

Fig. 2.5 Dropout

Dropout

Srivastava et al. [152] proposed dropout, a method of preventing overfitting in deep network training. In constrast to adding different norm regularisation, i.e., ℓ_1, ℓ_2 norms on the parameters, dropout is implemented by only keeping a neuron active with some probability p in the training stage. As shown in Fig. 2.5(a), in the training stage, dropout acts as the sampler in the neural network, and only updates the parameters of the sampled network based on the input data in the current step. The dropout can also be interpreted as a form of the model ensemble, averaging a large number of random 'smaller' sampled networks to obtain better generalisation capability. As shown in the Fig. 2.5(b), in the test stage, no dropout is applied, and all the neurons are always present, and so we need to adjust output $\boldsymbol{x} \rightarrow p\boldsymbol{x}$ to ensure the same expected output. Performing this multiplication at test time can be related to the process of iterating over all the possible sampled sub-networks and computing their ensemble prediction.

2.2.5 Deep Neural Network Architectures

In this section, we list some of the modern popular deep neural network architectures, that have formed the basis of many of our experiments.

AlexNet

Krizhevsky et al. propose AlexNet [100], which demonstrated that the deep CNN could achieve state-of-the-art performances in large-scale image classification tasks. With appropriate initialization [156], ReLu activation and [126] drop out [152] techniques, deep CNN outperforms traditional hand-crafted feature-based approaches by a large margin. Historicallly Graphics processing unit (GPU) did not have enough memory to support such a large network, consequently, AlexNet uses two filter groups in most layers to split the computation demand and model parameters across two GPUs. However, it is worth noting such a network

split is usually no longer necessary owing to improved GPU memory capacity. The overall structure of AlexNet is shown in Fig. 2.6(a).

VGG

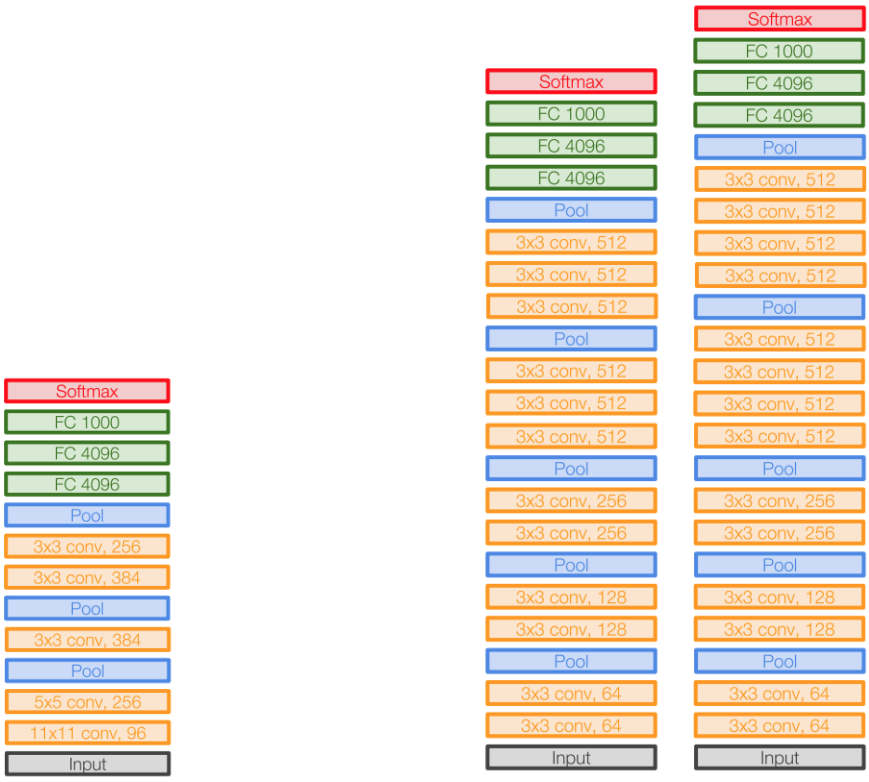
Since AlexNet, there have many improvements to the state-of-the-art result on the ILSVRC challenge [14] achieved by using improved CNN architectures [85] [157] [191]. In the ILSVRC 2014 challenge, one particular architecture called the VGG network [150] achieved 2^{nd} place in classification, and 1^{st} place in localisation performance by employing a natural extension of AlexNet. It follows two design principles: small filters and a deeper network. More specifically, using a stack of three 3×3 convolution layers (with stride 1) has same effective receptive field as one 7×7 convolution layer. This observation allows using a deeper network (having more non-linearities) with far fewer parameters thus increasing computational efficiency. The VGG network also demonstrated that using small filters and a deeper network could significantly improve generalisation. The prevailing VGG architecture has 16 or 19 layers, and the overall network architectures are shown in Fig. 2.6(b).

Inception

As measured by classification accuracy performance, the winner of the ILSVRC 2014 is the Inception architecture, also known as GoogLeNet [157]. This work proposes an *Inception Module*, which is a good local network topology (network within a network). The overall Inception network architecture stacks these modules on top of each other, which also enables a deeper network (22 layers) with efficient computation. The naive inception module design is shown in Fig. 2.7(a). In each inception module, parallel filter operations are performed on the input from the previous layer, including multiple filter sizes for convolution (1×1, 3×3, 5×5) and pooling (3×3) operations. After performing these different operations, the filter outputs are concatenated together depth-wise. As the pooling layer preserves feature depth, this means that the total depth after concatenation can only grow at each layer. To reduce the computation complexity, GoogLeNet also proposes using a 1×1 convolution layer to act as so called the 'bottleneck' layers, in order to reduce the feature depth. The inception module with dimension reduction is shown in Fig. 2.7(b).

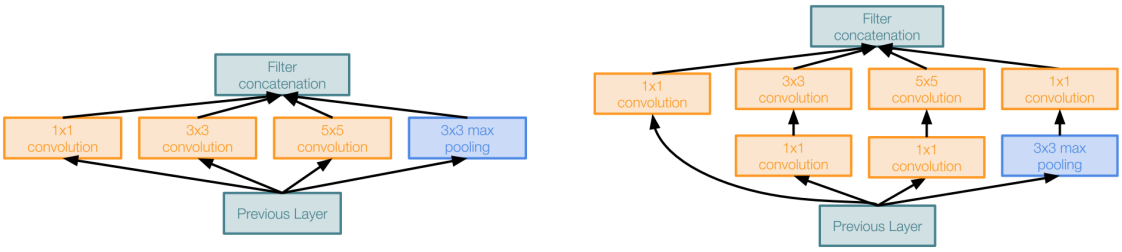
Residual Net

He et al. proposed residual net (ResNet) [85], that swept to 1^{th} place in all ILSVRC and COCO 2015 competitions. The underlying hypothesis of this paper is that the deeper model should be able to perform at least as well as the shallower model. He et al. are among the



(a) Alex Net architecture (b) VGG16 (left) and VGG19 (right) network architecture

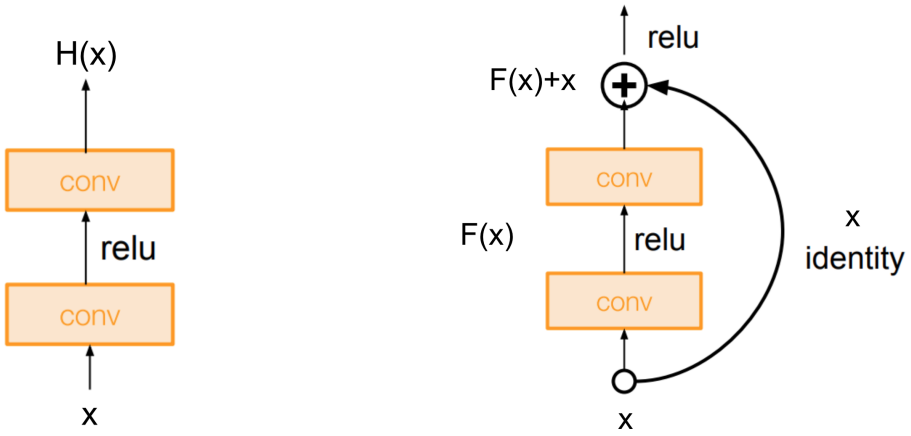
Fig. 2.6 AlexNet and VGGNet architectures. The description '$m \times m$ conv n' denotes there are n filters in this convolutional layer, each with spatial size $m \times m$. The description 'FC n' indicates there are n neurons in the fully connected layer.



(a) Naive Inception Module (b) Inception module with dimension reduction

Fig. 2.7 Inception Modules

first to observe that directly concatenating lots of layers into a very deep network (152 layers) will lead to a performance degradation owing to the difficulty in the optimisation. ResNet propose to use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping. A comparison of plain layer concatenation and the residual block is shown in Fig. 2.8. Given input $\boldsymbol{x}$, the residual block fits residual $F(\boldsymbol{x}) = H(\boldsymbol{x}) - \boldsymbol{x}$



(a) Plain layer concatenation (b) Residual block

Fig. 2.8 Comparison of plain layer concatenation and residual block

instead of $H(\boldsymbol{x})$ directly. The full residual net architecture stacks multiple residual blocks, each of them containing two convolution layers. It is worth noting that for deeper residual nets, inception modules have also been adopted in the middle in order to reduce computation complexity.

2.3 Application of Prior Information

Sparse representation and deep learning are presented as the basis for several general-purpose priors, which are used to explore a better feature representation for robust image classification. In this section, we summarize some related work regarding the application of prior information into the feature learning procedure.

Some existing related work applies sparse or hierarchical priors to the explanatory factors that describe the data in order to better discover the data structure. Discriminative dictionary learning (DL) methods [92][185] aim to incorporate the sparse prior into the feature learning procedure by finding the optimal dictionary that simultaneously improves the sparse representation and also maximizes its discriminative capability. However, current DL methods cannot achieve state of the art performance in large-scale image classification, since most DL models have only been evaluated with the use of traditional handcrafted features. In [154], it is observed that the deep neural network design incorporates hierarchical priors into the feature learning procedure by learning multiple levels of representation. In addition the nonlinear property inherent in the ReLu unit leads to sparsity in the neural activations, consequently improving the performance of deep learning methods. Normally, for each training image, around half of the neurons in the hidden layer are activated. However,

there is no effective mechanism to automatically adjust the sparsity level of the intermediate hidden units based on the training set. It is worth noting that [108][178] incorporate both sparse and hierarchical priors into feature learning by applying DL as a post-processing step trained separately from the training of the CNN. Arguably, this does not fully harness the strength of DL since it is not integrated with the deep network. To the best of our knowledge, there is no end-to-end learning framework to combine CNN and DL in image recognition, that can automatically adjust the sparsity level and discriminative capability of the feature representation. More detailed discussion about the difference between the most closely related prior works and ours that consider incorporating sparse and hierarchical priors into the feature learning will be discussed in Chapter 3 and 4.

There is some related work try to explore priors on the relationships between multiple factors. When multiple independent factors exist in the image generation process, there is a quantity of literature concerning learning disentangled representations. The bi-linear model is among the first to separate the content and style in the underlying set of observations [159]. With the recent development of deep learning, auto-encoders [33] [87] [88] and Boltzmann machines [143] are adopted as regularizers to combine the discrimination and self-reconstruction criteria, thus discovering the factors of variation besides those relevant for classification. In particular, Predictability Minimization [148] and the fair variational auto-encoder [115] encourage independence between different latent factors. In addition to reconstructing the input, [135] [188] synthesize other images with the same content but with a different style to implicitly disentangle features. With the help of GANs, the work of [45] [53] [102] [124] further explores the application of disentangled representations in computer graphics and video prediction. More detailed discussion about the difference between the most closely related prior works and ours regarding feature disentangling will be discussed in Chapter 5.

We observe that when common factors exist in multiple tasks, there is some related work leveraging common factors to make the learned feature representation more robust. Unsupervised domain adaptation (UDA) is a specific approach that address this situation, with the aim of transferring domain knowledge (common factors) from existing well-defined tasks to new ones where labels are unavailable. A lot of research has been conducted investigating measurements to estimate the distribution divergence of deep features among domains and the relevant methods to minimize them. As an unbiased estimate of distribution divergence, Maximum Mean Discrepancy (MMD) [77] has been employed in various CNN based methods for UDA [110] [113] [114] [163] [165] [179]. More recently, inspired by the best-performing adversarial training methods in generative models, state-of-the-art UDA methods utilize the Jenson-Shannon (JS) divergence or the more generalized f-divergence

[130] implemented using CNNs to estimate the distribution divergence [19] [61] [62] [109] [162]. However, both the MMD and f-divergence based methods require that the feature distribution of the source and target domain share a common support (i.e., significant overlap exists between the feature distributions of two domains). These methods fail to adapt between domains once their distributions do not have significant overlap. More detailed discussion about the difference between the most closely related prior works and ours for leveraging common factors in feature learning will be discussed in Chapter 6.

2.4 Chapter Summary

In this chapter, we first introduced the fundamental concepts of sparse representation, including sparse coding model, sparse representation based classification and dictionary learning. Then we focused on the topic of deep learning, including convolutional neural network and some training techniques. The sparse representation and deep learning are presented as the basis for several general-purpose priors, which are used to explore a better feature representation for robust image classification. For the remainder of this dissertation, we will present our enhancements by incorporating sparse and hierarchical priors in chapter 3 and 4, while improvements wrought by combining the prior knowledge about the relationship between multiple factors, in particular, independent or common explanatory factors, will be given in Chapters 5 and 6.

Chapter 3

Support Discrimination Dictionary Learning

Sparse representation and the associated learning techniques have received much attention in the past decade as introduced in Chapter 2. There are three aspects concerning the benefits of using a sparse prior to the feature learning model, as discussed below:

Firstly, the sparse prior encourages information disentangling. One of the claimed objectives of representation learning algorithms [12] is to disentangle the factors explaining the variations in the data. A dense representation is highly entangled, as only a small change in the input will modify most of the entries in its feature representation. Instead, if a representation is both sparse and robust to small input changes, the set of nonzero features (support) is almost always nearly conserved in response to a small change of input. Therefore, correctly using the sparse prior in the feature learning process can make the learned feature representation more robust to a variety of photometric and geometric deformations, such as a change of lighting conditions or a shift in the object location within an image.

Secondly, using the sparse prior implicitly allows variable-size feature representation for different inputs. As different inputs may contain different amount of information, it would be efficient to represent data using a variable-size data-structure. Varying the size of support size controls the effective dimension of the representation for a given input and the required precision.

Thirdly, although there has been a recent rapid increase in research activity on large-scale datasets, many small data sets still arise and already exist in the wild and could underpin in a lot of useful applications. Such small datasets are more difficult to handle owing to the possibility of over-fitting, and outliers become much more significant, thus dramatically reducing the classification accuracy. A sparse prior can serve as a regularizer to reduce the overfitting problem, consequently providing better image classification when only limited

labelled data is available. The sparse feature can sometimes even make the representation easier to explain, as only the most meaningful (explanatory) features remain.

In this chapter, we focus on exploring the role of a sparse prior in feature learning for the image classification task. By incorporating the structured sparse prior, we propose a new support discrimination dictionary learning method that uses the sparse structure, i.e., support, to learn a discriminant and robust feature representation for image classification. The rest of this chapter is organized as follows: Section 3.1 provides the most relevant prior works about combining sparse prior with feature learning approaches and demonstrates the primary motivation and contribution of this chapter; Section 3.2 presents the details of the novel support discrimination dictionary learning method for classification, including the optimisation algorithm and the classification scheme; In section 3.3, extensive experiments are performed on face, object and scene datasets to compare the proposed method with other state-of-art dictionary learning methods; Section 3.4 concludes this chapter.

3.1 Introduction

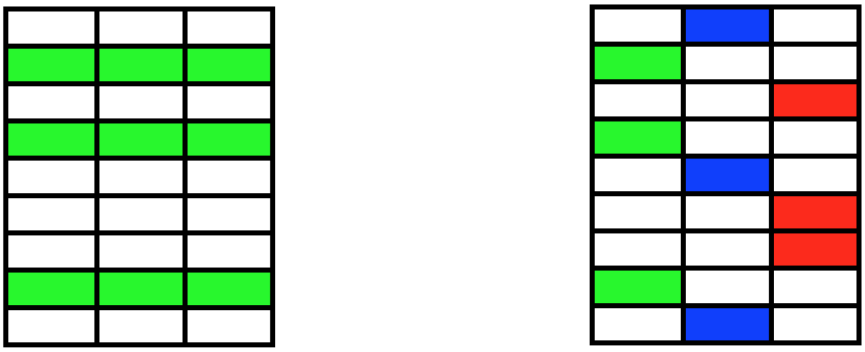
Sparse representation has been successfully applied to a variety of problems in image processing and computer vision, e.g., image denoising, image restoration and image classification. In the framework of sparse representation, an image can be represented as a linear combination of a few bases selected sparsely from an over-complete dictionary. The dictionaries can be predefined by the use of some off-the-shelf basis, such as the Discrete Fourier Transform (DFT) matrix and the wavelet matrix. However, it has been shown that learning the dictionary from the training data enables a more sparse representation of the image in comparison to using a predefined one, which can lead to improved performance in the reconstruction task. Some typical reconstruction dictionary learning methods include the Method of Optimal direction (MOD) [58], and K-SVD [2].

Sparse representation has also been considered in pattern recognition applications. For example, it has been used in the Sparse Representation based Classifier (SRC) [175], which achieves competitive recognition performance in face recognition. In contrast to image reconstruction which only concerns the sparse representation of an image, in pattern recognition, the main goal is to find the correct label for the query sample, consequently the discriminative capability of the learned dictionary is crucial. A variety of discriminative dictionary learning methods have recently been proposed, some of which have been briefly introduced in Chapter 2. The most relevant prior works on incorporating sparse prior and dictionary learning to feature learning are elaborated as follows. In general, we observe that two different strategies have been employed.

One strategy is to learn a class-specific dictionary, which discriminates different classes of images via a sparse representation residual. Instead of learning a dictionary shared by all classes, it seeks to learn a sub-dictionary for each class. Yang et al. [181] first sought to learn a dictionary for each class, and applied it to image classification. In [57], instead of considering the dictionary atoms individually at the sparse coding stage, the atoms are selected in groups according to some priors to guarantee the block sparse structure of each coding coefficient. In [155], a group-structured dirty model is used to achieve a hierarchical structure of each coding coefficient via estimating a superposition of two coding coefficients and regularising them differently. It is worth noting that the multi-task setting is adopted in [155]. However, the sub-dictionaries in all these methods are disjoint to each other, and how many and which atoms belong to each class is fixed during the entire dictionary learning process. In addition, although class-specific setting of the dictionary works well when the number of training samples in each class is sufficient, it is not scalable to the problem with a large number of classes.

Another strategy is to learn a dictionary that is shared by all classes. Commonly, a classifier based on the coding vectors is learned together with the shared dictionary by imposing some class-specific constraints on the coding vector. Rodriguez et al. [144] proposed that samples of the same class should have similar sparse coding vectors which are achieved by using linear discriminant analysis. Yang et.al. [184] proposed Fisher discrimination dictionary learning (FDDL) where the Fisher discrimination criterion is imposed on the coding vectors to enhance class discrimination. Cai et.al. proposed support vector guided dictionary learning methods (SVGDL) [22], which is a generalised model of FDDL, that considers the squared distances between all pairs of coding vectors. In all these methods, the similarity between two coding vectors is measured by the Euclidean distance, which allows two images of different classes to be represented by using the same set of dictionary atoms. To our knowledge so far, no multi-task setting has been used in the shared dictionary, since it is difficult to discriminate groups of coefficients between different classes owing to the lack of prior knowledge concerning subdictionary structure.

In recent years, it has been shown that adding structural constraints to the supports of coding vectors can result in improved representation robustness and better signal interpretation [56][90][190]. In our approach to be presented in this chapter, the multi-task setting adopts a shared dictionary, however, instead of learning the dictionary with discrimination based on the Euclidean distance between the coefficients for different classes, we consider a different principle as shown in Fig. 3.1: **The support of the coding vectors from one class should be similar, while the support of the coding vectors from different classes**



(a) Same Class:common sparse structure

(b) Different classes: minimise overlapped support

Fig. 3.1 The design principle of our proposed approach.

should be dissimilar. Here the support of a coding vector denotes the indices of the non-zero elements of the image sparse representation under some dictionary.

More specifically, by incorporating structural sparse priors into the feature learning procedure, we propose a support discrimination dictionary learning method (SDDL), that finds a dictionary under which the coefficients of images from the same class have a common sparse structure while the size of the overlapped signal support of different classes is minimised. Informed by the multitask learning framework [116], and the multiple measurement vector (MMV) model [37] in the signal processing field, an effective way to encourage a group of signals to share the same support is to simultaneously encode those samples. Based on this idea, we encode multiple images from the same class, requiring that their coefficient matrix is largely 'row sparse', where only a few rows have non-zero elements, as shown in Fig. 3.1(a). In addition to the similarity of intra-class coding vectors, the main contribution of our work is that we also design a new discriminative term to guarantee the dissimilarity of inter-class coding vectors by reducing the overlapped signal support from different classes, as shown in Fig. 3.1(b). This can be achieved by minisation of the ℓ_0 norm of the Hadamard product between any pair of coefficients in different classes. An iterative reweighting scheme that produces more accurate estimates is adopted as the optimization progresses.

Besides the advantage of sparse priors as discussed at the beginning of this chapter, the specific design of SDDL provides the following advantages. Firstly, the previous multi-task setting based dictionary learning methods all use disjoint sub-dictionaries, in which how many and which atoms belong to each class is fixed during the entire dictionary learning process. In contrast, a multi-task setting using a shared dictionary is adopted in SDDL. Our approach can automatically identify overlapped sub-dictionaries for different classes, where the size of each sub-dictionary is adjusted appropriately during the learning process to suit the

training dataset. Furthermore, our approach is scalable to allow for a large number of classes, while the previous sub-dictionary based approaches cannot. Secondly, instead of using the Euclidean distance to measure the similarity and dissimilarity between different coefficients, we achieve discrimination via the support. The structural sparse constraints ease the difficulty in solving the ill-posed inverse problem in comparison to the conventional element-sparse structure [43]. The superior performance of the proposed approach in comparison to the state-of-art is demonstrated using both face, object and scene datasets.

3.2 Support Discrimination Dictionary Learning

3.2.1 Problem Formulation

Assume that $\boldsymbol{x} \in \mathbb{R}^m$ is a m dimensional image with class label $c \in \{1,2,...,C\}$, where C denotes the number of classes. The training set with n images is denoted as $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n] = [\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_C] \in \mathbb{R}^{m \times n}$, where $\boldsymbol{X}_c$ includes n_c training images of class c. The learned dictionary is denoted by $\boldsymbol{D} = [\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_K] \in \mathbb{R}^{m \times K} (K < n)$, where $\boldsymbol{d}_k$ denotes the k^{th} atom of the dictionary. $\boldsymbol{A} = [\boldsymbol{A}_1, \boldsymbol{A}_2, ..., \boldsymbol{A}_C] = [\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_n] \in \mathbb{R}^{K \times n}$ are the coding coefficients of $\boldsymbol{X}$ over $\boldsymbol{D}$. Our dictionary learning problem can be described as

$$\min_{\boldsymbol{D},\boldsymbol{A}} R(\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}) + w_1 g(\boldsymbol{A}) + w_2 f(\boldsymbol{A}), \tag{3.1}$$

where $R(\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A})$ denotes the reconstruction residuals for all the images $\boldsymbol{X}$ with the sparse representation matrix $\boldsymbol{A}$ under the dictionary $\boldsymbol{D}$, $g(\boldsymbol{A})$ is a regulariser to promote intra-class similarity, $f(\boldsymbol{A})$ is the inter-class discriminative term based on the coding vectors $\boldsymbol{A}$, and $w_1 > 0$ and $w_2 > 0$ denote the weights for the final two terms in (3.1). In this optimisation problem, we learn a single dictionary shared among all classes while exploring the discrimination of the coding vectors.

In a common multi-task learning setting, a group of tasks share certain aspects of some underlying distribution. Here we assume the intra-class coding vectors share a similar sparse structure. In our fomulation, we use the joint sparsity regulatisation ℓ_p/ℓ_q norm of a coefficient matrix corresponding to one class, rather than encoding each training image separately. More specificaly, we set $p = 2, q = 0$, which means that the intra-class coefficient matrix should be 'row sparse', i.e., where each row is either all zero or mostly non-zero, and the number of non-zero rows is low. In this way, we can find the shared nonzero supports for each class automatically, rather than predefining their number and position. However,

minimizing the ℓ_2/ℓ_0 norm is NP hard, so in our approach, we use ℓ_2/ℓ_1 norm instead. In this way, we can design a regulariser to promote intra-class similarity as:

$$g(\boldsymbol{A}) = \sum_{i=1}^{C} \|\boldsymbol{A}_i\|_{2,1} = \sum_{i=1}^{C} \sum_{k=1}^{K} \left\|\boldsymbol{a}^{(ik)}\right\|_2, \tag{3.2}$$

where $\boldsymbol{A}_i$ represents the coefficient matrix for the i^{th} class and $\boldsymbol{a}^{(ik)}$ denotes the k^{th} row of coefficient matrix $\boldsymbol{A}_i$.

In general, discrimination for different classes can be assessed by the similarity of the intra-class coding vectors and the dissimilarity of inter-class ones. As mentioned previously, the similarity of intra-class coding vectors is promoted by the ℓ_2/ℓ_1 regulariser. To encourage dissimilarity of the inter-class coding vectors, we design the following discriminative term:

$$f(\boldsymbol{A}) = \sum_{i=1}^{C} \sum_{p} \sum_{q} \left\|\boldsymbol{a}_{i,p} \circ \boldsymbol{a}_{/i,q}\right\|_0, \tag{3.3}$$

where $\circ$ denotes the Hadamard (elementwise) product between two vectors $\boldsymbol{a}_{i,p}$ and $\boldsymbol{a}_{/i,q}$, where $\boldsymbol{a}_{i,p}$ and $\boldsymbol{a}_{/i,q}$ are the p^{th} column of $\boldsymbol{A}_i$ and the q^{th} column of $\boldsymbol{A}_{/i}$ respectively. $\boldsymbol{A}_i \in \mathbb{R}^{K \times n_i}$ represents the coefficient matrix for the i^{th} class, while $\boldsymbol{A}_{/i} \in \mathbb{R}^{K \times (n-n_i)}$ denotes a sub-matrix of $\boldsymbol{A} \in \mathbb{R}^{K \times n}$ without the columns in $\boldsymbol{A}_i$. Alternatively, the value of $\left\|\boldsymbol{a}_{i,p} \circ \boldsymbol{a}_{/i,q}\right\|_0$ is the size of the overlapped support between the p^{th} image of the i^{th} class and the q^{th} image that is not in class i. Therefore, $f(\boldsymbol{A})$ denotes the summation of overlapped supports between images in different classes. However, minimising $f(\boldsymbol{A})$ in Eq. (3.3) is an NP hard problem. Informed by many recent sparse approximation algorithms that rely on iterative reweighting schemes [27][30][171] to produce more focal estimates as optimization progresses, we use the iterative reweighted ℓ_2 minimization to approximate the ℓ_0 norm.

We use the vector $\boldsymbol{a}^{\odot 2}$ to represent the element by element square of vector $\boldsymbol{a}$, which is equal to $\boldsymbol{a} \circ \boldsymbol{a}$. We define the weight term $\boldsymbol{w}_{p,q}$ for a given pair of coefficients $(\boldsymbol{a}_{i,p}, \boldsymbol{a}_{/i,q})$ at each iteration as a function of those coefficients from the previous iteration as

$$\boldsymbol{w}_{i,p,q} = \frac{1}{(\boldsymbol{a}'_{i,p} \circ \boldsymbol{a}'_{/i,q})^{\odot 2} + \varepsilon} \tag{3.4}$$

where $\boldsymbol{a}'_{i,p}$ and $\boldsymbol{a}'_{/i,q}$ are the coefficients from the previous iteration and ε is a regularization factor that is reduced to zero as the number of iterations increases. In this case, the descrimination term $f(\boldsymbol{A})$ can be rewritten as

$$
\begin{aligned}
f(\boldsymbol{A}) &= \sum_{i=1}^{C}\sum_{p}\sum_{q}\left\|\boldsymbol{a}_{i,p}\circ\boldsymbol{a}_{/i,q}\right\|_0 = \sum_{i=1}^{C}\sum_{p}\sum_{q}\sum_{k} w_{i,p,q}^{(k)}\cdot(a_{i,p}^{(k)}\circ a_{/i,q}^{(k)})^2 \\
&= \sum_{i=1}^{C}\sum_{p}\sum_{q}\sum_{k}[w_{i,p,q}^{(k)}\cdot(a_{/i,q}^{(k)})^2]\circ(a_{i,p}^{(k)})^2 \\
&= \sum_{i=1}^{C}\sum_{p}\sum_{q} diag([\boldsymbol{w}_{i,p,q}\circ(\boldsymbol{a}_{/i,q})^{\odot 2}]\cdot(\boldsymbol{a}_{i,p})^{\odot 2} = \sum_{i=1}^{C}\sum_{p}\left\|\boldsymbol{\Omega}_{i,p}\boldsymbol{a}_{i,p}\right\|_F^2,
\end{aligned}
\tag{3.5}
$$

where k represents the index of the corresponding vector and

$$
\boldsymbol{\Omega}_{i,p} = diag(\sqrt{\sum_{q}(\sqrt{\boldsymbol{w}_{i,p,q}}\circ\boldsymbol{a}_{/i,q})^{\odot 2}}). \tag{3.6}
$$

However, minimising the above $f(\boldsymbol{A})$ is both time and memory consuming since we need to calculate a weight vector $\boldsymbol{w}_{i,p,q}$ and thus a distinct weight matrix $\boldsymbol{\Omega}_{i,p}$ for each $\boldsymbol{a}_{i,p}$. Considering the effect of the ℓ_2/ℓ_1 regulariser, different coefficients in the same class should have a similar sparse pattern, hence we use the average $(\tilde{\boldsymbol{a}_i'})^{\odot 2}$ instead of $(\boldsymbol{a}_{i,p}')^{\odot 2}$ in Eq. (3.4), where

$$
\forall p,\ \ (\boldsymbol{a}_{i,p}')^{\odot 2} \approx (\tilde{\boldsymbol{a}_i'})^{\odot 2} = \sum_{p}(\boldsymbol{a}_{i,p}')^{\odot 2}/n_i. \tag{3.7}
$$

That is, all p images of the class i share the same weight vector $\boldsymbol{w}_{\tilde{i},q}$ as

$$
\boldsymbol{w}_{\tilde{i},q} = \frac{1}{(\tilde{\boldsymbol{a}_i'})^{\odot 2}\circ(\boldsymbol{a}_{/i,q}')^{\odot 2}+\varepsilon}. \tag{3.8}
$$

Finally Eq. (3.5) can be rewritten as:

$$
f(\boldsymbol{A}) = \sum_{i=1}^{C}\sum_{p}\left\|\boldsymbol{\Omega}_{i,p}\boldsymbol{a}_{i,p}\right\|_F^2 = \sum_{i=1}^{C}\left\|\tilde{\boldsymbol{\Omega}}_i\boldsymbol{A}_i\right\|_F^2, \tag{3.9}
$$

where

$$
\tilde{\boldsymbol{\Omega}}_i = diag(\sqrt{\sum_{q}(\sqrt{\boldsymbol{w}_{\tilde{i},q}}\circ\boldsymbol{a}_{/i,q})^2}). \tag{3.10}
$$

By substituting the discrimination term given by Eq. (3.9) into (3.1), we can rewrite the dictionary learning problem as

$$
\min_{D,A}\sum_{i=1}^{C}\left\|\boldsymbol{X}_i-\boldsymbol{D}\boldsymbol{A}_i\right\|_F^2 + w_1\left\|\boldsymbol{A}_i\right\|_{2,1} + w_2\left\|\tilde{\boldsymbol{\Omega}}_i\boldsymbol{A}_i\right\|_F^2. \tag{3.11}
$$

Although the objective function in (3.11) is not jointly convex to $(\boldsymbol{D},\boldsymbol{A})$, it is convex with respect to $\boldsymbol{D}$ and $\boldsymbol{A}$ when the other is fixed. In the sequel, we provide an algorithm which alternately optimises $\boldsymbol{D}$ and $\boldsymbol{A}$.

3.2.2 Optimisation

Finding the solution of the optimisation problem in (3.11) involves two sub-problems, i.e., to update the coding coefficients $\boldsymbol{A}$ with fixed $\boldsymbol{D}$, and to update $\boldsymbol{D}$ with fixed coefficients $\boldsymbol{A}$.

First suppose that $\boldsymbol{D}$ is fixed, and the optimisation problem can be reduced to a sparse coding problem to calculate $\boldsymbol{A} = [\boldsymbol{A}_1, \boldsymbol{A}_2, .., \boldsymbol{A}_C]$ with two constraints. Here, we compute the coefficients matrix $\boldsymbol{A}_i$ class by class. More specifically, all $\boldsymbol{A}_j (\boldsymbol{j \neq i})$ are fixed thus $\tilde{\boldsymbol{\Omega}}_i$ is fixed when computing the $\boldsymbol{A}_i$. In this way, the objective function can be further reduced to

$$\min_{\boldsymbol{A}_i} \|\boldsymbol{X}_i - \boldsymbol{D}\boldsymbol{A}_i\|_F^2 + w_1 \|\boldsymbol{A}_i\|_{2,1} + w_2 \left\|\tilde{\boldsymbol{\Omega}}_i \boldsymbol{A}_i\right\|_F^2. \tag{3.12}$$

We choose the alternating direction method of multipliers (ADMM) as the optimisation approach because of its simplicity, efficiency and robustness [21][43][182]. By introducing one auxiliary variable $\boldsymbol{Z}_i = \boldsymbol{A}_i \in \mathbb{R}^{K \times n_c}$, this problem can be reformulated as

$$\min_{\boldsymbol{A}_i, \boldsymbol{Z}_i} \|\boldsymbol{X}_i - \boldsymbol{D}\boldsymbol{A}_i\|_F^2 + w_1 \|\boldsymbol{Z}_i\|_{2,1} + w_2 \left\|\tilde{\boldsymbol{\Omega}}_i \boldsymbol{A}_i\right\|_F^2 \quad s.t.\, \boldsymbol{A}_i - \boldsymbol{Z}_i = 0. \tag{3.13}$$

Therefore, the augmented Lagrangian function with respect to $\boldsymbol{A}_i, \boldsymbol{Z}_i$ can be formed as

$$\begin{aligned} L_u(\boldsymbol{A}_i, \boldsymbol{Z}_i) = {} & \|\boldsymbol{X}_i - \boldsymbol{D}\boldsymbol{A}_i\|_F^2 + w_1 \|\boldsymbol{Z}_i\|_{2,1} + w_2 \left\|\tilde{\boldsymbol{\Omega}}_i \boldsymbol{A}_i\right\|_F^2 \\ & - \boldsymbol{\Lambda}_1^T (\boldsymbol{Z}_i - \boldsymbol{A}_i) + \frac{u_1}{2} \|\boldsymbol{Z}_i - \boldsymbol{A}_i\|_2^2, \end{aligned} \tag{3.14}$$

where $\boldsymbol{\Lambda}_1 \in \mathbb{R}^{K \times m}$ are the Lagrangian multipliers for equality constraints and $u_1 > 0$ is a penalty parameter. The Augmented Lagrangian function can be minimised over $\boldsymbol{A}_i, \boldsymbol{Z}_i$ by fixing one variable at a time and updating the other one. The entire procedure is summarised in Algorithm 1. The *Shrink* function in Eq. (3.17) updates $\boldsymbol{Z}_i$ by using row-wise shrinkage, which can be represented as

$$\boldsymbol{z}^r = max\{\|\boldsymbol{q}^r\|_2 - \frac{w_1}{u_1}, 0\} \frac{\boldsymbol{q}^r}{\|\boldsymbol{q}^r\|_2}, r = 1, ..., K, \tag{3.15}$$

where $\boldsymbol{q}^r = \boldsymbol{a}^r + \frac{\boldsymbol{\lambda}_1^r}{u_1}$ and $\boldsymbol{z}^r, \boldsymbol{a}^r, \boldsymbol{\lambda}_1^r$ represent the r^{th} row of matrix $\boldsymbol{Z}_i, \boldsymbol{A}_i, \boldsymbol{\Lambda}_i$ respectively.

Since the ADMM scheme computes the exact solution for each subproblem, its convergence is guaranteed by the existing ADM theory [68][69]. After we obtain the sparse coding,

Algorithm 1: Sparse coding using ADMM

Input: Training Data $\boldsymbol{X} \in \mathbb{R}^{m \times n}$, learned dictionary $\boldsymbol{D} \in \mathbb{R}^{m \times K}$, Number of classes C, regularisation parameters w_1, w_2, penalty parameter u_1 and step length γ_1.

Initialising $\boldsymbol{A}^0 = 0, \boldsymbol{\Lambda}_1^0 = 0$, Iteration number $k = 0$;

for $i = 1 : C$ **do**

while *until converge* **do**

Set the matrix $\tilde{\boldsymbol{\Omega}}_i^k$:

$$\tilde{\boldsymbol{\Omega}}_i^k = diag(\sqrt{\sum_q (\sqrt{\boldsymbol{w}_{i,q}^k} \circ \boldsymbol{a}_{/i,q}^k)^2}) \tag{3.16}$$

Fix $\boldsymbol{A}_i$ and update $\boldsymbol{Z}_i$ by row-wise shrinkage

$$\boldsymbol{Z}_i^{k+1} = Shrink(\boldsymbol{A}_i^k + \frac{1}{u_1}\boldsymbol{\Lambda}_1^k, \frac{1}{u_1}w_1) \tag{3.17}$$

Fix $\boldsymbol{Z}_i$ and update $\boldsymbol{A}_i$ by:

$$\begin{aligned} \boldsymbol{A}_i^{k+1} =& arg\min_{A_i} L_u(\boldsymbol{A}_i, \boldsymbol{Z}_i^{k+1}) \\ =& (\boldsymbol{D}^T\boldsymbol{D} + w_2\tilde{\boldsymbol{\Omega}}_i^{k\boldsymbol{T}}\tilde{\boldsymbol{\Omega}}_i^k + u_1\boldsymbol{I})^{-1}(\boldsymbol{D}^T\boldsymbol{X} + u_1\boldsymbol{Z}_i^{k+1} - \frac{1}{2}\boldsymbol{\Lambda}_1^k) \end{aligned} \tag{3.18}$$

Update Lagrange multipliers $\boldsymbol{\Lambda}_1$:

$$\boldsymbol{\Lambda}_1^{k+1} = \boldsymbol{\Lambda}_1^k - \gamma_1 u_1(\boldsymbol{Z}_i^{k+1} - \boldsymbol{A}_i^{k+1}) \tag{3.19}$$

Increment k.

Output: Estimated sparse code $\boldsymbol{A}$

we secondly update dictionary $\boldsymbol{D}$ column by column with fixed $\boldsymbol{A}$. When updating $\boldsymbol{d}_i$, all the other columns $\boldsymbol{d}_j, j \neq i$ are fixed. Now the objective function in Eq. (3.13) is reduced to

$$\min_{\boldsymbol{D}} \|\boldsymbol{X} - \boldsymbol{D}\boldsymbol{A}\|_F^2, s.t. \|\boldsymbol{d}_i\|_2 = 1. \tag{3.20}$$

In general, we require that each column of the dictionary $\boldsymbol{d}_i$ is a unit vector. Eq. (3.20) is a quadratic programming problem and it can be solved by using the K-SVD algorithm, which updates $\boldsymbol{d}_i$ atom by atom. In practice, the exact solution by K-SVD can be computationally demanding, especially when the number of training images is large. As an alternative, in the following experiments, we use the approximate KSVD to reduce the complexity of this task [1]. The detailed derivation can be found in Algorithm 5 in [145].

Algorithm 2: Overall Framework

Input: Training Data $\boldsymbol{X}$, learned dictionary $\boldsymbol{D}$, Number of classes C, test sample $\boldsymbol{y}$ and regularisation parameters w_1, w_2, w_3.

Initialising $k = 0$;

while *until converge* **do**

  Fix $\boldsymbol{D}^k$ and update $\boldsymbol{A}^{k+1}$ by Algorithm 1;

  Fix $\boldsymbol{A}^{k+1}$ and update $\boldsymbol{D}^{k+1}$ by approximate K-SVD in [145];

  Increment k.

Use $\boldsymbol{A}^{k+1}$ of $\boldsymbol{X}$ to train a linear classifier $\boldsymbol{W}$

Calculate the sparse coefficient $\boldsymbol{a}_{test}$ for $\boldsymbol{y}$ by Eq. (3.22)

Classify the test sample $\boldsymbol{y}$ by $c = arg\max_c \boldsymbol{W}\boldsymbol{a}_{test}$.

Output: Classification result

3.2.3 The Classification Scheme

After obtaining the learned dictionary $\boldsymbol{D}$, a test sample $\boldsymbol{y}$ can be classified based on its sparse coefficients over $\boldsymbol{D}$. We choose a linear classifier both for its simplicity and for the purpose of fair comparison with other dictionary learning methods, although we note that a better classifier design (e.g., SRC) can potentially improve the performance further. We design the linear classifier $\boldsymbol{W} \in \mathbb{R}^{C\times K}$ as [92][155]:

$$\boldsymbol{W}^T = (\boldsymbol{A}\boldsymbol{A}^T + \eta \boldsymbol{I})^{-1}\boldsymbol{A}\boldsymbol{H}^T, \tag{3.21}$$

where $\boldsymbol{A} \in \mathbb{R}^{K\times n}$ is the final rounded coefficients of the training set. The matrix $\boldsymbol{H} \in \mathbb{R}^{C\times n}$ contains the label information of the training set. If the training data $\boldsymbol{x}_i$ belongs to the class c, the element $H_{c,i}$ in vector $\boldsymbol{h}_i$ is one and all the other elements in the same columns are zero. The parameter η controls the tradeoff between the classification accuracy and the smoothness of the classifier.

Next, we can compute the sparse coefficients of the each test sample $\boldsymbol{y}$ using the following objective function:

$$\min_{\boldsymbol{a}} \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{a}\|_F^2 + w_3 \|\boldsymbol{a}\|_1, \tag{3.22}$$

where w_3 is a constant. Finally we apply the linear classifier $\boldsymbol{W}$ to the sparse coding of a test sample to get the label vector $\boldsymbol{h}_y$ and assigned it to the class $c = \arg\max_c \boldsymbol{h}_y$. The overall procedure is summarised in Algorithm 2.

3.3 Experiments and Results

In this section, we compare our proposed Support discrimination dictionary learning (SDDL) method with some other existing Dictionary learning (DL) based classification approaches, that also leverage sparse priors in feature learning. We implemented our approach using Matlab, and the relevant Code is available at https://github.com/BeCarefulPlease/Ch3. We verify the classification performance on various datasets, such as face, object and scene recognition. The classification performance is measured by the percentage of correctly classified test data. The public datasets used are the Extended-Yale B Face Dataset [64], the AR Face Dataset [123], the Caltech 101 object dataset [59] and the 15 Scene Categories dataset [103]. We selected these datasets because they are the benchmarks used in the literature for evaluating the performance of dictionary learning approaches for image classification. The benchmark algorithms used for comparison are the Sparse Representation based Classification (SRC) [175], K-SVD [2], Label-Consistent K-SVD (LC-KSVD) [92], Fisher Discrimination Dictionary Learning (FDDL) [184], Support Vector Guided Dictionary Learning (SVGDL) [22] and Group-structured Dirty Dictionary Learning method (GDDL) [155]. For all the competing methods, we tune their parameters for the best performance.

3.3.1 Parameter Selection

Dictionary size: In all experiments, the initial dictionary is randomly selected from the training data. As shown in [92][184], the larger the size of the dictionary, the better is the performance it can achieve. The disadvantage of a large dictionary is that the problem size becomes large, which is computationally demanding. Therefore, the ideal dictionary learning method should achieve an acceptable level of performance using a relatively small size of dictionary. Here we use the Caltech 101 object dataset as an example. For each class, we randomly choose 30 images for training and the rest for testing. The number of dictionary atoms per class varies from 10 to 30. As shown in Fig. 3.2, all the DL methods tested improve performance when the dictionary size becomes larger. Also, our proposed SDDL method achieves high classification accuracy and consistently outperforms all the other DL-based methods. The basic reason for good recognition performance, even with only a small size dictionary, is that SDDL learns a shared dictionary for all classes, while it can automatically identify sub-dictionaries for different classes, where the size of each sub-dictionary is adjusted appropriately during the learning process.

Regularisation parameters: For SDDL, there are 3 regularisation parameters w_1, w_2, w_3 that need to be tuned, two in the dictionary learning stage and one in the classifier. In

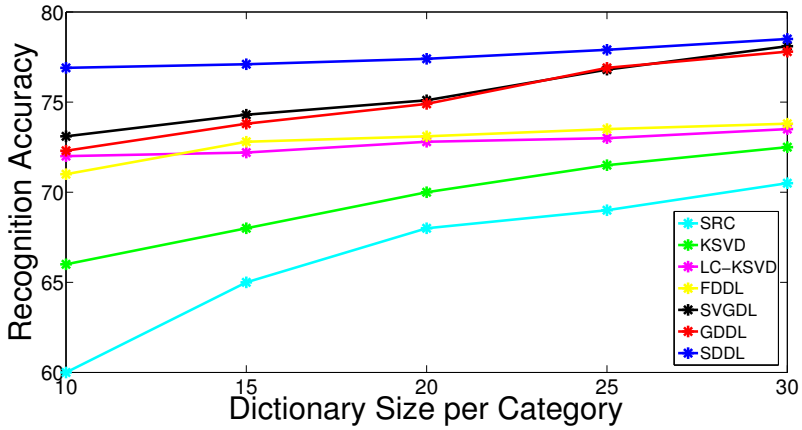


Fig. 3.2 Effect of dictionary size on the classification performance of various DL methods. For the Caltech 101 dataset, the size of training samples per class is fixed to 30. The dictionary atoms per class is varied from 10 to 30. As can be seen, our proposed method outperforms the other DL-based methods.

this paper, we employ cross validation to find the regularisation parameters that give the best result.

Stopping criterion: The proposed algorithm will stop either if the values of the objective function in Eq. (3.11) in adjacent iterations are sufficiently close in value, or if the maximum number of iterations is reached. In Fig. 3.3 we show empirically the value of the objective function as the number of iterations increases using the AR dataset, where we can see that the SDDL method converges rapidly.

3.3.2 Factors Analysis

We will now investigate how the performance is affected by different factors in the proposed method using the face datasets, i.e., the Extended Yale B dataset and the AR dataset. We will discuss two factors as follows:

Factor 1: Function of the ℓ_2/ℓ_1 regularisation term

As mentioned in section 3.2.1, the ℓ_2/ℓ_1 regularisation term is adopted to make the coefficients from the same class share a similar sparse structure. In this section, we provide a visual illustration to see if the ℓ_2/ℓ_1 regularisation term can be truly helpful in the representation of the images from the same class. We compare the sparse codings of the same test samples from two dictionaries, where one is learned with ℓ_1 regularisation while the other with ℓ_2/ℓ_1 regularisation. Fig. 3.4(a) shows 4 test samples of the 2^{nd} subject in the Extended Yale B database; Fig. 3.4(b) and Fig. 3.4(c) show the four coefficients corresponding with the two dictionaries respectively. Looking at the coefficients in Fig. 3.4(b), in which the dictionary is

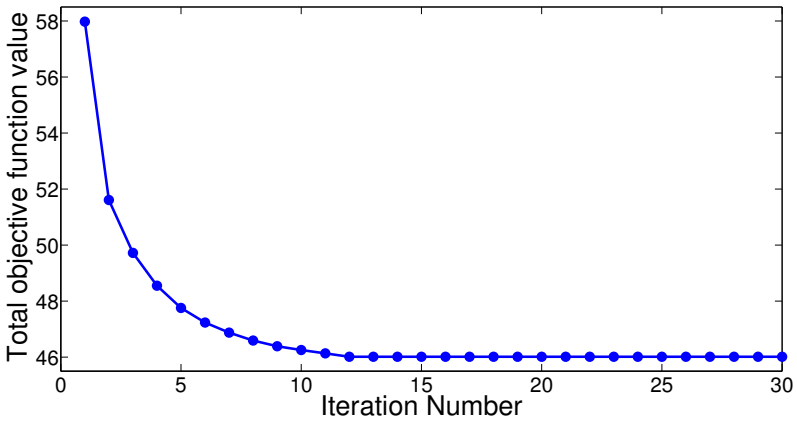


Fig. 3.3 The convergence curve of objective function on the AR database.

learned with ℓ_1 regularisation, it can be seen that the coding vectors corresponding to the fourth image are significantly different to the other three coding vectors of the same class, which is not discriminative, owing to the poor quality of the image. However, in the Fig. 3.4(c), the coding vector of the fourth image now look more similar to the other coding vectors in the class, which has a high probability of being classified correctly. A benefit of such a multi-task learning framework is that 'good quality' images help constrain the coding vector of 'poor quality' ones in the training stage. In this way, even the 'poor quality' images contribute appropriately to the dictionary update.

Factor 2: Function of the discriminative term $f(\boldsymbol{A})$

As described in section 3.2.1, the term $f(\boldsymbol{A})$ is utilised in the objective function to guarantee the discrimination of coding vectors from different classes. In this section, we illustrate both visually and numerically the influence of the discriminative term $f(\boldsymbol{A})$ with an example from the AR database, as shown in Fig. 3.5 and Fig. 3.6.

To clearly show the discrimination of coding vectors between subjects in the AR database (100 subjects in total), we calculate a symmetric scatter matrix $\boldsymbol{S} \in \mathbb{R}^{100\times 100}$, in which each element S_{ij} represents the similarity between sparse codings $\boldsymbol{A}_i$, $\boldsymbol{A}_j$ of i^{th} and j^{th} subject $(i, j \in [1, 100])$:

$$S_{ij} = \sum_{p}\sum_{q} \left\| \boldsymbol{a}_{i,p} \circ \boldsymbol{a}_{j,q} \right\|_1 , \tag{3.23}$$

where $\boldsymbol{a}_{i,p}$ and $\boldsymbol{a}_{j,q}$ are the p^{th} column of $\boldsymbol{A}_i$ and the q^{th} column of $\boldsymbol{A}_j$ respectively. Following this, two scatter matrices are calculated based on the sparse codings of the same test samples from two dictionaries, where one is learned using the discriminative term while the other is not. Then for both scatter matrices, we normalise the largest element of each column or

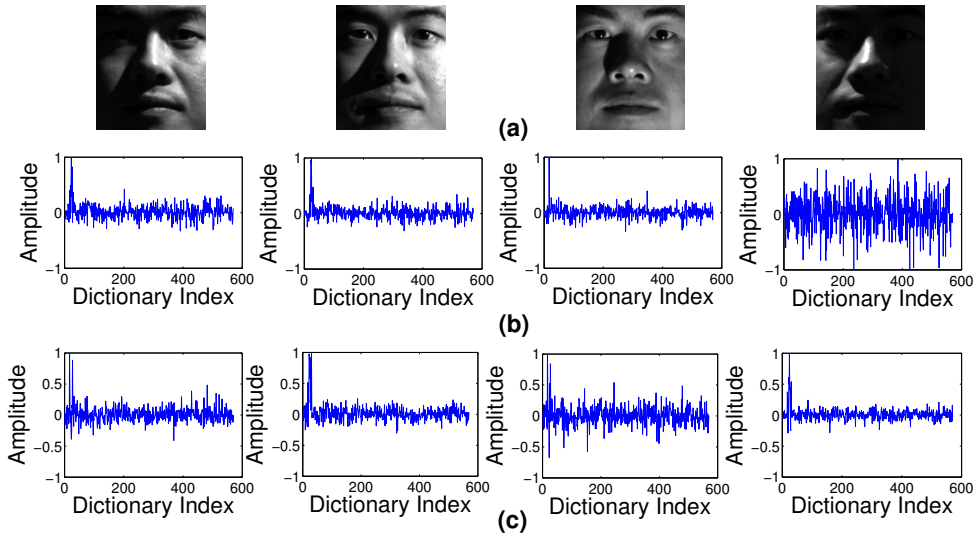


Fig. 3.4 An example for 4 test images and their corresponding coefficients. (a) shows 4 training samples of the 2^{nd} subject in Extended Yale B database; (b) and (c) show the four coefficients corresponding with two dictionaries, where one is learned with ℓ_1 regularisation while the other with ℓ_2/ℓ_1 regularisation respectively.

row to unity to permit comparison and plot them in Fig. 3.5. Accordingly, the diagonal elements represent the similarity of intra-class sparse codings while the off-diagonal elements shows the similarity of the between-subject ones. We see that, the diagonal elements of both figures are the largest, and that there is obviously more between-subject similarity in Fig. 3.5(a) than in Fig. 3.5(b). By summing the elements in the columns of the scatter matrix to quantify the similarity index for each subject, we then plot them in Fig. 3.6. The lower the similarity index, the less overlap there is between the pairs of coefficients between this subject and the others, i.e., the better is the discrimination of the coding coefficient. As shown in Fig. 3.6, the red curve learned using the discrimination term is lower than the blue one learned without the discrimination term for all the 100 subjects, which shows that learning the dictionary with the help with $f(\boldsymbol{A})$ can decrease the coefficient overlap between different subjects. These visual and numerical results both show that the dictionary learned with the $f(\boldsymbol{A})$ term can significantly enhance the discrimination performance of the coefficients. We use the Extended Yale and AR face databases to illustrate how this term can help to improve classification performance. With the help with the discrimination term $f(\boldsymbol{A})$, the recognition rate for the Extended Yale B is enhanced from 96.20% to 98.50%, and the recognition rate for the AR database is increased from 95.90% to 98.00%. The experimental setting used to obtain these result will be presented fully in section 3.3.4.

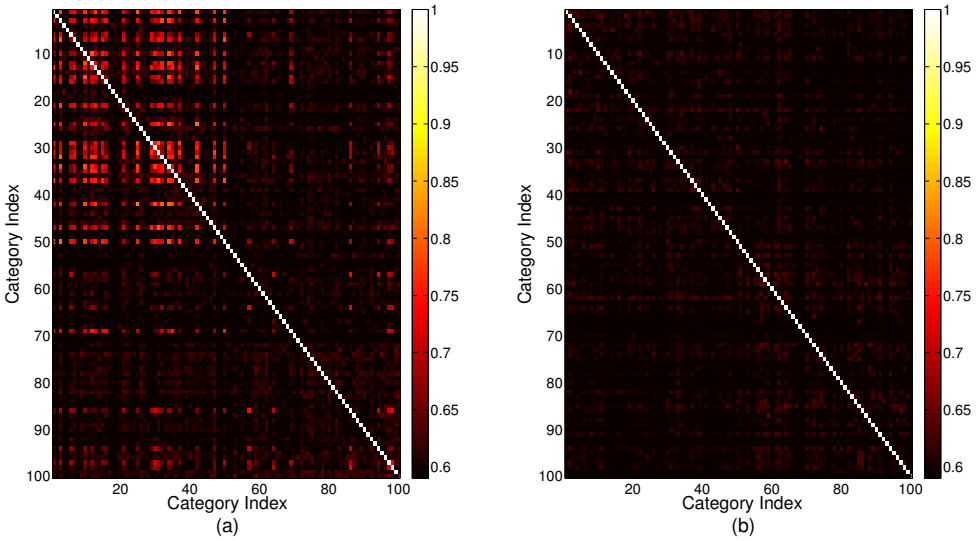


Fig. 3.5 Comparison between the scatter matrices calculated based on the sparse coding of the same test samples from two different dictionaries. In (a), the dictionary is learned without the discrimination term, and in (b), the dictionary is learned using the discrimination term.

3.3.3 Evaluation on Object Recognition Dataset

The Caltech 101 dataset is one of the benchmark datasets used in object classification research. Some examples of the dataset are shown in Fig. 3.7. It consists of 9144 images, split between 101 distinct object classes including animals, vehicles, as well as a background class. The sample from each class has significant shape variability. In the following experiments, the spatial pyramid features are used as the input for the classifier, which is the same as used in [22][92][184]. Following [92], We vary the number of training samples per class from 10 to 30. The size of the dictionary in SDDL is K=510, that is the same as the experimental setting in [22]. The experiments are carried out 10 times with differently chosen partitions. The average classification accuracy of the proposed method (SDDL) compared with other existing dictionary learning based methods is shown in Table 3.1. The regularisation parameters for the Caltech 101 dataset are $w_1 = 0.2, w_2 = 10, w_3 = 0.05$. The DL-based methods perform better than SRC, which shows that better performance can be achieved by learning a discriminative dictionary. Our proposed method consistently outperforms the other existing DL based methods, by at least 2.8 percentage points.

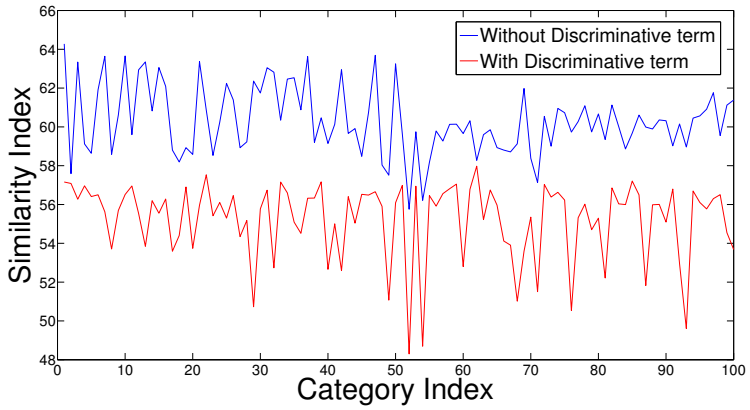


Fig. 3.6 The comparison between the similarity index calculated based on the sparse coding of the same test samples from two different dictionaries.The red line represents the similarity index calculated by the dictionary learned using the discrimination term, while the blue line represents the similarity index without.

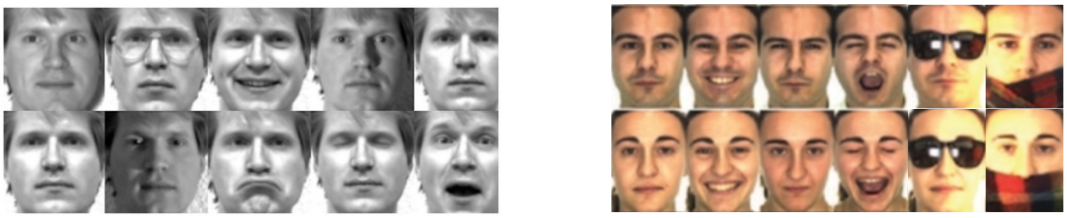
Fig. 3.7 Some sample objects from the Caltech-101 dataset.

Table 3.1 Recognition Rates (%) for Object Classification

No.Training	SRC	KSVD	LC-KSVD	FDDL	SVGDL	GDDL	SDDL
10	58.89	59.80	62.40	63.10	63.10	62.30	**66.80**
15	63.80	64.20	66.90	66.60	68.80	66.20	**71.60**
20	67.20	68.70	69.50	69.80	70.00	69.80	**73.60**
25	68.60	70.20	71.80	72.30	73.50	72.30	**76.50**
30	70.30	73.40	73.30	73.10	74.10	73.40	**76.90**

3.3.4 Evaluation on Face Recognition Dataset

The two benchmark face datasets employed are the Extended Yale B dataset and the AR dataset. Some sample images of Extended Yale B and AR dataset are shown in Fig. 3.8.



(a) Extended Yale B (b) AR face

Fig. 3.8 Some sample images in Extended Yale B and AR face dataset

With different illumination conditions and facial expressions, the Extended Yale B dataset consists of 2414 frontal images of 38 subjects (about 64 images per subject). We randomly select half as the training set and the rest as the test set for each class. As in the experimental setting in [92] [155], we crop each image to 192×168 pixels, and then normalise and project it to a 504 dimension vector using a random Gaussian matrix. The dictionary size of the Extended Yale B dataset is 570, which corresponds to an average of 15 atoms per subject. As discussed previously, there is no explicit correspondence between the dictionary atoms and the labels of the individual at the training stage.

Similarly, the AR face dataset consists of over 4000 images of 126 subjects, which is more challenging owing to the greater degree of variation, i.e., different illumination, expressions and facial occlusion (e.g., sunglasses, scarf). As in the experimental setting in [92] [155], we use the subset of the dataset which contains 2600 images for 50 male and 50 female subjects. For each subject, we randomly select 20 and 6 images for training and testing respectively. We crop each image to 165×120 pixels, and then normalise and project it to a 540 dimension vector using a Gaussian matrix. The dictionary size of the AR dataset is 500, that corresponds to an average of 5 atoms per subject. The dictionary is shared by all subjects.

The experiments are carried out 10 times with different chosen partitions. The average classification accuracy of the proposed method compared with other existing dictionary learning based methods are shown in the Table 3.2. The regularisation parameters for the Extended Yale B dataset are $w_1 = 0.04, w_2 = 2, w_3 = 0.005$, and for the AR face database are $w_1 = 0.05, w_2 = 3, w_3 = 0.005$. We can see that the proposed SDDL method achieves an improvement of at least 1.7 and 2 percentage points over the next best scheme in terms of classification accuracy for the Extended Yale B and the AR datasets respectively.

Table 3.2 Recognition Rates (%) for Face Classification

Method	SRC	KSVD	LC-KSVD	FDDL	SVGDL	GDDL	SDDL
Extended Yale	80.54	93.40	94.50	94.92	95.70	96.80	**98.50**
AR	66.57	86.30	93.70	94.10	96.00	96.00	**98.00**

Fig. 3.9 Some sample images from the 15 Scene Categories dataset.

3.3.5 Evaluation on Scene Recognition Dataset

The 15 Scene Categories dataset is one of the benchmark datasets used in scene classification. As shown in Fig. 3.9, this dataset contains a wide range of outdoor and indoor scenes, split between 15 distinct classes, e.g., bedroom, kitchen, street and coast. The images within a given class have significant visual variability. Each category contains about 200 to 400 images, in which the size of each image is around 250×300 pixels. Following [92], the spatial pyramid features are used as the input for the classifier. We randomly select 100 samples per category as the training set and the rest as the test set. The dictionary size of the 15 Scene dataset is $K = 450$, that corresponds to an average of 30 atoms per class. The dictionary is shared by all classes. The regularisation parameters for the 15 Scene dataset are $w_1 = 0.01, w_2 = 1, w_3 = 0.02$. The experiments are carried out 10 times with different chosen partitions. The average classification accuracy of the proposed method (SDDL) compared with other existing sparse based methods is shown in Table 3.3. Again, it is observed that our proposed approach achieves the best classification performance in comparison to other state-of-the-art approaches.

Table 3.3 Recognition Rates (%) for Scene Classification

Method	SRC	KSVD	LC-KSVD	FDDL	SVGDL	GDDL	SDDL
15 Scene	91.80	86.70	91.90	91.60	92.90	93.10	**93.70**

3.4 Chapter Summary

In this Chapter, we incorporate a structured sparse prior into the dictionary learning process and propose a support discrimination dictionary learning (SDDL) method for discriminative feature learning. In contrast to other methods, we use the sparse structure, i.e., support, to measure the similarity between the pairs of coefficients, rather than the Euclidean distance which is widely adopted in many dictionary learning approaches for classification. The discrimination capability of the proposed method is enhanced in two ways. First, a row sparse regulariser is adopted so that a shared support structure for each class can be learned automatically. Second, we adopt a discriminative term to make the coefficients from different classes have minimum support overlap between each other. It can be achieved by minimisation of the ℓ_0 norm of the Hadamard product between any pair of coefficients in different classes. It worth noting that our approach can automatically identify overlapped sub-dictionaries for different classes, where the size of each sub-dictionary is adjusted appropriately during the learning process to suit the training dataset. In this way, this proposed approach is scalable to classification tasks having a large number of classes. Extensive experimental results on object, face and scene recognition demonstrate the proposed method can generate more discriminative sparse coefficients and that it has superior classification performance to a number of state-of-the-art dictionary learning based methods.

Chapter 4

Dictionary Learning Inspired Deep Network for Scene Recognition

Taking inspiration from biology, deep neural networks have in recent years come to dominate the previously separate fields of research in machine learning, computer vision, natural language understanding and speech recognition as discussed previously in Chapter 2. A deep neural network incorporates hierarchical priors on feature representation learning, i.e., it constructs multiple levels of representations that correspond to different levels of abstraction or equivalently learning a hierarchy of features. More specifically, a deep neural network will first learn low level features for lines, dots, curves etc, and then compose them into high level features, which describes the common objects and their shapes. There are two aspects of the benefits of using hierarchical priors to the feature learning model, as discussed below:

Firstly, using a hierarchical prior promotes the reuse of features. The notion of reuse, which explains the power of distributed representations The depth, which is the length of the longest path from the input to output node of a deep neural network, is a key aspect that determines the representation power. The number of paths in the network, i.e., ways to reuse different parts, can grow exponentially with its depth. Some theoretical results demonstrate families of functions where a deep representation can be exponentially more efficient than one that is insufficiently deep [10] [11] [13] [82] [83].

Secondly, using a hierarchical prior can lead to abstract representation at higher layers of representation, which are generally more invariant to most local changes of the input. As composition is useful for humans to describe the world around us efficiently, a deep neural network can first learn simple concepts (low level) and then compose them into more complex ones (high level). For example, in the basic Convolutional neural network (CNN), abstraction can be obtained by pooling or by a strided convolution mechanism. Generally, more abstract concepts are generally highly non-linear functions of the raw input, which is

invariant to most local changes of the input. In basic CNN, we can use the fully connected layer to combine low level features achieved by the early convolutional and pooling layers and obtain a semantic categorical concept, which potentially has greater predictive power.

In this chapter, we not only incorporate hierarchical priors on the feature representation learning via the use of deep neural networks, but also integrates sparse priors into a unified framework. We propose an approach to automatically adjust the sparsity level of the intermediate hidden units based on the training set, as shown in Fig. 4.1. Besides the advantages of hierarchical and sparse prior as discussed before, this specific design brings three significant advantages: (1) It makes each neuron in the deep network have its maximum discrimination ability; (2) It makes images belonging to different classes maximally distinguishable; (3) It provides potential to automatically select the most reusable low level features (especially in the feature transfer task). More specifically, in this chapter, under the framework of deep neural networks, we replace the conventional FCL and ReLu with a new dictionary learning layer (DLL). We note that the DLL is composed of a finite number of recurrent units to simultaneously enhance the both sparse representation and discriminative abilities of the features via the determination of optimal dictionaries. In this chapter, we use the challenging scene recognition task to examine the performance of our proposed method, as unlike the situation in generic object categorisation, there are many discriminative part regions in scene images which is prone to produce complex distributions of the intermediate feature representation. To address this complex distribution, our DLL contributes specifically to the scene recognition task in that the appropriate sparsity level can be learned in spite of the unknown and complex sparse prior of the intermediate feature representations.

The rest of this chapter is organized as follows: Section 4.1 provides the most relevant prior works about combining sparse priors with a deep neural network (hierarchical prior) and demonstrate the primary motivation and contributions of this chapter. Section 4.2 presents the details of the novel DLL, that is composed of a finite number of recurrent units to simultaneously incorporate the sparse prior into the feature learning procedure. With the help of the structure of the dictionary, we also proposed a new label discriminative regressor and the use of overfitting prevention modules to boost the discrimination and generalization ability of the learned feature representation. In section 4.3, we present an ablation study of our proposed approach and it shows superior performance to many state-of-the-art approaches in various challenging scene datasets; Section 4.4 presents the conclusions.

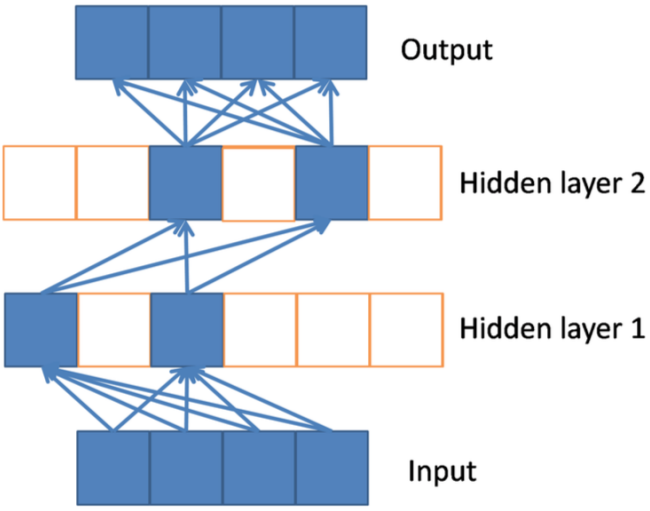


Fig. 4.1 Incorporate sparse and hierarchical priors into feature learning.

4.1 Introduction

Scene recognition remains one of the most challenging problems in image understanding due to illumination changes, high intra-class variance, and background occlusion. To tackle these issues, convolutional neural networks (CNNs) [99] [150] trained on the large scale Places dataset (Places-CNNs) [197] have yielded improved performance for the scene recognition task. The power of CNNs is achieved by learning a strong feature representation in an end-to-end manner for the classification task, instead of hand-crafting features with heuristic parameter tuning. The CNN can be considered as a universal feature extractor, which learns a new representation of the data that permits computationally easier and more effective classifier design. Many deep CNN architectures have been proposed, but fully connected layers (FCL) followed by rectified linear units (ReLu) are now prevalently used to combine the local features extracted from the early convolution and pooling layer. [154] observed that the nonlinear property inherent in the ReLu unit leads to sparsity in the neural activations, consequently improving the performance of deep learning methods. Normally, for each training image, around half of the neurons in the hidden layer are activated. They argued that moderate sparse on neuron activation not only makes each neuron have its maximum discrimination ability, but also makes images of different classes maximally distinguishable. However, there is no effective mechanism to automatically adjust the sparsity level of the intermediate hidden units based on the training set.

It would be beneficial to take advantage of the sparse model to adjust the sparsity level and enforce different categories to have different subsets of neurons activated, consequently

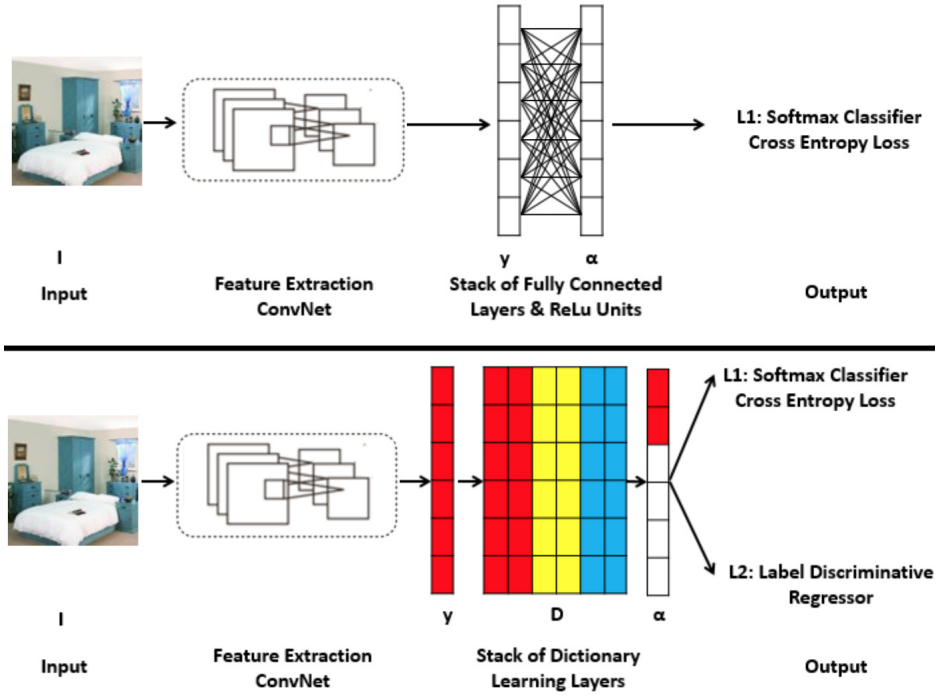


Fig. 4.2 Comparison between the conventional CNN architecture and our CNN-DL architecture.

maximizing discriminative power. As an extension of the standard reconstructive dictionary [2], known as discriminative dictionary learning (DL) methods [92][185] aim to find the optimal dictionary that simultaneously improves the sparse representation and also maximize its discriminative capability. However, the current DL methods cannot achieve state of the art performance in large-scale image classification, especially in scene recognition, in part since most DL models have only been evaluated with traditional handcrafted features, e.g., BOF [180] and SIFT [117].

One way to utilize DL to improve scene recognition performance produced by CNN features is to apply DL as a post-processing step trained separately from the training of the CNN, an approach that is adopted in [108][178]. Arguably, this does not fully harness the strength of DL since it is not integrated with the deep network. It is also worth noting that although the work in [168] addresses object detection, it does combine the DL and conventional CNN layers into the end-to-end training framework by simply replacing the softmax classifier by a DL classifier. However, the DL classifier has not been used to replace any fully connected layers before the classifier. In addition, without an explicit parameter for sparsity control of the sparse coding step in the DL classifier, it is difficult to know whether the sparsity of the total network is best optimized and further, to say that the sparseness will

be a good regularization for training the low-level layers in the entire network. Considering the optimization loss function, it is well known that the cross entropy is more robust to outliers than is Mean-Squared Error (MSE) estimation. However, due to the design of their DL classifier, they have to use the MSE of the reconstruction error, rather than the more robust cross-entropy loss in the conventional classification task. In addition, [169] also combined CNN and sparse coding for image super-resolution recovery. Although they cascaded the Learned Iterative Shrinkage-Threshold Algorithm (LISTA) [76] network to control the sparsity via an explicit shrinkage function, their loss function is still the MSE loss of the reconstruction error, without any effort to improve feature discriminative capability. To the best of our knowledge, there is no end-to-end learning framework to combine CNN and DL in scene recognition, that can automatically adjust the sparsity level and discriminative capability of the feature representation.

In this proposed approach, our formulation combines the strength of both conventional CNNs and DL procedures in a unified framework, namely CNN-DL. The overall framework for CNN-DL is illustrated in the lower diagram in Fig. 4.2, where arrows indicate the forward propagation direction. In Fig. 4.2, the upper image is the conventional CNN architecture; the lower image is the CNN-DL architecture. We replace the FCL and ReLu units by the DLL, in order to achieve enhanced sparse and discriminative feature representation. The CNN-DL network has two sibling output layers. The first outputs a discrete probability distribution over different categories, computed by a softmax classifier, while the second sibling layer outputs the corresponding discriminant sparse representation that depends upon the dictionary structure.

Our contributions are summarized as following. Firstly, inspired by the Approximate Message Passing (AMP) [50] algorithm and the LISTA network, we design a non-linear DLL composed of a finite number of recurrent units to integrate the sparse coding and dictionary learning into the deep network architecture. Such a layer can solve the sparse coding with a better convergence rate than LISTA, while also contributing the gradient flow to update the dictionary in each recurrent iteration. In addition, we decoupled the shrinkage function threshold into two factors, so that the threshold is optimally learned to the unknown sparse prior of the neural activation maps. Secondly, we build a new network architecture named as CNN-DL, which replaces the conventional FCLs and ReLu units with a DLL, that exhibits an enhanced sparse and discriminative feature representation. In addition, we take advantage of the structure of the dictionary and design a new label discriminative regressor to further improve the discriminative capability. We argue that such a layer can be used to replace any fully connected layer (FCL) and can be cascaded as in the conventional deep learning framework. Thirdly, we propose new constraints to prevent overfitting, i.e., by taking

advantage of both the Mahalanobis and Euclidean distances measures, thereby achieving a balance between recognition accuracy for the training set and the generalization performance. Fourthly, we suggest an optimization algorithm for the end-to-end training to jointly learn the parameters in the CNN and DL, that is also compatible with the conventional back propagation scheme. Fifthly, we analyze various factors in CNN-DL that affect the scene recognition performance, including different loss functions and DLL settings. Finally, the CNN-DL model is extensively evaluated using various scene datasets, and it shows superior performance to many state-of-the-art scene recognition algorithms.

4.2 Integration of Dictionary Learning and CNN

4.2.1 Network Architecture

Similarly to most scene recognition methods based on deep learning networks, our network architecture is designed to learn the transformation from the original image $\boldsymbol{I}_i$ to its corresponding label vector $\boldsymbol{h}_i$ in an end-to-end manner, where i indicates the index of the training samples. The overall network structure is illustrated in the lower image in Fig.4.2, where arrows indicate the forward propagation direction. The details of each layer are discussed as follows.

To extract the local features of the input image $\boldsymbol{I}_i$, it first goes through the ConvNet, which contains a stack of conventional convolution and pooling layers. To measure the improvements brought by our proposed DLL and the novel loss function in a fair setting, the configurations of all our ConvNet layers are designed based on the same principles and parameter settings used in [99] and [150]. We denote the features obtained from the ConvNet as $\boldsymbol{y}_i = f(\boldsymbol{I}_i) \in \mathbb{R}^M$, where f represents the transformation in the convolution and pooling layers.

Further to obtain the combination of local features, instead of forwarding the features $\boldsymbol{y}_i$ into FCLs (followed by ReLu) as shown in the upper figure of Fig.4.2, we design the non-linear DLLs so as to obtain a sparse feature representation. Each DLL is parameterized with a dictionary $\boldsymbol{D} \in \mathbb{R}^{M \times N}$, composed of a finite number of recurrent units to mimic the sparse coding procedure. Specifically, the dictionary will be ideally learned if the following equation is satisfied for any given extracted feature representation $\boldsymbol{y}_i$:

$$\boldsymbol{\alpha}_i = arg\min_{\boldsymbol{\alpha}_i} \|\boldsymbol{y}_i - \boldsymbol{D}\boldsymbol{\alpha}_i\|_F^2 + \lambda \|\boldsymbol{\alpha}_i\|_1, \tag{4.1}$$

where $\boldsymbol{\alpha}_i$ is the corresponding sparse feature representation.

In the proposed network, we replace the conventional FCL by the DLL, which is a sub-network aimed at enforcing the sparsity prior on the representation. Conventionally, as in [99], a generic ReLu is used for nonlinear mapping. Since the DLL is designed based on our domain knowledge from sparse coding, we are able to adjust the sparsity level of the neuron activation based on the training set, consequently, obtaining a better interpretation of the layer response. The implementation of this layer should be such that it is capable of passing the error differentials from its outputs to inputs during back-propagation to update the dictionary. The detailed description of the DLL and its relevant optimization rules are discussed in the section 4.2.3, i.e., the non-linear DLL section.

The design of DLL is not only aimed at a good representation of the given features, its structural flexibility also has the potential to enhance the discriminative capability of the network. This can be achieved by exploiting the dictionary structure and the label information of the input images as part of the final loss function. Inspired by the label consistent K-SVD method [92], we integrate a label discriminative regressor $\mathcal{L}_2 = \sum_i \lambda_1 \|\boldsymbol{q}_i - \boldsymbol{B}\boldsymbol{\alpha}_i\|_F^2 + \lambda_2 \left\|\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{I}\right\|_F^2$ into the final loss objective to measure the discriminative capability of the sparse codes, which forces the training samples from the same class to have similar sparse codes, while forcing samples from different classes to have different sparse codes. The matrix $\boldsymbol{B} \in \mathbb{R}^{N\times N}$ is the optimal linear mapping matrix to transform the original sparse code $\boldsymbol{\alpha}_i$ to the most discriminate sparse feature domain. The ideal discriminative sparse coefficient corresponding to a training sample $\boldsymbol{y}_i$ is denoted $\boldsymbol{q}_i$. The nonzero values in $\boldsymbol{q}_i$ only occur at the places where the input training sample $\boldsymbol{y}_i$ and the dictionary atom $\boldsymbol{d}_n$ share the same label. Thus we define the discriminant sparse code $\boldsymbol{q}_i$ as

$$\boldsymbol{q}_i = [q_i^1, q_i^2, ..., q_i^N] = [0...0, 1...1, 0...0]^T \in \mathbb{R}^N. \tag{4.2}$$

To prevent over-fitting during training, we also add a constraint term on matrix $\boldsymbol{B}$ to balance the Euclidean and Mahalanobis distances as $\|\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{I}\|_F^2$. The details of the label discriminative regressor and its implementation is introduced in the section 4.2.2.

Besides utilizing the loss function $\mathcal{L}_2$ to maintain the discriminative sparse code, $\boldsymbol{\alpha}_i$ can also be considered directly as a discriminant feature of original image $\boldsymbol{I}_i$ for classification. Here, a conventional softmax classifier is used to compute the predicted label, and the cross-entropy loss function $\mathcal{L}_1$ is used to compare between the predicted label $\boldsymbol{h}_i^*$ and ground truth label $\boldsymbol{h}_i$. Finally, the overall loss function of our network can be expressed as

$$\min_{\Theta} \mathcal{L}_1 + \lambda_1 \sum_i \|\boldsymbol{q}_i - \boldsymbol{B}\boldsymbol{\alpha}_i\|_F^2 + \lambda_2 \left\|\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{I}\right\|_F^2, \tag{4.3}$$

where Θ represents the parameters within the network, and λ_1 and λ_2 balance the contribution of different terms. We will see in the experiments that the design of the proposed DLL and the novel objective function both contribute to better scene recognition results and a smaller model size than a conventional CNN.

4.2.2 Loss Function Design with Integration of the Label Discriminative Regressor

As part of the final loss function in (4.3), the objective function to improve discrimination can be expressed as

$$\min_{\Theta} \lambda_1 \sum_i \|\boldsymbol{q}_i - \boldsymbol{B}\boldsymbol{\alpha}_i\|_F^2 + \lambda_2 \left\|\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{I}\right\|_F^2 . \tag{4.4}$$

It is worth noting that in the (4.4), we adopted a new constraint regularized by λ_2 to prevent overfitting. It is designed to incorporate the strength of the Mahalanobis and Euclidean distances, and to achieve a balance between the recognition accuracy and the generalization ability. To understand this constraint, we argue that the term $\|\boldsymbol{q}_i - \boldsymbol{B}\boldsymbol{\alpha}_i\|_F^2$ can be regarded as measuring the Mahalanobis distance [1] between the optimal and calculated sparse codes $\boldsymbol{\alpha}_i^*$ and $\boldsymbol{\alpha}_i$ under the transformation matrix $\boldsymbol{B}$, as:

$$\begin{aligned} \|\boldsymbol{q}_i - \boldsymbol{B}\boldsymbol{\alpha}_i\|_F^2 &= \|\boldsymbol{B}\boldsymbol{\alpha}_i^* - \boldsymbol{B}\boldsymbol{\alpha}_i\|_F^2 \\ &= (\boldsymbol{\alpha}_i^* - \boldsymbol{\alpha}_i)^T \boldsymbol{B}^T\boldsymbol{B}(\boldsymbol{\alpha}_i^* - \boldsymbol{\alpha}_i) . \end{aligned} \tag{4.5}$$

Although the Mahalanobis distance is well known for its discriminative capability, it is prone to be overfitting. Compared with the Mahalanobis distance, the Euclidean distance has a worse ability to discriminate but a better ability to generalize, because it does not take into consideration data correlation across dimensions [121]. Here, we impose a constraint to ensure that the matrix $\boldsymbol{B}^T\boldsymbol{B}$ has large values on the diagonal and small values elsewhere, so that we can obtain a compromise between the Mahalanobis distance and the Euclidean distance. The constraint is formulated as the Frobenius norm of the difference between $\boldsymbol{B}^T\boldsymbol{B}$ and the identity matrix $\boldsymbol{I}$. More specifically, when λ_2 is small, the Mahalanobis distance should be dominant to measure the distance between $\boldsymbol{\alpha}_i^*$ and $\boldsymbol{\alpha}_i$. In contrast, with the choice of a larger value of λ_2, the matrix $\boldsymbol{B}^T\boldsymbol{B}$ is forced to be close to the identity matrix so that the Euclidean distance is more dominant. In this situation, the Euclidean distance may generalize better to unseen test sets. To sum up, we incorporate the advantage of Mahalanobis and Euclidean distances to maintain the discriminative capability while reducing the overfitting problem.

[1] The Mahalanobis distance is formulated as $d(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2) = \sqrt{(\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2)^T \boldsymbol{M}(\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2)}$

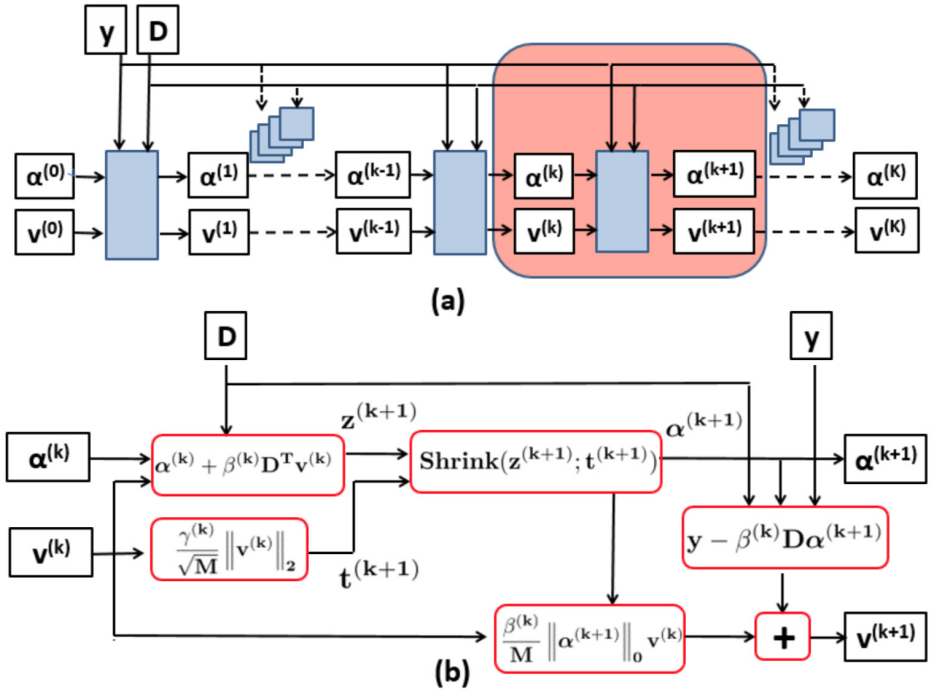


Fig. 4.3 The design of a non-linear dictionary learning layer. Each blue box in the Fig.4.3(a) represents a recurrent unit, which corresponds to one iteration in the AMP sparse coding procedure. The red box in Fig.4.3(a) shows a recurrent unit at the k^{th} iteration, the details of which are given in Fig.4.3(b).

4.2.3 Nonlinear Dictionary Learning Layer

In this section, we introduce details of the fast yet accurate non-linear DLL which is inspired by the Approximate Message Passing (AMP) algorithm [50] and the on-line dictionary learning method in [119]. As shown in Fig.4.3(a), given the input data from previous layers $\boldsymbol{y}$, the DLL is able to compute the final output sparse codes $\boldsymbol{\alpha}^{(K)}$ efficiently using K stacked recurrent units (blue boxes), each of which exhibits a similar function to that of an iteration in the AMP algorithm. As shown in Fig.4.3(b), we represent such adjacent recurrent units in a data flow graph, where the blue box represents the recurrent unit, a node represents a variable and the directed edge represents the flow between two variables. The operations between the adjacent k^{th} and $(k+1)^{th}$ units in the data flow graph can be represented as:

$$\boldsymbol{z}^{(k+1)} = \boldsymbol{\alpha}^{(k)} + \beta^{(k)} \boldsymbol{D}^T \boldsymbol{v}^{(k)} \tag{4.6}$$

$$\boldsymbol{t}^{(k+1)} = \frac{\gamma^{(k)}}{\sqrt{M}} \left\| \boldsymbol{v}^{(k)} \right\|_2 \tag{4.7}$$

$$\boldsymbol{\alpha}^{(k+1)} = max(\left|\boldsymbol{z}^{(k+1)}\right| - \boldsymbol{t}^{(k+1)}, 0)\frac{\boldsymbol{z}^{(k+1)}}{\left|\boldsymbol{z}^{(k+1)}\right|} \tag{4.8}$$

$$\boldsymbol{v}^{(k+1)} = \boldsymbol{y} - \beta^{(k)}\boldsymbol{D}\boldsymbol{\alpha}^{(k+1)} + \frac{\beta^{(k)}}{M}\left\|\boldsymbol{\alpha}^{(k+1)}\right\|_0 \boldsymbol{v}^{(k)}, \tag{4.9}$$

where γ and β represent the learnable scaling factors and $\boldsymbol{t}$ represents the threshold vectors, M represents the dimension of the input feature $\boldsymbol{y}$, the first layer inputs are $\boldsymbol{\alpha}^{(0)} = 0$ and $\boldsymbol{v}^{(0)} = \boldsymbol{y}$. Note that sparse coding steps can be achieved in a relatively small number of iterations to fit existing data sets. We used $K = 2$ recurrent units within each DLL throughout this chapter unless otherwise specified. Since the dictionary is shared among recurrent units within one DLL, the parameters in each DLL can then be represented as $\Theta = \{\boldsymbol{D}, \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{k=K}\}$.

There are three major differences between our DLL and the LISTA algorithm: firstly inspired by the AMP, the conventional residual $\boldsymbol{v}^{(k+1)}$ is corrected by use of the Onsager correction [50] term: $\frac{\beta}{M}\left\|\boldsymbol{\alpha}^{(k+1)}\right\|_0 \boldsymbol{v}^{(k)}$ in our DLL layer. This correction has been proved to enable the input of the shrinkage function, $\boldsymbol{z}^{(k+1)}$ to be treated as an approximately Additive white Gaussian noise (AWGN) corrupted optimal sparse code. Therefore, many shrinkage functions can be applied easily. Secondly, the threshold selection $\boldsymbol{t}^{(k+1)}$ in our shrinkage function design is decoupled into the scaled factor $\gamma^{(k)}$ and the realization $\boldsymbol{v}^{(k)}$. This enables the threshold of the shrinkage function to be optimized when the prior of sparse codes is not easily modeled. We argue that this will be useful for a deep learning framework, as estimating the prior of the neural activation will be a difficult task. More specifically, learning from training samples, $\boldsymbol{\gamma}^{(k)}$ is adaptive to the unknown and complex prior of the neural activation. Thirdly, except for the dictionary matrix, we do not have to learn an additional transformation matrix (matrix $\boldsymbol{B}$ in the LISTA framework [76]), but decouple it into dictionary $\boldsymbol{D}$ and the scaling parameter β. This lowers the model complexity and ensures that the input to the shrinkage function is still essentially AWGN corrupted.

As the overall objective function in (4.3) does not depend on $\boldsymbol{D}$ explicitly, it is difficult to compute the gradient with respect to $\boldsymbol{D}$ in each recurrent unit. Therefore, we consider the dictionary $\boldsymbol{D}$ as a shared parameter implicitly in each recurrent unit and propose to compute the gradient of the loss function $\mathcal{L}$ with respect to $\boldsymbol{D}$ using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{D}} = \sum_{k=1}^{K} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}^{(k)}} \frac{\partial \boldsymbol{\alpha}^{(k)}}{\partial \boldsymbol{z}^{(k)}} \frac{\partial \boldsymbol{z}^{(k)}}{\partial \boldsymbol{D}}, \tag{4.10}$$

where

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}^{(k)}} = \begin{cases} \prod_{k+1}^{K} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}^{(k)}} \frac{\partial \boldsymbol{\alpha}^{(k)}}{\partial \boldsymbol{z}^{(k)}} \frac{\partial \boldsymbol{z}^{(k)}}{\partial \boldsymbol{\alpha}^{(k-1)}} & if\, k < K \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}^{(k)}} & if\, k = K \end{cases} \tag{4.11}$$

It is also worth noting three points: first, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}^{(K)}}$ can be calculated based on the overall objective function (4.3); Second, the q^{th} element of $\frac{\partial \boldsymbol{\alpha}^{(k)}}{\partial \boldsymbol{z}^{(k)}}$ is set to 0 if the q^{th} element $\boldsymbol{\alpha}_q^{(k)} = 0$, otherwise, it is set to 1. Thirdly, $\frac{\partial \boldsymbol{z}^{(k)}}{\partial \boldsymbol{\alpha}^{(k-1)}}$ can be calculated based on equation Eq (4.6) and (4.9) and we use the $\mathcal{L}_1$ norm as an approximation of $\mathcal{L}_0$ norm in Eq (4.9) to enable standard back propagation in our implementation.

Once $\frac{\partial \mathcal{L}}{\partial \boldsymbol{D}}$ is calculated, the dictionary $\boldsymbol{D}$ in the DLL layer can be updated by stochastic gradient descent. To make this new layer compatible with the other layers in the current CNN framework, we need to consider how the loss function can be back-propagated through this DLL layer to the previous layers. We need to calculate $\frac{\partial \mathcal{L}}{\partial \boldsymbol{y}}$, and once it is obtained, we can perform standard back propagation [99] to update the CNN parameters in the previous layers. In fact, $\boldsymbol{y}$ is similar to the case of dictionary $\boldsymbol{D}$, and can also be obtained by use of the chain rule.

From the perspective of the overall network architecture, different DLLs (cascaded in series) are parametrized by separate dictionaries and all the parameters within the overall network can be trained by standard back-propagation. It will be shown in the experiments that with the integration of the newly proposed DLL, the overall CNN-DL structure is able to generate better scene recognition performance than can the conventional CNN structure.

4.3 Experiments and Results

In this section, we evaluate our approach using various scene recognition datasets and compare with other state-of-the-art approaches. We first introduce the datasets and the parameter settings, and second give an analysis of the factors that effect in the CNN-DL method. Then, we report the recognition performance of the proposed method and compare it with other scene recognition algorithms in the following section. The evaluation is performed for various network architectures and scene datasets.

4.3.1 Datasets and Experimental Settings

Since all the selected comparison methods only present their results for the 15 Scene [103], MIT Indoor-67 [138], or Sun 397 [177], we choose to employ them in our scene recognition experiments. More detailed information about the 15 Scene dataset can be found in section 3.3.5 and Fig.3.9. The MIT 67 indoor set is provided to address the challenging indoor scene recognition problem. This database contains 67 Indoor categories, and contains a total of 15620 images. The number of images varies across categories, but there are at least 100 images per category, and the images are in jpg format. Some sample images from MIT

Fig. 4.4 Some sample images from the MIT 67 dataset.

Indoor-67 dataset can be seen in Fig. 4.4. The SUN 397 dataset contains 397 well-sampled categories categories in total. The number of images varies across categories, but there are at least 100 images per category, and there are 108,754 images in total. The images are in jpg, png, or gif format. Some sample images from Sun 397 dataset can be found in Fig. 4.5.

We use the average accuracy to evaluate the recognition performance. The parameters in the objective function (4.3) are determined by 5-fold cross-validation for different datasets as listed in Table 4.1. We follow the same training-test partition used in [108] [178]. 15 Scene includes 100 images per class for training and the rest for testing. MIT Indoor 67 includes 80 images of each category for training and 20 images for testing. SUN 397 includes multiple train/test splits, with 50 images per class in the testing set. We present results for the average accuracy over the splits.

To completely understand and evaluate effects of different factors in CNN-DL, a detailed factor analysis regarding parameter selection and layer settings are performed in the next section. We implement our approach based on Tensorflow and the relevant code is available at https://github.com/BeCarefulPlease/Ch4. For comparisons between other methods, the CNN network architectures we adopt are AlexNet [99] and VGG net [150]. The model is first trained on the large auxiliary dataset with image level supervision, including ImageNet data [99] or Place205 data [197]. To adapt the pre-trained CNN to the new scene dataset, we perform domain-specific fine-tuning on the three scene datasets. In the following experiments, we replace the FCLs by an equivalent number of the proposed non-linear DLLs and fine-tune the pretrained CNN. It worth noting that the number of dictionary atoms per class should be one in the final DLL. Detailed information about the dictionary size of the previous to last DLL layer are shown in Table 4.1. To balance the trade-off between speed and accuracy,

Fig. 4.5 Some sample images from the SUN 397 dataset.

Table 4.1 Dataset and Experimental Details

Dataset	Dictionary Size	λ_1	λ_2
15 Scene	450 atoms (30 atoms/class)	0.2	0.001
MIT Indoor 67	2680 atoms (40 atoms/class)	0.5	0.005
SUN 397	3970 atoms (10 atoms/class)	0.3	0.02

we used $K = 2$ recurrent units within each DLL throughout the evaluation unless otherwise specified.

4.3.2 Factor Analysis

In this section, we investigate how the performance of CNN-DL is affected by different factors using the MIT Indoor 67 dataset. All the analysis in this section is based on results achieved by CNN-DL on AlexNet (pre-trained on ImageNet data). We will discuss two main factors, including the configuration of the DLL and the use of different loss functions.

Factor A: Proposed Dictionary Learning Layer

In this section, we verify the effectiveness of DLL components in different configuration settings, including dictionary size (factor A1), the number of recurrent units (factor A2) and the use of cascaded DLLs (factor A3). In some of the tests, we also compare the performance of DLL with its conventional alternative FCL+ ReLu. Note that in order to remove the

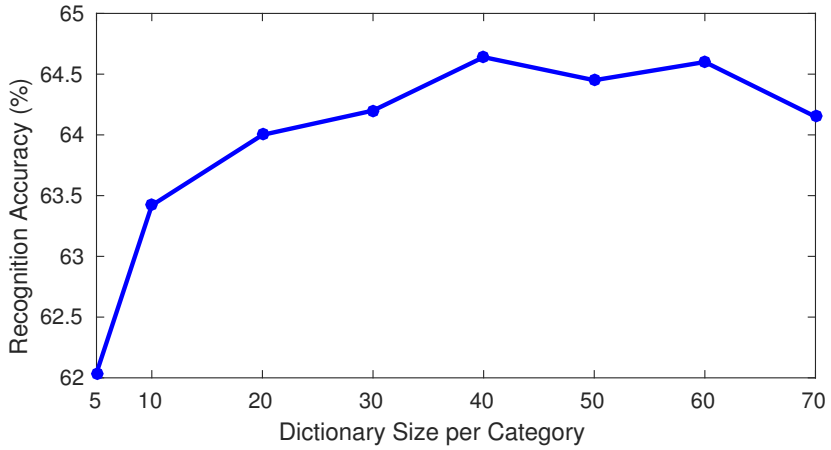


Fig. 4.6 Effect of dictionary size on the recognition accuracy.

improvement brought by the novel objective function design, we use the simple cross entropy loss $\mathcal{L}_1$ directly to enable a fair comparison.

Factor A1: Different Dictionary Size in the DLL

It is worth noting that an important factor within the DLL is the dictionary size, as it controls the model capacity. As shown in [92][185], the larger the size of the dictionary, the better is the performance of the traditional DL model. However, as the input of the DLL are the features learned by the deep network, it is still unclear what dictionary size fits best to these high level features. In general, the ideal DL method should achieve an acceptable level of performance using a relatively small dictionary size.

In this section, we use the MIT Indoor 67 dataset as an example for the evaluation and analysis. Specifically, we only replace the conventional FCL7 by our DLL followed by FCL8 as the classifier. For each class, we randomly choose 80 images for training and the rest for testing, similarly to the experimental setting in [108]. The parameters of the dictionary are initialized using a truncated normal distribution, with the number of dictionary atoms per class varying from 5 to 70. Fig.4.6 shows that the performance of the CNN-DL method improves when the dictionary size increases and reaches a maximum at around 40 atoms per category. This may relate to the diversity of training data under the particular category, while too many dictionary atoms per class may introduce information redundancy or noise into the feature representation.

Factor A2: Number of recurrent unit in the DLL

In this section, we evaluate the effect of the number of recurrent units in the DLL. Specifically, we only replace the last FCL with a DLL in the experiments. As shown in Fig.4.7, we plot the $\mathcal{L}_1$ sparseness measure (LSM) [106] of the conventional FCL output

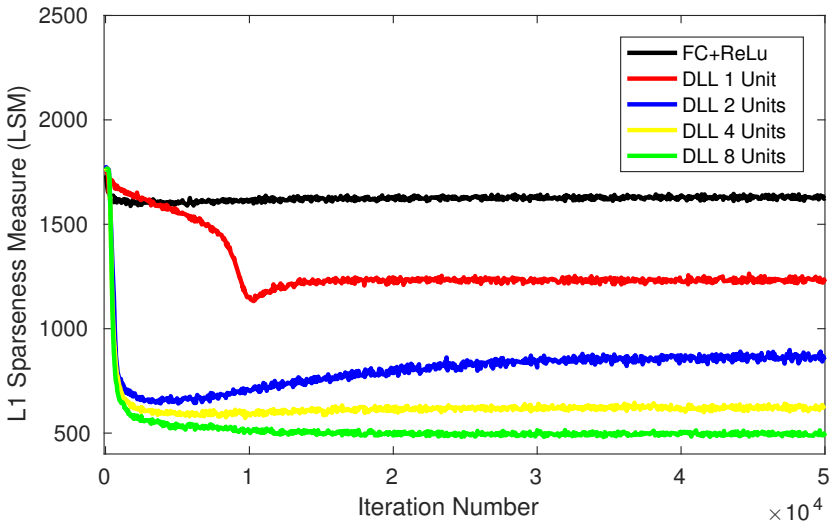


Fig. 4.7 LSM of DLL activations using different layers

Table 4.2 Effect of number of Recurrent Units in DLLs

Loss	Layer Setting	No.Units	Recognition Rate
$\mathcal{L}_1$	FCL7+FCL8	-	61.80
$\mathcal{L}_1$	FCL7+DLL8	1	63.21
$\mathcal{L}_1$	FCL7+DLL8	2	64.63
$\mathcal{L}_1$	FCL7+DLL8	4	64.32
$\mathcal{L}_1$	FCL7+DLL8	8	63.08

and the DLL output based on using 1,2,4 and 8 recurrent units. These LSM losses are based on inputting the same test datasets and calculating the $\mathcal{L}_1$ norm of their final outputs. It can be clearly shown that output of the FCL is less sparse than the others and with an increasing number of recurrent units, the DLL outputs are more sparse. Further, for the different sparseness levels, we can present the corresponding recognition rates in Table 4.3. The recognition rate increases and achieves the best rate of 64.63% when the number of recurrent units is two. As more units are utilized, the recognition rate falls to 63.08%, but is still better than using the FCL only. This experiment shows that only appropriate levels of sparseness can lead to good recognition rates.

Factor A3: Usage of Cascaded DLLs

Here, we analyze the effect of cascaded DLLs in the deep CNN framework by using different layerwise configurations, as shown in Table 4.4. For a fair comparison, we used 2 recurrent units and fixed the dictionary size to 2680. It is worth noting that, as the LISTA network also involves sparse coding, we compare our DLL with LISTA in this section. As

Table 4.3 Effect of number of Recurrent Units in DLLs

Layer Setting	No.Units	Recognition Rate
FCL	-	61.80
DLL	1	63.21
DLL	2	**64.63**
DLL	4	64.32
DLL	8	63.08

Table 4.4 Analysis for Factor A3: cascaded DLLs

Loss function	Layer Setting	Recognition Rate
$\mathcal{L}_1$	FC7+FC8	61.80
$\mathcal{L}_1$	DLL7+FC8	64.46
$\mathcal{L}_1$	FC7+DLL8	64.63
$\mathcal{L}_1$	DLL7+DLL8	65.19
$\mathcal{L}_1$	LISTA7+LISTA8	63.90

Table 4.5 Recognition Rates (%) for different factors setting

Loss function	Layer setting	Recognition Rate
$\mathcal{L}_1$	FC7+FC8	61.80
$\mathcal{L}_1$	DLL7 +DLL8	65.19
$\mathcal{L}_1 + \mathcal{L}_2$ (-)	DLL7 +DLL8	66.11
$\mathcal{L}_1 + \mathcal{L}_2$	DLL7 +DLL8	66.60

shown in Table 4.4, replacing FCL7 and FCL8 by two LISTA layers improves performance from 61.80% to 63.90% and replacing a single FCL with our proposed DLL increases performance to around 64.46% and 64.63%, that is already a 0.5% improvement to that yielded by the LISTA layers. Finally, the recognition rate rises to 65.19% when we replace both FCL layers with DLLs, yielding an improvement of 3.5% improvement compared to using FCLs only. This experiment shows the superiority of our approach compared with LISTA and FCL layers, and also shows that using two DLLs outperforms the use of one DLL. The reason that our DLL outperforms LISTA may be owing to the decoupling of the learned threshold into two factors, that is able to adapt the threshold in the shrinkage function of the DLL under the unknown sparse prior of the activation outputs. In addition, due to the fact that LISTA needs to learn an extra matrix in the sparse coding procedure while our DLL does not, our model can be trained more efficiently with fewer parameters. To sum up, the new proposed DLL contributes in yielding better scene recognition performance.

Factor B: Effect of different terms in the Objective Function

Table 4.6 Recognition Rates (%) for different architectures and multi-scale variations

Architecture	Pretrain Set	No. scales	15 Scenes		MIT Indoor 67		SUN 397	
			Alex	VGG	Alex	VGG	Alex	VGG
Baseline	IN	1	86.50	89.89	61.80	71.11	46.33	54.09
CNN-DL	IN	1	88.33	92.90	66.60	78.33	51.67	60.23
Baseline	PL	1	89.96	91.20	72.88	79.45	57.07	65.10
CNN-DL	PL	1	91.30	93.30	76.10	82.86	60.82	67.90
Hybrid-CNN	IN+PL	1	53.86	–	70.80	–	53.86	–
DAG-CNN	IN	1	–	91.90	–	77.50	–	56.20
DSP-CNN	IN	1	–	91.78	–	78.28	–	59.78
Dual	IN	2	92.16	93.84	71.87	79.04	56.62	61.07
Dual CNN-DL	IN	2	92.31	94.54	**76.56**	83.37	**59.03**	65.20
Dual	PL	2	93.80	95.18	76.87	83.43	62.60	67.59
Dual CNN-DL	PL	2	94.20	96.03	80.30	86.43	66.10	70.13
MOP-CNN	IN	3	–	–	68.88	–	51.98	–
CNN-DBL	IN	3	–	–	74.09	82.24	57.31	64.53
MFAFV-NET	IN/PL	3	–	–	75.01	82.66	57.15	64.59
URDL	IN	4	91.15	–	71.90	–	–	–
SFV	IN	4	–	–	72.86	–	54.40	–
SFV+PLACES	IN/PL	4+1	–	–	79.00	–	61.72	–

We will now investigate how the performance of CNN-DL is affected by the design of the objective function. We evaluate the loss function with and without the label discriminative regressor $\mathcal{L}_2$ of the sparse code and the corresponding constraint. A comparison of the recognition performances are shown in the Table 4.5[2]. We can see that with the help of the label discriminative regressor, the performance is consistently better than using only the simple softmax function. It verifies that the adoption of the novel discrimination term takes advantage of the structure of the dictionary in the DLL, consequently enhancing the discriminative capability of the overall network. In addition, the loss function with the constraints always shows superior performance to those without it. This shows that the constraint we developed for preventing over-fitting in training is effective.

4.3.3 Results and Comparisons

In this section, we compare our CNN-DL method with other existing scene classification approaches. The classification performance metric is the percentage of correctly classified test data. The benchmark algorithms for comparison are Hybrid-CNN [197], Multiscale orderless pooling CNN (MOP-CNN) [71], Semantic Fisher vector CNN (SFV) [46], hybrid

[2] $\mathcal{L}_1 + \mathcal{L}_2$ (-) represents using $\mathcal{L}_1 + \mathcal{L}_2$ as the objective function without the term to prevent overfitting

CNN and dictionary-based model (CNN-DBL) [178], Unified representative and discriminative learning model (URDL) [108], Directed acyclic graph CNN (DAG-CNNs) [186], Deep spatial pyramid CNN (DSP-CNN) [63] and Mixture of factor analyzers Fisher vector network (MFAFV-NET) [107]. Detailed information about each model is shown in Table 4.6, including the pre-train auxiliary dataset and the number of scales. As mentioned in section 4.3.1, we employ three widely used scene recognition datasets in our evaluation experiments, specifically, 15 Scene [103], MIT Indoor-67 [138], and Sun 397 [177].

More specifically, we first train the CNN based network on the auxiliary data, including ImageNet data (IN) or Place205 data (PL), then fine tune the network as described previously for different scene datasets. By replacing the FCLs with the proposed DLLs, we can evaluate the performance achieved by the CNN-DL method. It can be seen that the network pre-trained by PL always performs better than the counterpart pre-trained by IN, which corresponds with the findings in [197]. In addition, the VGG network architecture always demonstrates superior performance to the Alex network architecture owing to the exploitation of deeper models. When evaluating various methods on a single global scale, our CNN-DL method consistently outperforms the other CNN based scene recognition methods, which justifies the effectiveness of integrating the DLL into the network architecture. This may be because the DLL considers sparsity control of the neuron activation and the discrimination capability simultaneously. Furthermore, usage of the new constraint in (4.4) is able to prevent overfitting, balancing between the recognition accuracy of the training set and the generalization performance.

It worth noting that among the comparison approaches, the CNN-DBL and URDL methods are based on dictionary learning models, however, these techniques apply the dictionary learning model as a post-processing step disconnected from the training of CNN, and so do not fully harness the strength of DL since it is not integrated with the deep network. As these methods show improved performance when evaluated on multiple scales [178] [108], we also consider the multiple scale case for our CNN-DL method. Consequently, we evaluate pairwise combinations of CNNs used at two different scales, that we identify as Dual CNN-DL. As in [86], the dual architecture consists of two CNNs processing images at two scales. We regard this as the baseline for the two scales evaluation. Instead of concatenating the two resulting final FCL activations into a feature and training the SVM as in [86], we concatenate the sparse feature representation into a feature and train the SVM in this new discriminant latent space. The results in Table 4.6 show the improvement in the performance achieved by Dual CNN-DL (Pre-trained on IN), yielding an accuracy of 76.56% on MIT Indoor 67 and 59.03% on SUN 397 for only two scales. Compared with the other dictionary learning based models, our method obtains better performance on MIT Indoor 67 and SUN 397, while using only two scales compared with 3 or 4 scales in CNN-DBL and URDL.

In this work, we only combine two scales in a dual architecture for CNN-DL, though it is also possible to consider the combination of more scales at the cost of a more complex architecture.

4.4 Chapter Summary

In this chapter, we replace the conventional FCL and ReLu by our proposed nonlinear DLL. The proposed approach combines the strength of both a hierarchical prior (CNN) and a sparse prior (DL) on the feature learning procedure, i.e., is capable of controlling the sparsity of the neuron activation in the forward pass, while passing the error differentials from its outputs to inputs during back-propagation to update the dictionary. Our CNN-DL architecture takes advantage of the potential structure of the dictionary, in order to harness the discriminant capability of the features via a new label discriminative regressor. In addition, we propose new constraints to prevent overfitting, by incorporating the advantage of the Mahalanobis and Euclidean distances and balancing the recognition accuracy and generalization performance. We suggest an algorithm to train the whole network end-to-end by performing conventional back propagation and avoiding offline post-processing. The superior performance of the proposed method in comparison to the state-of-the-art is demonstrated using experiments employing various scene datasets.

Chapter 5

Multi-Task Adversarial Network for Disentangled Feature Learning

Multiple independent factors exist in the image generation process, however, in most applications, only some of the explanatory factors are of interests to us. Different explanatory factors of the data tend to change independently of each other in the input distribution, and only a few at a time tend to change when one considers a sequence of consecutive real-world inputs. These factors always influence the observation in a complex and correlated way that makes the representation learning for an AI-related task (i.e., image classification) quite challenging. In this chapter, our goal is to build a model that disentangles the underlying factors of image variants associated with particular attributes of interest, which effectively leads to an informative disentangled feature representation, consequently benefiting image understanding. There are three aspects concerning the benefits of disentangling feature representation, as will now be discussed:

Firstly, learning a disentangled feature representation that separates the various explanatory sources, should give rise to a representation significantly more robust to the complex and richly structured variations extant in natural data sources. A good feature representation should disentangle the underlying factors so that what one learns about one factor could generalise to many configurations of the other factors.

Secondly, a disentangled representation enables controllable generation of new data through a generative model. In other words, the use of a disentangled representation enables different realistic versions of an input image to be generated by varying the attribute values. By using continuous attribute values, we can even choose how much a specific attribute is perceivable in the generated image.

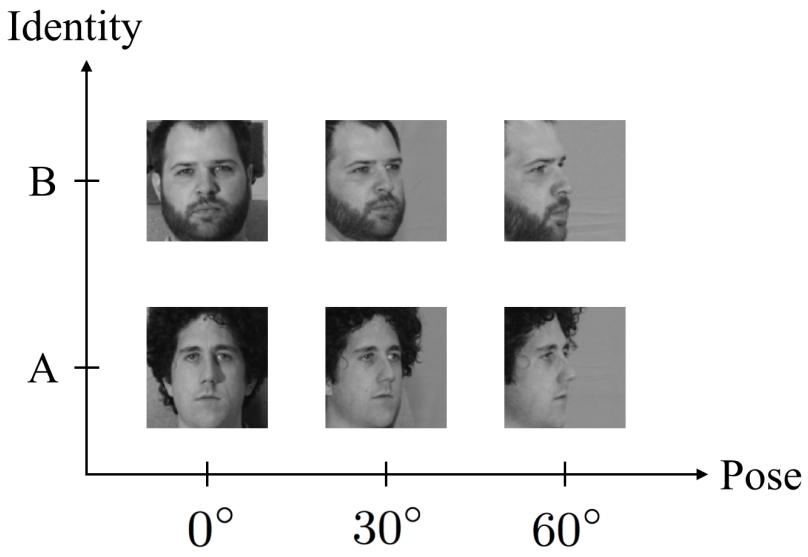


Fig. 5.1 Images of human faces are determined by two independent factors, i.e., identity and pose..

Thirdly, learning a disentangled feature representation leads to decisions that are potentially comprehensible to humans, i.e., it can improve the interpretability of the learned feature.

In this chapter, we address the problem of image feature learning for applications where multiple independent factors exist in the image generation process, and only some factors are of our interest. For example in Fig. 5.1, images of human faces are determined by two independent factors, i.e., identity and pose. If the application is face recognition, the identity is the factor of our interest. Our ultimate goal is to learn a disentangled representation of image variants associated with particular attributes of interest. More specifically, we present a novel multi-task adversarial network based on an encoder-discriminator-generator architecture. The encoder extracts a disentangled feature representation for the factors of interest. The discriminators classify each of the factors as individual tasks. The encoder and the discriminators are trained cooperatively on factors of interest, but in an adversarial way on factors of distraction. The generator provides further regularisation on the learned feature by reconstructing images with shared factors as the input image. We also design a new optimisation scheme to stabilise the adversarial optimisation process when multiple distributions need to be aligned. The rest of this chapter is organised as follows: Section 5.1 introduces the primary motivation and contribution of this chapter. Section 5.2 discusses some related work, including prior works on disentangled representation and adversarial training. Section 5.3 presents the details of the proposed multi-task adversarial network,

including the overall objective functions and the novel optimisation scheme to stabilise the adversarial optimisation process. We also compare our proposed approach with the three most relevant generative adversarial networks and auto-encoder variants. In section 5.4, the experiments conducted on face recognition and font recognition tasks show that our method outperforms the state-of-the-art methods in terms of both recognising the factors of interest and generalisation to images with unseen variations. Section 5.5 concludes this chapter.

5.1 Introduction

Image feature representation learning has been one of the central problems in Computer Vision. One of the most significant developments in the recent years in image feature learning is the resurgence of convolutional neural networks combined with large-scale datasets [99]. In this chapter, we are interested in extending convolutional neural network based feature learning to the problems where multiple independent underlying factors determine the image generation process but only some factors are of our interest.

For many practical applications, the image generation process can be well approximated by a small number of factors. For instance, images of printed text are determined by factors such as font and glyph and images of human faces are determined by factors such as identity, pose, and illumination. We further assume there exists a primary factor for a given application. For instance, in the case of text images, if the application is font recognition, then font is the primary factor. But if the application is character recognition, then glyph is the primary factor. Multi-task learning [28] is the traditional approach to leverage the additional factors that are present in the image generation process. It learns a shared representation to predict all the factors. By doing so, we obtain features that can potentially outperform those learned from individual factors. However, if we are only interested in the performance of the primary factor, such as the identity for face images, can we do better than conventional multi-task learning?

Another major challenge in feature learning is generalization. We want the features learned from training data to perform well on test data that have never been seen in training. In the case of factored image generation processes, one particularly interesting generalization is to unseen variations of non-primary factors. For instance, if our problem is font recognition, we are interested in a feature representation that is robust to glyphs that have never been seen in training. Generalization is usually accomplished by seeing as many data variations as possible in training. However, in the case of images with factors, this would mean that we need to potentially see images with all the combinations of all the factors. We end up with an explosion of images (exponential with respect to the number of factors in the worse case).

The interesting question is whether it is necessary to train on all the combinations of all the factors in order to generalize if there is a primary factor.

In this chapter, we propose a novel feature learning algorithm for factored image generation processes that answers the two questions posed in the previous two paragraphs. Without loss of generality, we assume that the image generation process contains two independent (uncorrelated) factors and we are only interested in recognizing one of two. We refer to the factor of interest as the *content factor* and the other as the *style factor*. The key idea of the proposed apporach is that instead of learning from both factors in a cooperative way (traditional multi-task learning where both tasks help each other), we formulate the problem as learning from two *adversarial* tasks. To be more precise, given an input image with a content label and a style label, one task is to learn a content classifier and a shared image feature that labels the image correctly according to the content label. The other task is to learn a style classifier and the same image feature that would label the image maximally incorrectly according to the style label. Through the adversarial process, we learn an image feature that outperforms that of multi-task learning on the content factor and generalizes to new images with both unseen content and style factors.

The overall framework of the proposed multi-task adversarial network (MTAN) is shown in Fig. 5.2. There are four main components in our adversarial multi-task formulation: an encoder network, a generator network, and two discriminator networks. An input image is fed into the encoder network which produces the target feature representation. The feature is used as input to a content discriminator and a style discriminator. Both the encoder and the content discriminator work cooperatively to minimize a classification loss driven by the content label, while the encoder and the style discriminator play an adversarial game in which the interaction is modeled by a minimax optimization over the prediction of the style label. The two classification tasks are essentially competing with each other as the difference between the content and style classification losses is used to train the feature encoder. To ensure the encoded feature contains a full description of the image content, we also add a generator network to produce an image that matches the content of the input image and the style of a given style indicator. Depending on whether the style indicator matches the style label of the input image, the generator is trained to either reconstruct the input image or transfer it to a different style. By combining the encoder, the generator, and the two discriminators, we obtain a feature that is optimized with respect to the content factor while being style-agnostic. In this way the feature can generalize to unseen style factors without causing confusion on content understanding.

Similarly to other generative adversarial networks (GAN) [4][73], the training process of the propose network architecture tends to suffer from unstable numerical optimization

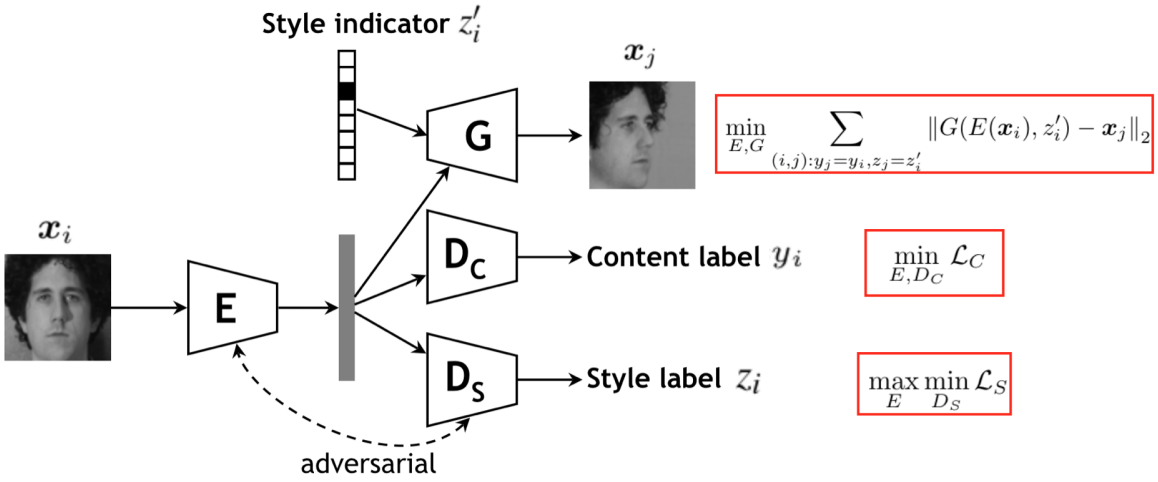


Fig. 5.2 Overall network architecture of our proposed adversarial disentangling model. The model is composed of four components: an encoder, a generator, a content discriminator, and a style discriminator. The encoder is used to extract the content feature representation of the image, which is good for content recognition but not for differentiating the styles.

due to the minimax loss function. Moreover, in the problems we are interested in, the style discriminator may need to distinguish as many as hundreds or thousands of classes as opposed to a binary decision (real or fake) as in most existing GANs. If we break the multi-class problem into a set of binary classification problems, we are actually required to solve a set of minimax problems coupled by the same encoder, which is much more challenging than for GAN. To tackle the problem, we extend the Wasserstein GAN (WGAN) [4] algorithm to address the multi-class scenario, which significantly improves the training stability.

The main contributions of this chapter are three-fold: 1. We propose a multi-task adversarial network that learns a disentangled feature representation through adversarial training of competing tasks on independent (uncorrelated) image factors. 2. We achieve stable optimization of multiple minimax losses by extending the WGAN algorithm [4] to the multi-class scenario. 3. Our feature representation outperforms standard cooperative multi-task learning methods, and achieves the state-of-the-art performance on the face recognition and font recognition datasets. Our approach can better generalize to unseen variations in both content factors and style factors.

5.2 Related Work

5.2.1 Disentangled Representation

There is a large quantity of literature concerning learning disentangled representations. The bi-linear model is among the first to separate the content and style in the underlying set of observations [159]. With the recent development of deep learning, auto-encoders [33] [87] [88] and Boltzmann machines [143] are adopted as regularizers to combine the discrimination and self-reconstruction criteria, thus discovering the factors of variation beside those relevant for classification. In particular, Predictability Minimization [148] and the fair variational auto-encoder [115] encourage independence between different latent factors. In addition to reconstructing the input, [135] [188] synthesize other images with the same content but with a different style to implicitly disentangle features. With the help of GANs, the work of [45] [53] [102] [124] further explores the application of disentangled representations in computer graphics and video prediction.

Our proposed method differs from the previous methods by combining cross-style image generation with an adversarial training strategy to learn disentangled features. It is worth noting that although [160] also uses this combination for feature disentangling, our proposed multi-task adversarial network differs in the following respects. Since our main goal is to improve content classification performance instead of synthesizing high-quality images, we employ multi-task adversarial training on the latent feature representation, instead of on the synthesized image and the real image. This is designed for explicitly learning a disentangled latent feature that is good for content recognition but not for style recognition. By combining cross-style image generation, the learned feature is not only inclusive or generative for synthesizing a content-preserving image, but also exclusive or invariant to style variations, thus benefiting image classification.

5.2.2 Adversarial Learning

Adversarial training has been explored for representation learning in various computer vision applications. In most GANs, the aim is to minimize the divergence between the distribution of real and fake images. A similar adversarial training strategy has also been adopted in feature learning in domain adaptation [162] and video prediction [45]. However, these methods use binary adversarial objective functions, which means their adversarial training can only be generalized to the cases where data comes from no more than two distributions. In contrast to their work, our proposed adversarial method considers multiple distributions. As shown in the adversarial branch in Fig. 5.2, if we break the multi-class problem into a

set of binary classification problems, we are actually required to optimize a set of minimax problems simultaneously, which are coupled by the same encoder.

It is worth noting that some works on GANs have claimed that they consider multiple categorical GANs, e.g., the Semi-Supervised GAN [131] and the DR-GAN [160]. Indeed, they have added a new branch for the multi-categorical classification, but in these previous approaches, the competing adversarial loss only confuses the discriminator by using two distributions (real or generated) and no adversarial strategies are adopted between different categories in the auxiliary multi-categorical classifier branch. In our work, the target of the encoder is to confuse the style classifier with any two classes, which aims to reduce the feature distribution discrepancy of any two style classes. There is no "real" reference class that can guide the distribution of other classes. In Section 5.3.2, we will provide an in-depth discussion concerning the difference between our proposals and the most relevant work concerning conventional GANs.

It is also worth noting that min-max optimization always suffers from training instability as has been observed previously in related GAN research. In our model formulation, multiple min-max problems require to be optimized simultaneously, which further aggravates the training difficulties. Recent work in [4][136] have proposed methods, that are resilient to vanishing gradient and model collapse even with an over-trained loss function or a mildly changed network architecture. In this way, the GAN network can be trained without properly balancing the generator and discriminator, which leads to improved training stability. However, these methods do not consider the adversarial loss for multiple distributions larger than two. Therefore, our work verifies whether these approaches can be extended to address multiple distributions.

5.3 Multi-Task Adversarial Network

The overall framework of multi-task adversarial network (MTAN) is illustrated in the Fig. 5.2, where arrows indicate the forward propagation direction. It is composed of four components, including the encoder E, the generator G, the content classifier D_C and the style classifier D_S.

Assume we have an image $\boldsymbol{x}$ with discrete content label $y \in \mathcal{Y}$ and discrete style label $z \in \mathcal{Z}$. The image is first mapped by the encoder to its latent feature representation $E(\boldsymbol{x})$. Based on this latent representation, the discriminators try to predict class distributions $D_C(E(\boldsymbol{x})) \in \mathbb{R}^{|\mathcal{Y}|}$ and $D_S(E(\boldsymbol{x})) \in \mathbb{R}^{|\mathcal{Z}|}$ for content and style, respectively. As our goal is to encode image content information while removing any style variations in the learned feature representation, a good encoder E should extract a feature that is good for the content discriminator D_C but bad for the style discriminator D_S. Based on this intuition, we formulate

the following adversarial multi-task training objectives:

$$\min_{E,D_C} \mathcal{L}_C \tag{5.1}$$

$$\max_{E} \min_{D_S} \mathcal{L}_S. \tag{5.2}$$

As can be seen, E and D_C work cooperatively to minimize the content classification loss $\mathcal{L}_C$, which is a conventional cross-entropy loss between ground truth y and prediction $D_C(E(\boldsymbol{x}))$. On the other hand, E and D_S play an adversarial game on the style loss $\mathcal{L}_S$, where E tries to minimize the divergence of feature distributions for different style classes so that D_S fails to correctly classify sample style no matter how hard it tries. Ideally, at the end of the competition, D_S can perform no better than a random guess. The idea behind (5.2) is the same as in GAN [73], although our goal is to learn disentangled features instead of generating images. Moreover, the style loss $\mathcal{L}_S$ typically involves a large number of classes as opposed to the binary classification in GAN. In our setting, a "real" reference class does not exist that can guide the distribution of other classes; yet no trivial solution for E will be obtained as the encoder is also constrained by (5.1). The details of the loss function and training scheme for (5.2) will be given in Section 5.3.1.

Besides the multi-task classification branches, our network also includes a generation branch. The generator takes an encoded latent feature as well as a target style indicator z' as inputs, and outputs a synthesized image $G(E(\boldsymbol{x}), z')$ which shares the same content of $\boldsymbol{x}$ but is rendered in the style of z'. The training for the generation branch is guided by an $\mathcal{L}_2$ reconstruction loss:

$$\min_{E,G} \sum_{(i,j):y_j=y_i,z_j=z_i'} \left\| G(E(\boldsymbol{x}_i), z_i') - \boldsymbol{x}_j \right\|_2, \tag{5.3}$$

where subscripts i and j denote data indices. z_i' is randomly sampled from $\mathcal{Z}$. When $z_i' = z_i$, G tries to reconstruct the original input $\boldsymbol{x}_i$; otherwise, G generates a style-transferred version of $\boldsymbol{x}_i$ that matches a corresponding sample $\boldsymbol{x}_j$ with style z_i' in the training database. The encoder-generator design contributes to content feature disentangling in an implicit way, and makes the encoded feature more inclusive of the image content.

5.3.1 Multi-Class Adversarial Training

Here we discuss the loss function and optimization strategy for the adversarial style classification in equation (5.2). The classification loss on a training pair $\{\boldsymbol{x}, z\}$ can be defined as the

cross-entropy between predicted class distribution $D_S(E(\boldsymbol{x}))$ and ground truth label z:

$$\ell_{CE}(\boldsymbol{x}, z) = -\sum_{k \in \mathcal{Z}} \delta(z-k) \log D_S(E(\boldsymbol{x}), k), \tag{5.4}$$

where $\delta(\cdot)$ is the Dirac delta function, and $D_S(E(\boldsymbol{x}), k)$ denotes D_S's prediction score for the k-th style class. However, as pointed out in [4], cross-entropy is not a stable loss if there is a big disparity between the predicted distribution and target distribution. With the loss in (5.4), optimization in our case becomes even more unstable due to the large number of style classes and the absence of a fixed reference distribution.

Following the idea of WGAN [4], we improve optimization stability by replacing the cross-entropy loss with Earth Mover's Distance (EMD). As the goal of encoder E here is to match the feature distributions of multiple style classes, we need to calculate EMD for each pair of distinct styles. A more efficient alternative is to construct the pairs in a one-versus-all way, which gives the following multiple-distribution matching objective:

$$\min_{E} \sum_{k \in \mathcal{Z}} W\left(p(E(X)|Z=k), p(E(X)|Z \neq k)\right), \tag{5.5}$$

where $W(\cdot,\cdot)$ is the EMD distance, X and Z are the random variables for $\boldsymbol{x}$ and z. With the same approximation used in WGAN, the problem above can be converted to

$$\min_{E} \sum_{k \in \mathcal{Z}} \max_{D_S \in \mathcal{D}} \mathbb{E}_{\boldsymbol{x} \sim p(X|Z=k)} D_S(E(\boldsymbol{x}), k) - \mathbb{E}_{\boldsymbol{x} \sim p(X|Z \neq k)} D_S(E(\boldsymbol{x}), k), \tag{5.6}$$

where $\mathcal{D}$ is the space of all K-Lipschitz functions for some K. As D_S is shared by all the $|\mathcal{Z}|$ EMD operations, it is very hard to simultaneously achieve the optima for all the inner max problems. As an approximation, we switch the order of summation and maximization in (5.6), and arrive at our final objective function:

$$\min_{E} \max_{D_S \in \mathcal{D}} \sum_{i} -\ell_{EMD}(\boldsymbol{x}_i, z_i), \tag{5.7}$$

and

$$\begin{aligned} \ell_{EMD}(\boldsymbol{x}, z) &= -D_S(E(\boldsymbol{x}), z) + \frac{\sum_{k \neq z} D_S(E(\boldsymbol{x}), k)}{|\mathcal{Z}| - 1} \\ &= < \boldsymbol{z}, D_S(E(\boldsymbol{x})) >, \end{aligned} \tag{5.8}$$

where $\boldsymbol{z} \in \mathbb{R}^{|\mathcal{Z}|}$ is a vector representation of z, with the z-th element equal to -1 and all the others equal to $1/(|\mathcal{Z}|-1)$. Note that when $|\mathcal{Z}| = 2$, our optimization objective reduces to the original WGAN.

To enforce the K-Lipschitz condition for D_S, we select $K = 1$ and adopt a gradient loss as in the improved WGAN [79] which is extended to multi-class as follows:

$$\mathcal{L}_R = \sum_{k \in \mathcal{Z}} \mathbb{E}_{\boldsymbol{u}}(\|\nabla_{\boldsymbol{u}} D_S(\boldsymbol{u}, k)\|_2 - 1)^2. \tag{5.9}$$

In practice, we sample latent feature $\boldsymbol{u}$ uniformly along the straight lines connecting pairs of training data $(E(\boldsymbol{x}_i), E(\boldsymbol{x}_j))$, where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are randomly sampled from training batch with different style labels: $z_i \neq z_j$. Note we interpolate feature points between any two style distributions rather than just two distributions as in the improved WGAN.

Finally, based on the loss terms in (5.8) and (5.9), we can formalize the style loss $\mathcal{L}_S$ as

$$\mathcal{L}_S = \sum_i \ell_{EMD}(\boldsymbol{x}_i, z_i) + \lambda \mathcal{L}_R, \tag{5.10}$$

where λ is a weighting parameter.

5.3.2 Comparison with Prior Adversarial Models

We compare MTAN with the three most relevant GAN and Auto-encoder variants as shown in Fig. 5.3.

Semi-Supervised GAN: The Semi-Supervised GAN aims to learn a discriminative classifier where the discriminator D is trained to not only distinguish between real and fake, but also classify the real image in to K classes. D outputs a $(K+1)$-dim vector, in which the last dimension represents real/fake decision. The generator G aims to fool D by aligning two distributions, i.e., the distribution for real and fake images. MTAN differs to semi-supervised GAN in two aspects. Firstly, the input of the discriminator is the latent feature representation, instead of the real and synthesized image, and the goal of the encoder is to align the feature distributions between any two different style classes. Secondly, the encoder-generator structure learns a disentangled content representation implicitly by utilizing a target style indicator to generate images.

Adversarial Auto-encoder (AAE): In AAE, the auto-encoders $(E+G)$ reconstruct the input image, and the latent vector generated by the encoder matches an arbitrary prior distribution by training discriminator D. The MTAN model shares a similar image generation loss for generator but has two major differences. Firstly, besides the latent vector, we provide a target style indicator to the generator and generate a new image with the same content but in the style indicated. Secondly, we take advantage of an additional style variation classification task to disentangle the content-preserving feature representation explicitly.

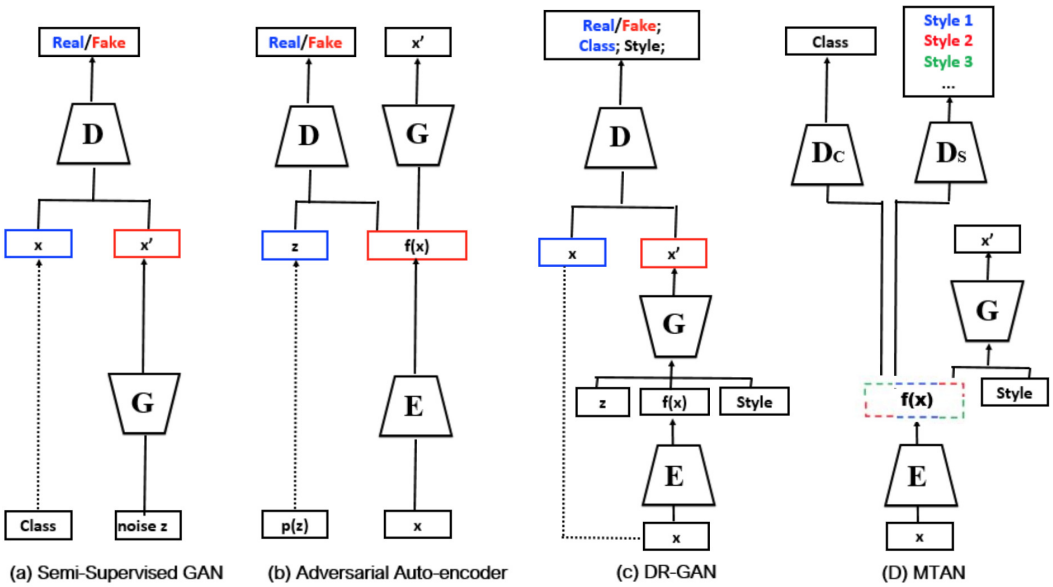


Fig. 5.3 Comparison of previous GAN and auto-encoders architectures with our proposed MTAN.

DR-GAN: DR-GAN uses the encoder-generator structure to synthesize images. The goal of the encoder and generator is to fool Discriminator D to classify the synthesized image $\boldsymbol{x}'$ to the identity of input $\boldsymbol{x}$ and target style variation. Compared with MTAN, although they both adopt the encoder-generator structure for image synthesis, the goal of the discriminator and encoder is different in two respects. Firstly, the input of the discriminator is the extracted feature representation instead of the real and synthesized images. Secondly, the goal of the encoder is to extract such a latent feature representation, that contains little discriminative information about the style type, thus fooling the discriminator to make a random guess. In other words, the encoder needs to match or align the feature distribution between any two different style classes, instead of only real and fake distributions.

5.4 Experiments and Results

In this section, we evaluate our feature disentangling method on two content-style image classification datasets, i.e., font and face datasets. We implement our approach based on Tensorflow and the relevant code is available at https://github.com/BeCarefulPlease/Ch5. We quantitatively evaluate the recognition accuracy using the disentangled representation as the content features with a cosine distance metric [137]. We also show qualitative results of

(a) Example of three different Fonts (b) Example of three different Glyphs

Fig. 5.4 Examples of the images with different fonts or glyphs. (a) Example of three different Fonts ; (b) Example of three different Glyphs.

synthetic images to demonstrate that the learned feature is inclusive for generating content-preserving images.

5.4.1 Evaluation on Font Recognition Dataset

We evaluate our method using various sub-models (that utilize selected elements of the MTAN model) in order to study the effectiveness and significance of each part of the MTAN model. We also analyze the training stability and generalization ability by testing on large or unseen variations.

Dataset and Experiment Setting

For our evaluation, we built a Japanese font dataset, specifically for the font recognition task. The reason we choose the Japanese language is that a large number of glyphs in the Japanese language introduce a large intra-class variation for the font recognition task, which makes the problem more challenging. Fig. 5.4(a) presents a glyph with three different fonts, while Fig. 5.4(b) presents three different glyphs having the same font.

We collected 300 fonts in total. In addition, we randomly split the data into 200 font classes for training and used the remaining 100 font classes for testing. In the training stage, we use 50 frequently used glyphs as the style variation within each font class, and use the font file of a particular font to render this predefined glyph to form a training sample. We consider the following three settings to evaluate the generalization performance of our method when partial font classes or glyph styles are missing in the training stage. More specifically, one image per font category (with the same glyph) is randomly selected as the gallery and the others are the query ones throughout this chapter unless otherwise specified.

Unseen Font: We test our model on the images from the remaining 100 unseen font classes, in which the glyph style is the same as that covered in the training stage.

Unseen Glyph: We select another 50 glyphs, that are different from the training ones, and rendered them by the 200 seen font classes as the test set. We use the trained font classifier D_C for testing.

Table 5.1 Network Structure for Font Recognition

Encoder and Discriminators			Generator		
Layer	Filter/Stride	Output Size	Layer	Filter/Stride	Output Size
			$FConv$		$6 \times 6 \times 512$
$Conv11$	$3 \times 3/1$	$96 \times 96 \times 32$	$FConv5$	$3 \times 3/2$	$12 \times 12 \times 256$
$Conv12$	$3 \times 3/1$	$96 \times 96 \times 64$	$FConv4$	$3 \times 3/2$	$24 \times 24 \times 128$
$Conv21$	$3 \times 3/2$	$48 \times 48 \times 64$	$FConv32$	$3 \times 3/1$	$24 \times 24 \times 64$
$Conv22$	$3 \times 3/1$	$48 \times 48 \times 64$	$FConv31$	$3 \times 3/2$	$48 \times 48 \times 64$
$Conv31$	$3 \times 3/2$	$24 \times 24 \times 64$	$FConv22$	$3 \times 3/1$	$48 \times 48 \times 64$
$Conv32$	$3 \times 3/1$	$24 \times 24 \times 128$	$FConv21$	$3 \times 3/2$	$96 \times 96 \times 64$
$Conv4$	$3 \times 3/2$	$12 \times 12 \times 256$	$FConv12$	$3 \times 3/1$	$96 \times 96 \times 32$
$Conv5$	$3 \times 3/2$	$6 \times 6 \times 512$	$FConv11$	$3 \times 3/1$	$96 \times 96 \times 1$
$AvgPool$	$6 \times 6/1$	$1 \times 1 \times 512$			
$FC1(D_C)$		256			
$FC2(D_C)$		N^C			
$FC1(D_S)$		256			
$FC2(D_S)$		N^S			

Unseen Font and Glyph: We select another 50 glyphs, that are different from the training ones, as the variation for the test set, and the images are rendered by other 100 unseen font classes.

Network structure and Implementation Details

For the encoder and generator, layer normalization is applied after each convolutional layer. Since the stability of the adversarial training suffers if sparse gradient layers are used, we replace MaxPool and ReLu with stride convolution and exponential linear units respectively. Each discriminator (D_C and D_S) contains two fully connected layers. The output of the encoder is the disentangled content feature representation $f(x) \in \mathbb{R}^{512}$. $f(x)$ is then concatenated with a target glyph indicator. The generator contains a series of fractional-stride convolutions [15], which transforms the (512+$|\mathcal{Z}|$)-dim concatenated vector into a synthesized image, which has the same size as the original input image. The detailed information of the network structure for font recognition is provided in the Table 5.1.

We render all font images to size 96×96. The image intensities are linearly scaled to the range of [-1,1]. The batch size is set to be 256. An Adam optimizer [96] is used with a learning rate of 0.001 and momentum 0.5. We alternate between one step of optimizing the discriminators and generator, and one step of optimizing the encoder.

We evaluate the font recognition performance under the three test settings and compare with the following prior work, i.e., the Controlled Pose Feature (CPF) [188] and the Disentangled Representation learning-Generative Adversarial Network (DR-GAN) [160]. We

Table 5.2 Recognition rate (%) comparison on Font database

Model	Unseen Font	Unseen Glyph	Unseen Both
CPF [188]	45.1	44.6	28.3
DR-GAN [160]	46.4	50.5	31.9
D_C	36.5	38.6	22.9
G	44.4	42.0	27.7
$D_{(C+S)}$	24.3	42.3	13.8
$D_{(C-S)}$	43.2	49.5	30.4
$D_{(C-S)}+G$	**47.9**	**52.8**	**34.8**

also present an ablation study for each module that we designed in our proposed method. Specifically, besides the models in [160] [188], we also evaluate and compare with the following models:

1. **Single-Task** (D_C): trained on encoder and the font classifier $\boldsymbol{D}_C$ only using a soft-max cross entropy loss.
2. **Encoder-Generator** (G): trained on the encoder and the generator only using an $\mathcal{L}_2$ reconstruction loss for image generation.
3. **Multi-Task** $(D_{(C+S)})$: trained on encoder, font and glyph classifiers with conventional multi-task training to improve both the font and glyph recognition performance cooperatively.
4. **Adversarial-Task** $(D_{(C-S)})$: trained on encoder, font and glyph classifiers with adversarial training to a learn a disentangled representation as proposed in our model.
5. **MTAN** $(D_{(C-S)}+G)$: trained on all modules designed in the proposed method, including the adversarial training and image reconstruction requirement to learn a disentangled representation.

The performance of the 7 chosen models are presented in Table 5.2. The single-task model trained on the encoder and font classifier is only intended to serve as the baseline. As shown in the table, introducing different combinations of the modules we designed all boost the performance under all three settings. It is worth noting that although the multi-task and the adversarial task model both leverage the font and glyph label supervision, the disentangled feature obtained from adversarial training performs much better than that achieved by conventional multi-task learning, especially when testing on the unseen font. Combining the adversarial training and image reconstruction requirement together, MTAN achieves the best performance among these 7 models in all test settings. It is worth noting that although DR-GAN also takes advantages of this combination, our method outperforms DR-GAN by around 2%. This may be because the MTAN uses the multi-task adversarial training on the latent feature representation, instead of on the real and synthesized images, to

disentangle the content feature explicitly. It makes the disentangled feature become more discriminant for the font classification task.

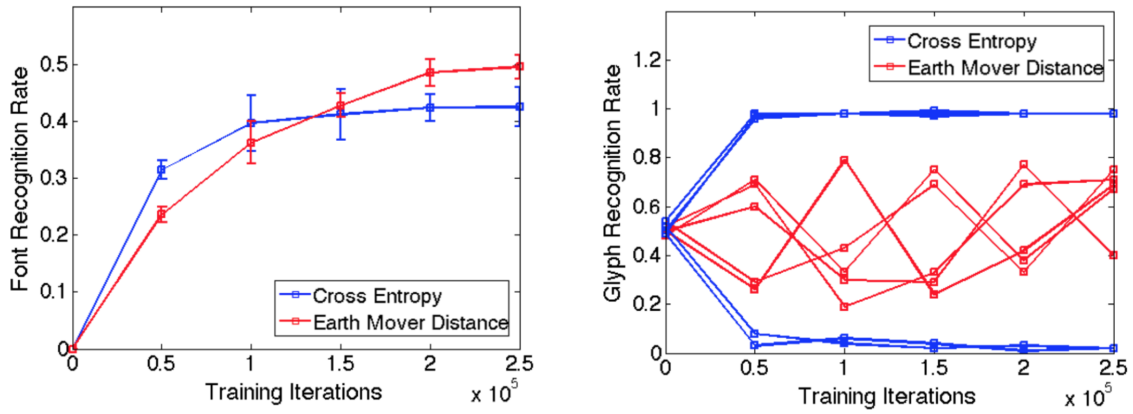
Another interesting observation is that, for the test on the unseen font, when only adding one module into the single-task network, adding the generator module yields most of the available performance gain. It means that image generation makes the learned representation more inclusive or generative for synthesizing content-preserving images. The use of image generation also makes the learned feature applicable for extracting the content feature for the images coming from novel classes (not covered in the training stage). On the other hand, for the test on an unseen glyph, when only adding one module into the single-task network, using the adversarial discriminators yields most of the available performance gain. It means that the multi-task adversarial training between the encoder and two discriminators acts as the critical role in learning a disentangled content representation that is exclusive or invariant to glyph variations. It is ideal for the font recognition, especially on images with large or unseen variations during training.

Analysis

Training stability: We analyze the training stability for our proposed model and compare it with the model trained using cross-entropy as the style classification loss. Each model is trained for 5 times to get reliable observations. Fig. 5.5 shows the accuracy for font recognition (a) and glyph recognition (b) on the test set achieved by the two models during training. Fig. 5.5 (a) shows that using the EMD loss give results that consistently converge to a higher font recognition accuracy than those using cross-entropy loss, although the convergence rate is slower. The error bars in (a) represent the standard deviation of the test accuracy.

Fig. 5.5 (b) plots the glyph recognition accuracy for all the training trials of the two models. The model trained with cross-entropy converges to either 1 or 0 quickly, indicating that the balance between E and D_S cannot be well maintained and the model collapses to a local optimum. In contrast, the glyph accuracy of the model trained with EMD loss mildly oscillates around 50%, and the variation among different trials is small. In this way, the competition between E and D_S is more effective. Therefore, our proposed EMD loss can improve multi-class adversarial optimization stability, leading to better disentangled representation for font content.

Font image synthesis: Our generator is trained to synthesize new glyphs with the same font style as an input font feature. The feature could be calculated based on one input glyph image or the average of multiple glyphs from the same font. The identity of the new glyph is specified by the style (glyph) indicator. Fig. 5.6 shows some visualizations of the



(a) Font recognition accuracy (b) Glyph recognition accuracy

Fig. 5.5 Comparison of font and glyph recognition accuracy on test set between models trained with different adversarial losses. (a) the mean and standard deviation (error bar) of font recognition; (b) the glyph recognition accuracy in multiple training trails.

synthesized images. The synthetic glyphs are similar to the ground truth with well-preserved font attributes such as weights, and some fine-grained attributes like serif or sans-serif are also captured. The synthesized images demonstrate the learned disentangled feature includes most of the content-relevant information (font) which is faithfully reconstructed in the synthesized images.

5.4.2 Evaluation on Face Reognition Dataset

We also evaluate our method on the face recognition dataset and compare with other state-of-the-art pose-invariant face recognition approaches. We also demonstrate our method can be used to disentangle more than one type of style variation.

Dataset and Experiment Setting

Multi-PIE [78] is a large database used for evaluating face recognition under pose, illumination, and expression variations in a controlled setting. The Multi-PIE face dataset contains more than 750,000 images of 337 people recorded in up to four sessions over the span of five months. Subjects were imaged from 15 view points and for 19 illumination conditions while displaying a range of facial expressions. We adopt this dataset for our evaluation because there are multiple distraction style factors, which makes the problem more challenging. Following the experimental setting in [199], we use 337 subjects with the neutral expression, 9 poses within 60^0, and 20 illuminations for evaluation of our proposed approach. The first 200 subjects are used for training and the remaining 137 for testing. For testing, one image

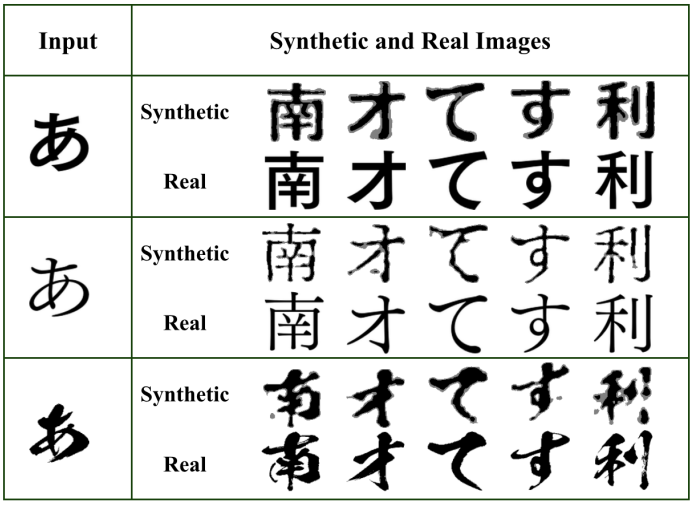


Fig. 5.6 Synthetic images that matches the font of the input image and the style of given style indicators. We compare synthetic images (top) and their ground truth images (bottom).

per subject with the frontal view and neutral illumination is the gallery and the others are the query samples. For Multi-PIE experiments, we disentangle more than two style factors, i.e., illumination and pose from the face identity. More specifically, we add additional pose and illumination codes as the input of the generator, and use two style discriminators to disentangle features explicitly.

Network Structure and Implementation Details

To be consistent with the experimental setting of the comparison approaches, we adopt CASIA-NET [187] for the encoder and the generator design, where batch normalization and an exponential linear unit are utilized after each convolutional layer. The identity, pose and illumination discriminators are stacked after the encoder. Each of them contains one fully connected layer. The output of the encoder is the identity representation $f(x) \in \mathbb{R}^{320}$, and this feature representation is then concatenated with a target pose indicator $\boldsymbol{z}_p \in \mathbb{R}^9$ and a target illumination indicator $\boldsymbol{z}_i \in \mathbb{R}^{20}$. Finally, the generator, which contains a series of fractional-stride convolutions [15], transforms the concatenated vector into a synthetic image. The detailed information of the network structure for face recognition is provided in Table 5.3.

We follow the same data pre-processing as [160] [187]. The batch size is 64, and all weights are initialized from the zero-centered normal distribution with a standard deviation 0.02. An Adam optimizer [96] is used with a learning rate of 0.0002 and momentum 0.5. We

Table 5.3 Network Structure for Face Recognition

Encoder and Discriminators			Generator		
Layer	Filter/Stride	Output Size	Layer	Filter/Stride	Output Size
			FC		$6 \times 6 \times 320$
$Conv11$	$3 \times 3/1$	$96 \times 96 \times 32$	$FConv52$	$3 \times 3/1$	$6 \times 6 \times 160$
$Conv12$	$3 \times 3/1$	$96 \times 96 \times 64$	$FConv51$	$3 \times 3/2$	$12 \times 12 \times 256$
$Conv21$	$3 \times 3/2$	$48 \times 48 \times 64$	$FConv43$	$3 \times 3/2$	$12 \times 12 \times 256$
$Conv22$	$3 \times 3/1$	$48 \times 48 \times 64$	$FConv42$	$3 \times 3/1$	$12 \times 12 \times 128$
$Conv23$	$3 \times 3/1$	$48 \times 48 \times 128$	$FConv41$	$3 \times 3/1$	$12 \times 12 \times 192$
$Conv31$	$3 \times 3/2$	$24 \times 24 \times 128$	$FConv33$	$3 \times 3/2$	$24 \times 24 \times 192$
$Conv32$	$3 \times 3/1$	$24 \times 24 \times 96$	$FConv32$	$3 \times 3/1$	$24 \times 24 \times 96$
$Conv33$	$3 \times 3/1$	$24 \times 24 \times 192$	$FConv31$	$3 \times 3/1$	$24 \times 24 \times 128$
$Conv41$	$3 \times 3/2$	$12 \times 12 \times 192$	$FConv23$	$3 \times 3/2$	$48 \times 48 \times 128$
$Conv42$	$3 \times 3/1$	$12 \times 12 \times 128$	$FConv22$	$3 \times 3/1$	$48 \times 48 \times 64$
$Conv43$	$3 \times 3/1$	$12 \times 12 \times 256$	$FConv21$	$3 \times 3/1$	$48 \times 48 \times 64$
$Conv51$	$3 \times 3/2$	$6 \times 6 \times 256$	$FConv13$	$3 \times 3/2$	$96 \times 96 \times 64$
$Conv52$	$3 \times 3/1$	$6 \times 6 \times 160$	$FConv12$	$3 \times 3/1$	$96 \times 96 \times 32$
$Conv53$	$3 \times 3/1$	$6 \times 6 \times 320$	$FConv11$	$3 \times 3/1$	$96 \times 96 \times 1$
$AvgPool$	$6 \times 6/1$	$1 \times 1 \times 320$			
$FC(D_C)$		N^C			
$FC(D_S)$		N^S			

update G more frequently than D, i.e., 5 steps for optimizing the encoder and 1 step for the classifiers.

Results and Comparisons

In this section, we compare our proposed method with other existing pose-invariant face recognition approaches. The benchmark algorithms for comparison are Face Identity-Preserving (FIP) [199], multi-view perception (MVP) [200], multi-view deep network (MvDN) [93], Controlled Pose Feature (CPF) [188] and Disentangled Representation learning-Generative Adversarial Network (DR-GAN) [160].

Table 5.4 shows the face recognition performance on MultiPIE of our methods compared with the existing methods under the same setting, except for DR-GAN which uses multipe images for testing [160]. The components of the proposed MTAN are evaluated individually under the following settings:

1. **MTAN w/o D**: trained on the encoder and the generator only for image generation without using any adversarial training.

2. **MTAN1** ($D_{(C-S1)} + G$): trained on all modules, using adversarial training and having an image reconstruction requirement to disentangle two factors (identity and pose).

Table 5.4 Recognition rate (%) comparison on Multi-PIE database

Method	0^o	15^o	30^o	45^o	60^o	Avg.
FIP [199]	94.3	90.7	80.7	64.1	45.9	72.9
MVP [200]	95.7	92.8	83.7	72.9	60.1	79.3
MvDN [93]	96.1	93.1	83.3	75.1	61.2	80.1
CPF [188]	**99.5**	95.0	88.5	79.9	61.9	83.3
DR-GAN [160]	97.0	94.0	**90.1**	86.2	83.2	89.2
MTAN w/o D	94.1	92.7	83.7	72.9	60.1	79.3
MTAN1	95.2	93.2	88.9	84.7	82.6	88.9
MTAN2	96.5	**95.3**	89.7	**87.9**	**84.1**	**89.6**

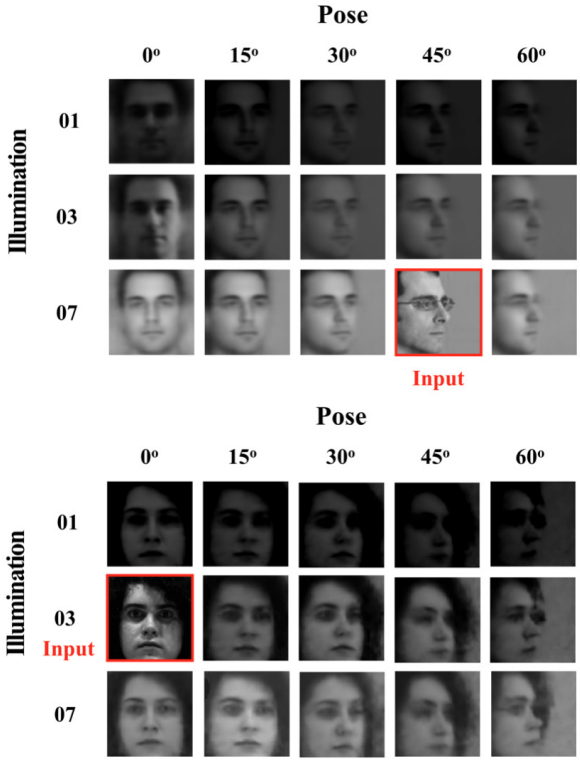


Fig. 5.7 Face synthesis with varying poses and illuminations conditioned on single input image (indicated by red boxes).

3. **MTAN2** ($D_{(C-S1-S2)} + G$): trained on all modules, using adversarial training and an image reconstruction requirement to disentangle three factors (identity, pose and illumination).

Our method shows an improvement for faces with extreme pose variations by disentangling the features using multi-task adversarial training and the image generation requirement. Compared with the other methods, the variation of recognition rates across different poses is

much lower except for DR-GAN, which suggests that our learned disentangled representation is more robust to the pose variation. The model MTAN2 which disentangles three latent factors further boosts the performance, which achieves comparable performance with the state-of-the-art performance, while not using multiple testing images as is done in DR-GAN. We also show some synthetic images in Fig. 5.7. In the synthetic images, the identity of the input image can be faithfully preserved and the style is controlled by arbitrary style (pose, illumination) indicators. This means that the learned content (identity) representation is largely disentangled from other style variations (pose and illumination).

5.5 Chapter Summary

In this chapter, we address the problem where multiple independent factors exist in the image generation process, but only some factors are of interest to us. We propose a new deep network architecture based on a novel type of multi-task learning to disentangle image variation factors in the learned feature representation. The network includes an encoder-generator structure as well as a set of adversarial discriminators. Through the interaction with the discriminators and generator, the encoder learns to extract features good for content factor recognition but not useful for style factor recognition. The overall network can be trained stably with a new loss function which is an extension of WGAN for multi-class scenario. Quantitative and qualitative evaluation on both font and face datasets demonstrate the superiority of our proposed model over the current state-of-the-art approaches.

Chapter 6

Re-weighted Adversarial Adaptation Network for Domain Adaptation

Conventional machine learning algorithms address isolated tasks; however, they work well only under the assumption that the training and test data are drawn from the same feature space and from the same distribution. When the distribution changes, most statistical models need to be rebuilt from scratch using newly collected training data. In many real-world applications, it is expensive or impossible to recollect the required training data and rebuild the models. It would be advantageous to reduce the need for and the effort required to recollect the training data. In this chapter, we incorporate prior knowledge about commonalities (common factors) between multiple learning tasks, in order to share statistical strength. More specifically, we leverage some labelled data that already exists for some related task or domain. As shown in Fig. 6.1, when multiple tasks share common factors, we can adapt the domain knowledge gained in solving the source task (domain) and apply it to our problem of interest, which is represented as the target task (domain). It is worth noting that domain adaptation is a common requirement in image classification tasks as often the data that has easily accessible labelled information and the data that we care about are different. There are two aspects concerning the benefits of using prior information about common factors in the feature learning model, as will now be discussed.

Firstly, exploiting common factors among multiple tasks can make the learned feature more robust to the variance of uncommon factors. It is worth noting that in contrast to the approach demonstrated in Chapter 5, exploiting common factors among multiple tasks does not require us to define the type of nuisance factors we want to remove from the representation before learning. It can be interpreted as a way to wipe all the uncommon factors (out of our interest) from the feature representation.

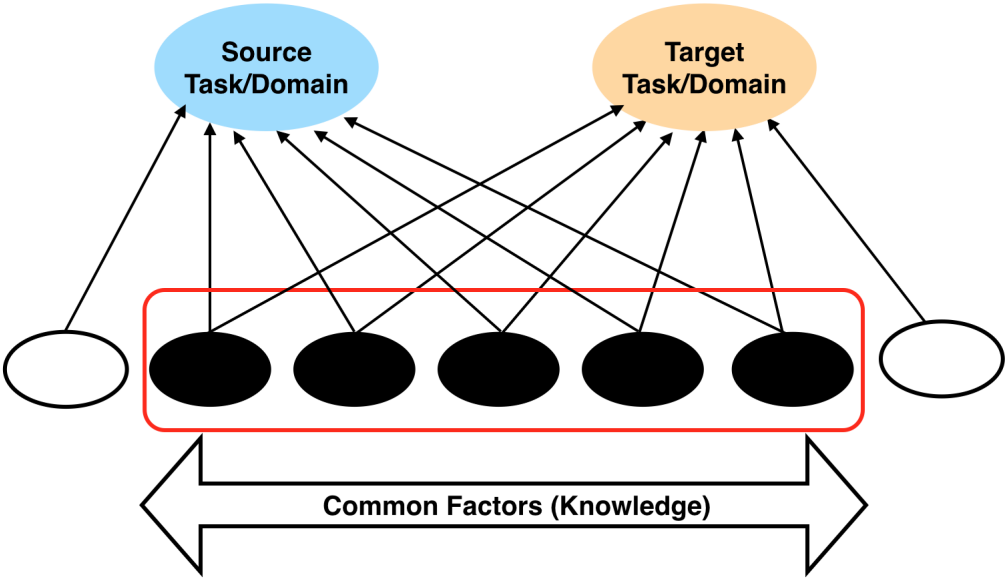


Fig. 6.1 Multiple Tasks share common factors.

Secondly, by exploiting common factors among multiple tasks, the model can generalise from very few labelled samples. As unlabelled data is easy and inexpensive to obtain, we can leverage unsupervised or semi-supervised learning to extract information from unlabelled data, thereby reducing the reliance on labelled samples. Unsupervised domain adaptation (UDA) is a specific application belonging to this situation, where no labelled data in the target domain are available while a lot of labelled data in the source domain are available. We will discuss more details about UDA in this chapter.

UDA aims to transfer domain knowledge (common factors) from existing well-defined tasks to new ones where labels are unavailable. The common factors in conventional UDA tasks is that two domains share the same set of image categories. In real-world applications, since the domain (task) discrepancies are usually uncontrollable, there is significant motivation to match the feature distributions even if the domain discrepancies are disparate. Additionally, as no label is available in the target domain, how to successfully adapt the classifier from the source to the target domain remains an open question. In this chapter, we propose the Re-weighted Adversarial Adaptation Network (RAAN) to reduce the feature distribution divergence and adapt the classifier when domain discrepancies are disparate. The rest of this chapter is organised as follows: Section 6.1 provides the primary motivation and clarifies our main contribution on UDA. Section 6.2 summarises the most relevant prior work including feature distribution matching approaches and instance re-weighting schemes. Section 6.3 presents the details of the proposed re-weighted adversarial

adaptation network. Specifically, to alleviate the need for common supports in matching the feature distribution, we aim to minimise optimal transport (OT) based Earth-Mover distance (EMD) by reformulating it to a minimax objective function. To further adapt the classifier, we also match the label distribution and embed it into the adversarial training. In Section 6.4, extensive experiments are performed on UDA datasets of varying difficulty that demonstrate the effectiveness of our proposed approach. We also provide an ablation study of the proposed approach. Finally Section 6.5 concludes this chapter.

6.1 Introduction

Recent developments in Convolutional Neural Networks (CNNs) have yielded state-of-the-art results from supervised learning applications in computer vision [49][85][150]. However, the success of CNNs requires a large amount of well annotated training data which is not always feasible to perform manually. Therefore, this has acted as a driver to transfer knowledge from datasets for which labels are well-defined. The Domain Adaptation (DA) problem [133] was proposed in this context where the data distribution between the target domain (where only a few labels are available) and the source domain (well-annotated labels) varies so that the discriminative features and the classifiers in the source domain cannot be transferred to the target domain[133][174]. Under this regime, unsupervised domain adaptation (UDA) is the most challenging problem where no label information in the target domain is available. To successfully conduct adaptation between domains in UDA, two essential problems are required to be addressed, specifically matching the feature distribution and adapting the classifier from source to target domains.

Since CNN based methods exhibit strong capacity to extract transferable feature representations among datasets, research has been conducted investigating measurements to estimate distribution divergence of deep features among domains and the relevant methods to minimize them. As an un-biased estimate of distribution divergence, Maximum Mean Discrepancy (MMD) [77] has been employed in various CNN based methods for UDA [110] [113] [114] [163] [165] [179]. More recently, inspired by the best-performing adversarial training in generative models, state-of-the-art UDA methods utilize the Jenson-Shannon (JS) divergence or the more generalized f-divergence [130] implemented using CNNs to estimate the distribution divergence [19] [61] [62] [109] [162].

However, both the MMD and f-divergence based methods require that feature distributions of the source and target domain share a common support (significant overlap exists between the feature distributions of two domains). We argue that this is an unrealistic condition that can rarely be met in the real-world adaptation tasks, since the domain discrepancies

are caused by a variety of factors that are difficult to control [39], such as light conditions, acquisition devices or even from different image formats e.g., Red-Green-Blue (RGB) and HHA[1] [80]. From this point of view, these methods fail to adapt between domains once their distributions do not have significant overlap. More recently, to alleviate the need of a common support in UDA, optimal transport (OT) based methods have been proposed to match the source and target feature distributions by minimizing the global transportation efforts [39] [40]. However, OT based methods have not been formalized and embedded into an end-to-end pipeline to train CNNs, which limits its application to large-scale UDA problems.

In UDA, besides selecting a good divergence measure of the marginal feature distribution, it is essential to adapt the classifier between domains. Long et.al [112][113] and Courty.et.al both [39] proposed to match the joint distribution of features and labels by regarding the transductive features from the final layer's activation map of the CNN as an approximation of the target domain labels. In fact, how to match the feature distribution and meanwhile adapting the classifier is still an open question in UDA.

In this chapter, we propose a Re-weighted Adversarial Adaptation Network (RAAN) for UDA to reduce disparate domain discrepancies and to adapt the classifier. More specifically, there are two main contributions:

1. To match feature distributions when domains discrepancies are disparate, we train a domain discriminator together with the feature encoders in an adversarial manner to minimize the OT based EMD. Compared with other methods adopting geometry-oblivious measures, RAAN can better reduce large feature distribution divergence.

2. To help adapt the classifier in UDA, we propose to match the label distribution by estimating a re-weighted source domain label distribution so that it can be similar to the unknown target label distribution. In addition, we embed it into the procedure of minimizing the EMD during the end-to-end adversarial training procedure. This not only adapts the classifier but also helps match the marginal feature distribution.

Finally, our proposed RAAN is evaluated by conducting a series of experiments using datasets with different domain distribution divergence.

[1]HHA encodes the image depth with three channels at each pixel: horizontal disparity, height above ground, and the angle the pixel's local surface normal makes with the inferred gravity direction.

6.2 Related Work

In this section, we review the state-of-the-art methods in reducing the domain distribution divergence for the UDA problem.

6.2.1 Matching Feature Distribution using Adversarial Training

JS-divergence based methods are the best-performing techniques for measuring the divergence of feature distributions in deep adaptation networks [19] [109] [162]. Although it is not a new statistical measure, the JS divergence or the f-divergence loss is widely adopted in CNNs trained in an adversarial manner [73] [130]. DANN [62] may be the first to add a domain classifier, with the aim of extracting not only discriminative features for the main classification task, but also indistinguishable ones for the domain classifiers. The adversarial loss of DANN is implemented by directly maximizing the domain classification loss and reversing the gradient in the back-propagation. DRCN [65] utilized the same approach but added another loss function to minimize the reconstruction error of the data samples between domains. More recently, ADDA [162] designed two separate feature extractors (one for each domain) to extract useful features for the main classification task. The domain discriminator network is added so that the target network and the domain discriminator network can compete with each other in an adversarial game until the target and source domain features cannot be distinguished.

Inspired by the good performance of adversarial training in generative models, some methods generate new images that are transferable in both domains. Co-GAN [109] may be the first to design two Generative Adversarial Nets (GANs) that generate diverse images for both source and target domains. Although Co-GAN achieved good performance in adapting domains having a small discrepancy, it cannot work well when the domain shifts are disparate [162]. In contrast to Co-GAN, the pixel-level domain adaptation network (pixelDA) proposed in [19] uses one generative network to generate images indistinguishable from source and target domains. In addition, constraints on pixel level similarity between the generated and source images are utilized. In fact, the ability of generative model based methods for UDA having large discrepancy is still under investigation.

6.2.2 Matching Feature Distribution using Optimal Transport

The most closely related approach to ours for reducing the distribution divergence is through solving the OT problem directly, as described in [39] [40]. However, this implementation has not been included into the end-to-end learning framework and only the stand-alone De-Caffe

features [49] from the DANN network are used. Instead, our proposed RAAN utilizes the domain discriminator network with the objective of minimizing the dual formulation of the Earth-Mover distance (EMD). From this perspective, the Wasserstein GAN [4] [79] is a special case to minimize the dual of the EMD, however, their ultimate goal is to generate images. To the best of author's knowledge, RAAN may be the first to learn domain invariant features for UDA utilizing the OT based EMD in a CNN. Note that the concurrent work [149] also adopted the EMD as the divergence measurement in UDA, however, we are handling the more generalized scenario with unbalanced datasets.

6.2.3 Instance Re-weighting Scheme

The instance re-weighting scheme is well documented in the literature [36] [189], for example in the instance re-weighting of the bias in the discriminative models [179], or in the causal inference regime [193]. In CNN based methods for UDA, Yan.et.al [179] recently proposed to learn the bias of the source domain instances by the classification expectation maximization (CEM) algorithms using the MMD as the divergence measure [29].

In contrast, our proposed approach (RAAN) differs from [179] as the instance re-weighting is achieved by estimating the density ratio vector of label distributions between domains. Specifically, the density ratio vector is embedded into the adversarial training, that can be learned or estimated via back-propagation. Finally, we also argue and explain why matching the label distribution helps to adapt the classifier in UDA.

6.3 Re-weighted Adversarial Adaptation Network

First, we introduce the notation and formulate our problem. Suppose we are given a n_c-class source domain set $\boldsymbol{S} = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ including n_s images $\boldsymbol{x}_i^s$ labeled by y_i^s and an unlabelled n_c-class target domain set $\boldsymbol{T} = \{(\boldsymbol{x}_j^t)\}_{j=1}^{n_t}$ composed of n_t images $\boldsymbol{x}_j^t$. The random variables representing the image and label in general are denoted as $\boldsymbol{X}$ and $\boldsymbol{Y}$ respectively. It is worth noting that the common factors between the two domains is that they have the same set of image categories. As illustrated in Fig. 6.2, RAAN is composed of four components, including the source feature encoder $\boldsymbol{E}_s$, the target feature encoder $\boldsymbol{E}_t$, the classifier $\boldsymbol{C}$ and the domain discriminator $\boldsymbol{D}$. The general function of each component and motivation for the design are provided in the caption of Fig. 6.2, with additional detail in the follow text.

The first objective of RAAN is to adapt the classifier, which is difficult without the target domain labels. However, as the label is a low-dimensional and discrete variable, it is fairly straightforward to match between domains and we argue that this can assist

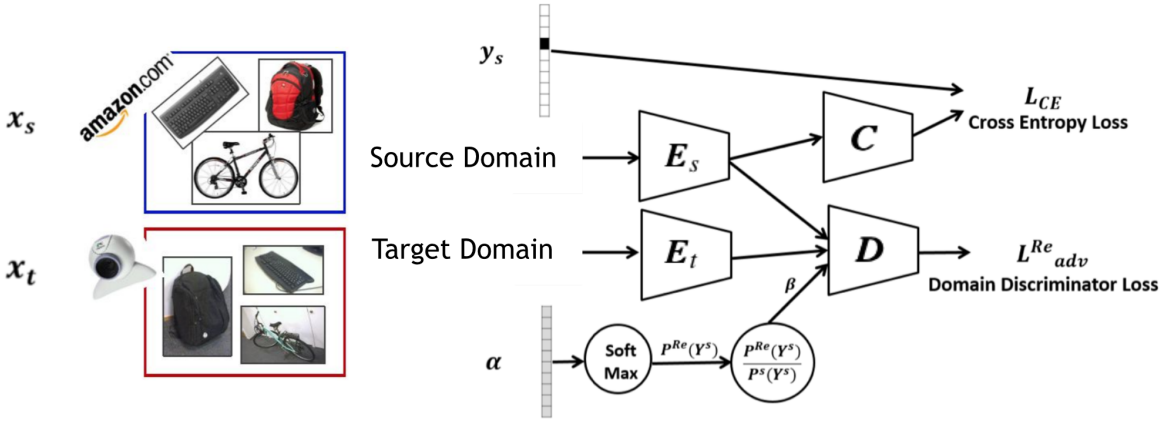


Fig. 6.2 RAAN's architecture: first, in the source domain, the source feature encoder E_s and the classifier C are trained to extract discriminative features from images $\boldsymbol{x}_s$ labeled by y_s by minimizing the cross entropy loss $\mathcal{L}_{CE}$. Second, to adapt the classifier by matching the label distribution between domains, the re-weighted source domain label distribution $P^{Re}(\boldsymbol{Y}^s)$ is computed by transforming a learnable variable $\boldsymbol{\alpha}$ using the soft-max function. We use the soft-max function in order to guarantee that the sum of all entries in $P^{Re}(\boldsymbol{Y}^s)$ is equal to 1. Then it is straightforward to obtain the ratio vector as follows: $\boldsymbol{\beta} = \frac{P^{Re}(\boldsymbol{Y}^s)}{P^s(\boldsymbol{Y}^s)}$. To extract transferable features for the target domain images $\boldsymbol{x}^t$, the target feature encoder E_t, domain discriminator D and the estimated density ratio vector $\boldsymbol{\beta}$ play the following adversarial game: $\boldsymbol{\beta}$ and D try to discriminate whether features are from the target or source domain, while E_t tries to confuse D and $\boldsymbol{\beta}$.

with the adaptation of classifiers (see the reasons in section 6.3.2). With this intuition, a re-weighted source domain label distribution $P^{Re}(\boldsymbol{Y}^s)$ is obtained by mapping a variable $\boldsymbol{\alpha} \in \mathbb{R}^{n_c}$ using the soft-max function. We use of the soft-max function here is to guarantee that each entry in $P^{Re}(\boldsymbol{Y}^s)$ is not less than zero and that the sum of all entries in $P^{Re}(\boldsymbol{Y}^s)$ is equal to 1. Then the estimation of $\boldsymbol{\alpha}$ aims to ensure that $P^{Re}(\boldsymbol{Y}^s)$ is similar to the unknown target label distribution $P^t(\boldsymbol{Y}^t)$. The density ratio vector is denoted as $\boldsymbol{\beta} \in \mathbb{R}^{n_c}$, with its $(y^s)^{th}$ element $\boldsymbol{\beta}(y^s)$ calculated by $\boldsymbol{\beta}(y^s) = \frac{P^{Re}(\boldsymbol{Y}^s=y^s)}{P^s(\boldsymbol{Y}^s=y^s)}$. As $\boldsymbol{\beta}$ can be directly computed based on $\boldsymbol{\alpha}$, consequently in the following context, we regard $\boldsymbol{\beta}$ as the variable under estimation. After learning the ratio vector $\boldsymbol{\beta}$, we can now reweight the source domain instances (assigning different significance to instances), in order to align the reweighted label distribution of the source domain with the unknown target label distribution, consequently helping the adaptation of the classifier.

The second objective of RAAN is to learn the Feature encoders E_s and E_t in order to extract the domain invariant features, so that the disparate divergence between marginal feature distributions $P^t(E_t(\boldsymbol{X}^t))$ and $P^s(E_s(\boldsymbol{X}^s))$ is reduced. Given the images and labels in

the source domain $\{\boldsymbol{x}^s, y^s\} \in \boldsymbol{S}$, with the aim of extracting discriminative features $E_s(\boldsymbol{x}^s)$ for the classification, it is straight-forward to train the classifier C and source feature encoder E_s cooperatively by minimizing the cross-entropy loss $\mathcal{L}_{CE}$ as follows:

$$\min_{E_s,C} \mathcal{L}_{CE}. \tag{6.1}$$

To obtain transferable features $E_t(\boldsymbol{x}^t)$ without labels, the target feature encoder E_t is trained by playing an adversarial game with the domain discriminator D and the ratio vector $\boldsymbol{\beta}$ so that the divergence between the re-weighted feature distribution in the source domain $\boldsymbol{\beta}(y^s)P^s(E_s(\boldsymbol{x}^s))$ and the target domain $P^t(E_t(\boldsymbol{x}^t))$ is reduced. Additionally, to better reduce the divergence between disparate domains, the OT-based EMD is reformulated in an adversarial manner, with more details given in section 6.3.1. Specifically, RAAN is trained in the following adversarial manner, where D with the help of $\boldsymbol{\beta}$ aims to discriminate whether features are from source or target domains, while E_t tries to confuse them. Based on the discriminator loss $\mathcal{L}^{Re}_{adv}$, the following objective function can be obtained:

$$\min_{E_t} \max_{D,\boldsymbol{\beta}} \mathcal{L}^{Re}_{adv}. \tag{6.2}$$

In fact, besides helping the adaptation of the classifier, matching the label distributions also eases the difficulty of matching the marginal feature distribution. The possible reason may be: if we assume that the feature generation processes are the same between domains, that is $P^s(E^s(\boldsymbol{X}^s)|\boldsymbol{Y}^s) = P^t(E^t(\boldsymbol{X}^t)|\boldsymbol{Y}^t)$, then $P^{Re}(\boldsymbol{Y}^s) = P^t(\boldsymbol{Y}^t)$ helps match the marginal feature distributions $P^s(E^s(\boldsymbol{X}^s)) = P^t(E^t(\boldsymbol{X}^t))$.

In section 6.3.1, to match the marginal feature distributions between domains, an OT based EMD is introduced and implemented in an adversarial manner in RAAN. Then in section 6.3.2, we propose to match label distributions between domains and embed it in the adversarial training. We also explain why this helps to adapt the classifier and meanwhile to match marginal feature distributions. Finally in section 6.3.3, we formulate the final objective function of RAAN.

6.3.1 Optimal Transport in Adversarial Training

Suppose that the empirical distributions of source and target domain features are denoted as $\boldsymbol{\mu}^s$ and $\boldsymbol{\mu}^t$ respectively as follows:

$$\boldsymbol{\mu}^s = \sum_i^{n_s} p_i^s \delta_{E_s(\boldsymbol{x}_i^s)}, \boldsymbol{\mu}^t = \sum_j^{n_t} p_j^t \delta_{E_t(\boldsymbol{x}_j^t)} \tag{6.3}$$

where $\delta_{E_s(\boldsymbol{x}_i^s)}$ and $\delta_{E_t(\boldsymbol{x}_j^t)}$ are the Dirac functions at location $E_s(\boldsymbol{x}_i^s)$ and $E_t(\boldsymbol{x}_j^t)$ and p_i^s and p_j^t are their probability masses. Then, the joint probabilistic coupling, or the transportation plan [40] between feature distributions in source and target domains can be defined as $\boldsymbol{\gamma}$ with the marginals $\boldsymbol{\mu}^s$ and $\boldsymbol{\mu}^t$. In the discrete version, the set of probabilistic couplings $\boldsymbol{B}$ can be defined as the following:

$$\boldsymbol{B} = \left\{ \boldsymbol{\gamma} \in (\mathbb{R}^+)^{n_s \times n_t} | \boldsymbol{\gamma} \mathbf{1}_{n_t} = \boldsymbol{\mu}^s, \boldsymbol{\gamma}^T \mathbf{1}_{n_s} = \boldsymbol{\mu}^t \right\}. \tag{6.4}$$

In general, to reduce feature distribution divergence, OT based methods first estimate the optimal transportation plan between two distributions and then learn the feature transformation to minimize the cost of such a plan. Therefore, we first define the metric $J(\boldsymbol{\mu}^s, \boldsymbol{\mu}^t)$ in Eq. (6.5) to measure the total cost of transporting probability masses from target to source domains,

$$J(\boldsymbol{\mu}^s, \boldsymbol{\mu}^t) = \langle \boldsymbol{\gamma}, \boldsymbol{M} \rangle_F, \; with \; \boldsymbol{\gamma} \in \boldsymbol{B}, \tag{6.5}$$

where $\boldsymbol{M}$ is the distance matrix whose $(i,j)^{th}$ element is defined by the distance cost function $m(E_s(\boldsymbol{x}_i^s), E_t(\boldsymbol{x}_j^t))$ between features. The $(i,j)^{th}$ element $\boldsymbol{\gamma}(i,j)$ indicates how much mass is moved from $E_t(\boldsymbol{x}_j^t)$ to $E_s(\boldsymbol{x}_i^s)$, and F is the Frobenius dot product. Subsequently for brevity, we drop the index i, j to represent $\boldsymbol{x}_i^s, \boldsymbol{x}_j^t$ as $\boldsymbol{x}^s, \boldsymbol{x}^t$. After that, the OT $\boldsymbol{\gamma}_0$ can be estimated by minimizing the cost $J(\boldsymbol{\mu}^s, \boldsymbol{\mu}^t)$ in equation (6.6), with the *optimal* transportation cost or the well-known EMD [94] defined by $W(\boldsymbol{\mu}^s, \boldsymbol{\mu}^t)$ in equation (6.7) . Finally, assuming the ideal source domain features $E_s(\boldsymbol{x}^s)$ are available as a reference to guide the feature learning in target domains, it is intuitive to train the Target feature encoder E_t under the objective of minimizing the EMD Loss $W(\boldsymbol{\mu}^s, \boldsymbol{\mu}^t)$, as shown in (6.8),

$$\boldsymbol{\gamma}_0 = \underset{\boldsymbol{\gamma} \in \boldsymbol{B}}{\operatorname{argmin}} J(\boldsymbol{\mu}^s, \boldsymbol{\mu}^t) \tag{6.6}$$

$$W(\boldsymbol{\mu}^s, \boldsymbol{\mu}^t) = \min_{\boldsymbol{\gamma} \in \boldsymbol{B}} J(\boldsymbol{\mu}^s, \boldsymbol{\mu}^t) \tag{6.7}$$

$$\min_{T_t} W(\boldsymbol{\mu}^s, \boldsymbol{\mu}^t). \tag{6.8}$$

To avoid using linear programs or iterative algorithms to compute the solution that satisfies the constraint of matrix $\boldsymbol{\gamma}$ in Eq.(6.4) directly, the dual formulation of $W(\boldsymbol{\mu}^s, \boldsymbol{\mu}^t)$ is utilized in Eq.(6.9) and (6.10) (following equation(5.3) in [166]), considering the capability of batch-wise back-propagation in CNN. More specifically, we use the domain discriminator D and its variant $\hat{D}$ as two dual functions in the following:

$$W(\boldsymbol{\mu}^s, \boldsymbol{\mu}^t) = \max_{D\,\hat{D}} \mathcal{L}_{adv},\ where$$

$$\mathcal{L}_{adv} = \mathop{\mathbb{E}}_{\boldsymbol{x}^s \sim P^s(\boldsymbol{X}^s)} D(E_s(\boldsymbol{x}^s)) + \mathop{\mathbb{E}}_{\boldsymbol{x}^t \sim P^t(\boldsymbol{X}^t)} \hat{D}(E_t(\boldsymbol{x}^t)) \tag{6.9}$$

$$s.t. D(E_s(\boldsymbol{x}^s)) + \hat{D}(E_t(\boldsymbol{x}^t)) \leq m(E_s(\boldsymbol{x}^s), E_t(\boldsymbol{x}^t)). \tag{6.10}$$

In this chapter, we choose the following distance cost function $m(E_s(\boldsymbol{x}^s), E_t(\boldsymbol{x}^t)) = \|E_s(\boldsymbol{x}^s) - E_t(\boldsymbol{x}^t)\|$ for reasons of computational efficiency and also for permitting gradient measurements, however, this does not infer that it is the only function that could be selected. According to the constraint (6.10), the best value that function $\hat{D}(E_t(\boldsymbol{x}^t))$ can take is $-D(E_s(\boldsymbol{x}^s))$, since $m(E_s(\boldsymbol{x}^s), E_t(\boldsymbol{x}^t))$ is defined to be non-negative. In this way, the constraint in (6.10) is equivalent to ensuring that D is a 1-Lipschitz function, or alternatively its gradient norm is smaller than 1. Therefore, if we use (6.9) and (6.10) in (6.8) to replace the EMD, the target feature encoder E_t and the domain discriminator D can be trained based on the mini-max objective function in (6.11),

$$\min_{E_t} W(\boldsymbol{\mu}^s, \boldsymbol{\mu}^t) = \min_{E_t} \max_{D} \mathcal{L}_{adv},\ where$$

$$\mathcal{L}_{adv} = \sum_{(\boldsymbol{x}^s, y^s) \sim P^s(\boldsymbol{X}, \boldsymbol{Y}^s)} D(E_s(\boldsymbol{x}^s)) P^s(E_s(\boldsymbol{x}^s)|y^s) P^s(y^s) - \mathop{\mathbb{E}}_{\boldsymbol{x}^t \sim P^t(\boldsymbol{X}^t)} D(E_t(\boldsymbol{x}^t))$$

$$s.t. \| \bigtriangledown_{E_t(\boldsymbol{x}^t)} D(E_t(\boldsymbol{x}^t)) \|_2 \leq 1, \| \bigtriangledown_{E_s(\boldsymbol{x}^s)} D(E_s(\boldsymbol{x}^s)) \|_2 \leq 1. \tag{6.11}$$

6.3.2 Adapting the Classifier by Label Distribution Matching

Although OT based EMD is utilized to match feature distributions $P^s(E_s(\boldsymbol{X}^s))$ and $P^t(E_t(\boldsymbol{X}^t))$, we argue that it is not enough to successfully adapt the classifier from source to target domain, since $P^s(E_s(\boldsymbol{X}^s)) = P^t(E_t(\boldsymbol{X}^t))$ does not infer $P^s(\boldsymbol{Y}^s|E_s(\boldsymbol{X}^s)) = P^t(\boldsymbol{Y}^t|E_t(\boldsymbol{X}^t))$. However, according to Bayes rule in (6.12), instead of matching $P^t(\boldsymbol{Y}^t|E_t(\boldsymbol{X}^t))$ and $P^s(\boldsymbol{Y}^s|E_s(\boldsymbol{X}^s))$ directly, we can learn E_t under the objective of matching the product ot distribution between $P^s(E_s(\boldsymbol{X}^s)|\boldsymbol{Y}^s)P^s(\boldsymbol{Y}^s)$ and $P^t(E_t(\boldsymbol{X}^t)|\boldsymbol{Y}^t)P^t(\boldsymbol{Y}^t)$,

$$P(E(\boldsymbol{X})|\boldsymbol{Y})P(\boldsymbol{Y}) \propto P(\boldsymbol{Y}|E(\boldsymbol{X})). \tag{6.12}$$

In fact, as no label information in the target domain $P^t(\boldsymbol{Y}^t)$ is available, it is non-trivial to directly match $P^t(E_t(\boldsymbol{X}^t)|\boldsymbol{Y}^t)P^t(\boldsymbol{Y}^t)$ and $P^s(E_s(\boldsymbol{X}^s)|\boldsymbol{Y}^s)P^s(\boldsymbol{Y}^s)$. However, as the label is a low-dimensional and discrete variable whose distribution is well-defined, it is more straightforward to match label distributions between domains compared with its conditional

variant. Therefore, we take a step back and propose to estimate the re-weighted source domain label distribution $P^{Re}(\boldsymbol{Y}^s)$ so that it is similar to the unknown $P^t(\boldsymbol{Y}^t)$ in the target domain. In fact, we argue that matching the label distributions between two domains can also help the adaptation of the classifier, because at least part of $P^t(E_t(\boldsymbol{X}^t)|\boldsymbol{Y}^t)P^t(\boldsymbol{Y}^t)$ and $P^s(E_s(\boldsymbol{X}^s)|\boldsymbol{Y}^s)P^{Re}(\boldsymbol{Y}^s)$ is matched. Based on such an assumption, since the EMD has been adopted to match $P^t(E_t(\boldsymbol{X}^t)$ and $P^s(E_s(\boldsymbol{X}^s)$, we propose to embed the re-weighted label distribution $P^{Re}(\boldsymbol{Y}^s)$ into the procedure of matching the marginal feature distribution $P^s(E_s(\boldsymbol{X}^s))$ and $P^t(E_t(\boldsymbol{X}^t))$ in the adversarial training.

To estimate the re-weighted label distribution $P^{Re}(\boldsymbol{Y}^s)$, the following constraint should be considered:

$$\sum_{i=1}^{n_c} P^{Re}(\boldsymbol{Y}^s = y_i) = 1, \tag{6.13}$$

where, y_i indicates the label of the i^{th} class. However, this constraint has already been considered in the implementation via the use of the softmax function.

Finally, to estimate the re-weighted label distribution, if we directly replace $P^s(\boldsymbol{Y}^s)$ by $P^{Re}(\boldsymbol{Y}^s)$ in the mini-max objective function $\mathcal{L}_{adv}$ in (6.11), a new one $\mathcal{L}_{adv}^{Re}$ is obtained in (6.14), where the domain discriminator D, target feature encoder E_t and the ratio vector $\boldsymbol{\beta}$ are trained in the following manner: D and $\boldsymbol{\beta}$ are trained in a cooperative way to estimate the EMD, while E_t is trained to minimize the EMD. From the perspective of implementation, $\boldsymbol{\beta}$ can be regarded as assigning different significance to images $\boldsymbol{x}^s$ in the source domain, so that the mini-batches in the two domains are sampled from similar distributions, which helps D and E_t to focus on matching $P^s(E_s(\boldsymbol{x}^s))$ and $P^t(E_t(\boldsymbol{x}^t))$,

$$\begin{aligned}
&\min_{E_t}\max_{D,\boldsymbol{\beta}} \mathcal{L}_{adv}^{Re}, \text{ where} \\
&\mathcal{L}_{adv}^{Re} = \sum_{(\boldsymbol{x}^s,y^s)\sim P^s(\boldsymbol{X}^s,\boldsymbol{Y}^s)} D(E_s(\boldsymbol{x}^s))P^s(E_s(\boldsymbol{x}^s)|y^s)P^{Re}(y^s) - \mathop{\mathbb{E}}_{\boldsymbol{x}^t\sim P^t(\boldsymbol{X}^t)} D(E_t(\boldsymbol{x}^t)) \\
&\quad = \sum_{(\boldsymbol{x}^s,y^s)\sim P^s(\boldsymbol{X}^s,\boldsymbol{Y}^s)} D(E_s(\boldsymbol{x}^s))P^s(E_s(\boldsymbol{x}^s)|y^s)\boldsymbol{\beta}(y^s)P^s(y^s) - \mathop{\mathbb{E}}_{\boldsymbol{x}^t\sim P^t(\boldsymbol{X}^t)} D(E_t(\boldsymbol{x}^t)) \\
&\quad = \mathop{\mathbb{E}}_{(\boldsymbol{x}^s,y^s)\sim P^s(\boldsymbol{X}^s,\boldsymbol{Y}^s)} \boldsymbol{\beta}(y^s)D(E_s(\boldsymbol{x}^s)) - \mathop{\mathbb{E}}_{\boldsymbol{x}^t\sim P^t(\boldsymbol{X}^t)} D(E_t(\boldsymbol{x}^t)) \\
&\quad s.t. \| \bigtriangledown_{E_t(\boldsymbol{x}^t)} D(E_t(\boldsymbol{x}^t))\|_2 \leq 1, \| \bigtriangledown_{T_s(\boldsymbol{x}^s)} D(E_s(\boldsymbol{x}^s))\|_2 \leq 1.
\end{aligned}$$

6.3.3 Optimization in RAAN

As shown in Fig. 6.2, RAAN is proposed to jointly minimize the cross entropy loss of the source domain samples and to reduce the divergence of the extracted feature distributions. First, we define the empirical estimate of the loss function $\mathcal{L}_{adv}^{Re}$ as follows:

$$\mathcal{L}_{adv}^{Re} = \frac{1}{n_s}\sum_{i=1}^{n_s} D(\beta(y_i^s)E_s(\boldsymbol{x}_i^s)) - \frac{1}{n_t}\sum_{j=1}^{n_t} D(E_t(\boldsymbol{x}_j^t)). \tag{6.14}$$

Following on from the idea of controlling the 1-Lipschitz function of the domain discriminator network D, we explicitly constrain the gradient norm penalty [79] term as follows:

$$\mathcal{L}_{gp} = \| \bigtriangledown_{\hat{E}(\hat{\boldsymbol{x}})} \mathcal{L}_{adv}^{Re} - 1\|_2, \tag{6.15}$$

where $\hat{E}(\hat{\boldsymbol{x}})$ is the weighted interpolation of samples $E_t(\boldsymbol{x}^t)$ and $E_s(\boldsymbol{x}^s)$. In summary, the total objective function in RAAN is formulated in the following adversarial manner:

$$\min_{E_t,D,\boldsymbol{\beta}} -\mathcal{L}_{adv}^{Re} + \lambda_{gp}\mathcal{L}_{gp} + \lambda_{reg}\|\boldsymbol{\beta}\|_2, \tag{6.16}$$

$$\min_{E_t} -\frac{1}{n_t}\sum_{j=1}^{n_t} D(E_t(\boldsymbol{x}_j^t)), \tag{6.17}$$

$$\min_{E_s,C} \frac{1}{n_s}\sum_{i=1}^{n_s} \mathcal{L}_{CE}(C(E_s(\boldsymbol{x}_i^s)), y_i^s), \tag{6.18}$$

where $\mathcal{L}_{CE}(C(E_s(\boldsymbol{x}_i^s)), y_i^s)$ indicates the cross-entropy function between the predicted label $C(E_s(\boldsymbol{x}_i^s)$ from the classifier C and the ground truth label y_i^s. Note that to train the networks stably, the source feature encoder E_s is trained first while E_t and D are then trained to match the feature distributions between $E_t(\boldsymbol{x}^t)$ and $E_s(\boldsymbol{x}^s)$ in an adversarial manner. In addition, to stably learning the ratio vector, we add the ℓ_2 norm of β as the regularization term in (6.16). λ_{gp} and λ_{reg} indicate the regularization weights of the gradient penalty term and the ℓ_2 norm of the ratio vector respectively.

6.4 Experiment and Results

In this section, RAAN is evaluated in three UDA tasks, specifically one between handwritten digit datasets, one between cross-modality datasets and the final one between object datasets. We implement our approach based on Tensorflow and the relevant code is available at https://github.com/BeCarefulPlease/Ch6. For all the experiments, RAAN achieves

competitive results compared with the state-of-the-art methods and outperforms them by a large margin when the distribution divergence is large between domains. We also perform an ablation study to verify the effectiveness of our proposed approach.

6.4.1 Datasets and Experimental Settings

In this section, we first introduce the datasets and adaptation tasks we used to evaluate the performance, and then provide the implementation details. We selected these datasets because they are the benchmarks for the evaluation of unsupervised domain adaptation in the literature.

Adaptation Tasks and Dataset

The first UDA task adapts between four hand written digit datasets, including MNIST[105], USPS [44], SVHN [129] and MNIST-M [61]. As shown in Fig. 6.3(a), adaptation between these four datasets are of varying difficulty. MNIST and USPS are both composed of grey-scale images in a fairly well-controlled environment while images in MNIST-M are synthesized using patches from BSDS500 dataset [3] as the background and MNIST images as the foreground. To evaluate RAAN in reducing large domain discrepancies, SVHN is also explored which is composed of RGB images in more complicated real-world scenarios, e.g., including misalignment of images and different light conditions. In addition, note that the number of sub-class instances between SVHN and the other datasets are largely unbalanced.

To evaluating RAAN in reducing large domain shifts, the second adaptation task is designed using the NYU-D dataset [127], adapting from the indoor object images in RGB format to the depth variants encoded by the HHA format [80]. The 19-class dataset is extracted following the scheme in [162]. As shown in Fig. 6.3(b), the domain shifts between images in RGB and HHA formats are fairly large, mainly due to the low image resolutions and potential mis-alignments caused by the coarse cropping box. In addition, as shown by the number of instances in Table 6.3, this dataset has unbalanced sub-class instances. Furthermore, it is particularly challenging as the images from the target domain are in a completely different format from those in the source domain.

We additionally evaluate RAAN's performance using the benchmark in the Office-Caltech dataset released in [70] to evaluate its performance on a small-scale dataset. The Office-Caltech dataset contains 10 categories in total and has 2533 images of which about half belong to Caltech256. We regard Amazon (A), DSLR (D), Webcam (W) and Caltech256 (C) as separate domains. In addition, note that the number of samples per category is unbalanced,

Fig. 6.3 DA datasets: (a) four hand-written digit datasets; (b) cross-modality dataset including RGB and RGB-depth images.

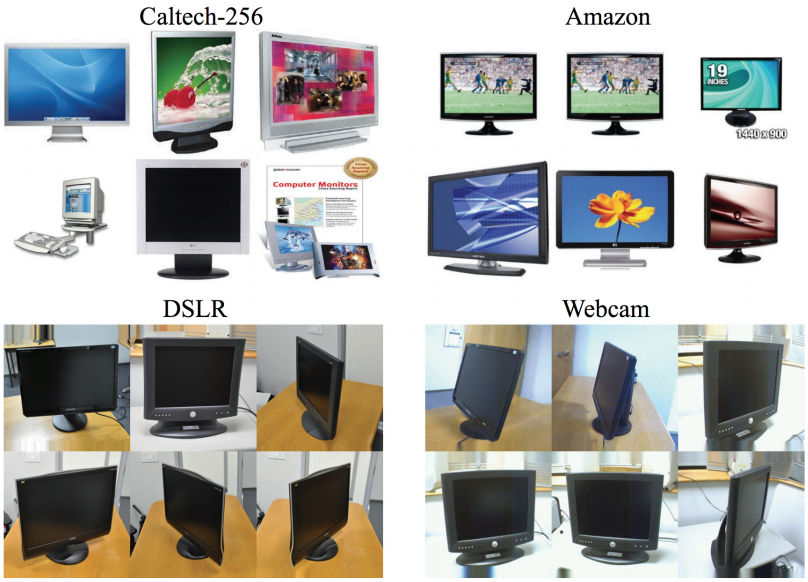


Fig. 6.4 Example images from the MONITOR category in Caltech256, Amazon, DSLR, and Webcam. Caltech and Amazon images are mostly from online merchants, while DSLR and Webcam images are from offices.

with approximately 8 to 151 samples per category per domain. Fig. 6.4 highlights the differences among these domains with example images from the MONITOR category.

Implementation Details

All the experiments are conducted on a GPU cluster.The regularization weights, λ_{gp} and λ_{reg} are chosen from the following sets $\{1, 10, 50, 100\}$ and $\{0.1, 1, 10, 50, 100, 500\}$ respectively. For all the experiments, we utilize the Adam optimizer, with the learning rate selected from the following set: $\{2e^{-5}, 5e^{-5}, 1e^{-4}, 2e^{-4}, 5e^{-4}, 1e^{-3}\}$. We used exponential decay for the learning rate, with a decay factor of 0.99 for every 1000 iterations. All experiments are run 10 times, each for 100000 iterations and we report the average results. For adapting from MNIST to MNIST-M, the batch size is 32 while for others, the batch size is 128.

6.4.2 Evaluation on Hand-Written Digit Dataset

For the task of adapting between hand written digit datasets, the following four adaptation directions are chosen for the evaluation: from MNIST to USPS, from USPS to MNIST, from SVHN to MNIST and from MNIST to MNIST-M. For the first three adaptation tasks, we adopt a variant of LeNet [105] as network architecture of the feature encoders E_s, E_t and the domain discriminator D is composed of three fully-connected layers activated by the rectified linear unit (ReLU) with output activation numbers of 512,512,1 respectively. For adapting from MNIST to MNIST-M, we adopt the basic model architecture of pixelDA [19] but change their domain discriminator network to an OT-based objective function and embed the ratio vector $\boldsymbol{\beta}$. As for the experiment protocols, we utilized the one in [111] for adapting between MNIST and USPS, while for adaptation from SVHN to MNIST, we choose that in [162]. The protocol used for adapting from MNIST to MNIST-M is the same as that in [19] to permit fair comparisons.

To assess the reasons underlying RAAN's performance, we evaluate RAAN(+) and RAAN(-) as RAAN with and without the re-weighting scheme respectively. As shown in Table 6.1, when adapting between MNIST and USPS, compared with ADDA and Co-GAN, the proposed RAAN(-) and RAAN(+) achieved competitive results and RAAN(+) slightly outperforms RAAN(-). In the most difficult task, i.e., adapting from SVHN to MNIST, RAAN(-) and RAAN(+) achieved 80.7% and 89.2% respectively, outperforming the state-of-the-art ADDA by 4.7% and 13.2% respectively, while Co-GAN does not converge in this experiment. It seems that the weight-sharing approach utilized in Co-GAN is not capable of generating transferable images between disparate domains such as MNIST and SVHN. As RAAN(-) utilized the same CNN architecture to ADDA's, RAAN(-)'s superior performance is mainly owing to the OT based objective function. We hypothesize that the OT based objective function is able to better reduce feature distribution divergence when the domains are disparate, e.g., SVHN and MNIST. In addition, based on the fact that RAAN(+) achieves

Table 6.1 Recognition rates of adapting hand-written digit datasets; RAAN(+) and RAAN(-) indicate results with and without the re-weighting scheme respectively.

Methods	MNIST to USPS	USPS to MNIST	SVHN to MNIST
Source Only	0.725	0.612	0.593
Gradient Reversal[62]	0.771	0.730	0.739
Domain Confusion [163]	0.791	0.665	0.681
Co-GAN[109]	**0.912**	0.891	No Converge
ADDA [162]	0.894	0.901	0.760
RAAN(-)(Ours)	0.883	0.915	0.807
RAAN(+)(Ours)	0.89	**0.921**	**0.892**

Table 6.2 Recognition rates of adapting from MNIST to MNIST-M; RAAN(+) and RAAN(-) indicate results with and without the re-weighting scheme respectively.

Source Only[19]	CORAL[153]	MMD[19]	DANN [62]	DSN [20]	PixelDA[19]	RAAN(+)/(-)
0.636	0.577	0.769	0.774	0.832	0.982	**0.985**

superior performances to both ADDA and RAAN(-), we hypothesize that matching the label distribution helps adapt the classifiers, and embedding it into minimizing the EMD of the feature distributions can be regarded as two cooperative tasks.

For adaption from MNIST to MNIST-M, as shown in Table 6.2, RAAN achieves slightly better performance than pixelDA. In addition, as expected RAAN(-) and RAAN(+) achieve similar results since the label distribution of the two domains are quite similar. Although it has been argued that the domain shift between MNIST and MNIST-M is large for a conventional CNN based method[62], we argue that reducing the domain shift caused by the background images in MNIST-M is easier than reducing the one between MNIST and SVHN if a generative model is utilized. The possible reason may be that the domain shifts led by background patches in MNIST-M exhibit fewer details and variations than the one caused by the complex real-world variations present in SVHN. In addition, we argue that the adversarial training based generative model is good at generating such background patches. These considerations may suggest and explain the slight outperformance achieved by RAAN in this task, compared with RAAN's large improvement when adapting from SVHN to MNIST.

6.4.3 Evaluation on the Cross-modality Dataset

In this section, RAAN is evaluated in the presence of large domain shifts that confront the adaptation from RGB images to RGB-depth images (HHA). To enable a fair comparison, we follow ADDA's experimental set-up [162] and utilized the VGG-16 architecture [150]

Table 6.3 Adaptation results in cross-modality dataset; RAAN(+) and RAAN(-) indicate results with and without the re-weighting scheme respectively.

class	bathtub	bed	bookshelf	box	chair	counter	desk	door	dresser	garbage bin
No. instances	19	96	87	210	611	103	122	129	25	55
Source Only	0.000	0.760	0.000	0.105	0.131	0.029	0.090	0.457	0.000	0.036
ADDA	0.000	0.469	0.000	0.005	**0.762**	0.194	0.016	0.519	0.040	0.018
RAAN(-) (ours)	0.000	**0.500**	**0.299**	0.000	0.629	0.019	0.000	0.271	0.000	0.000
RAAN(+) (ours)	0.000	0.104	0.000	**0.148**	0.703	**0.485**	0.000	**0.612**	0.000	0.000
$Est.\boldsymbol{\beta}$	2.040	1.395	1.568	0.872	0.492	1.440	1.297	1.206	1.640	1.629

class	lamp	monitor	night stand	pillow	sink	sofa	table	television	toilet	Overall
No. instances	144	37	51	276	47	129	210	33	17	2401
Source Only	0.125	0.000	0.000	0.549	0.000	0.023	0.148	0.030	0.000	0.189
ADDA	0.007	0.000	0.000	0.083	0.000	0.062	0.138	0.000	0.000	0.276
RAAN(-) (ours)	0.000	0.000	0.000	**0.692**	0.000	0.000	0.043	0.000	0.000	0.308
RAAN(+) (ours)	0.007	0.000	0.000	0.638	**0.106**	0.000	**0.205**	0.000	0.000	**0.343**
$Est.\boldsymbol{\beta}$	1.057	1.858	1.652	0.805	1.527	1.102	0.682	1.814	1.858	−−

for feature encoders E_s and E_t. The domain discriminator network D is composed of three fully-connected layers activated by the Relu, with 1024,2048,1 outputs respectively.

As shown in Table 6.3, we report the sub-class classification accuracy achieved by RAAN(-) and RAAN(+), along with the the estimated ratio vector $\boldsymbol{\beta}$ yielded by RAAN(+), to align the reweighted source label distribution $P^{Re}(\boldsymbol{Y^s})$ and the target distribution $P^t(\boldsymbol{Y^t})$. It can be observed from the overall recognition rates that RAAN(+) achieves an average of 34.3%, outperforming ADDA by 6.7% and RAAN(-) by 3.5%. In addition, RAAN(-) outperforms ADDA by 3.2%. For classes with fewer samples, RAAN(+) and RAAN(-) achieve better performances than ADDA. In fact, ADDA only achieved better performance in class 'chair' because that class has the largest number of samples. It can also be seen that RAAN(+) outperforms RAAN(-) not only in terms of the overall recognition accuracy but also from how many classes the classifier can recognize (classes with the recognition rates more than 0%). This is potentially due to the fact that the re-weighting scheme increases the significance of instances from the sub-classes with fewer instances. This can be verified by comparing the number of sub-class instances with the estimated ratio vector $\boldsymbol{\beta}$ in Table 6.3. The sub-classes with fewer instances, generally are assigned with a larger $\boldsymbol{\beta}$, that increases the significance of each instance from those sub-classes in classifier training.

Table 6.4 Adaptation Result on the Office-Caltech Dataset

	Source Only	MMD	DANN [62]	CORAL[153]	RAAN (-)	RAAN (+)
$A \rightarrow C$	84.0	**88.7**	87.8	86.2	87.1	**88.4**
$A \rightarrow D$	81.1	90.5	82.5	91.2	**93.1**	**93.1**
$A \rightarrow W$	75.4	91.6	77.8	90.5	89.5	**92.0**
$C \rightarrow A$	91.6	93.1	93.3	93.0	**93.7**	93.3
$C \rightarrow D$	87.7	91.2	91.2	89.5	94.2	**95.4**
$C \rightarrow W$	84.2	91.6	89.5	92.6	91.8	**93.6**
$D \rightarrow A$	84.4	90.1	84.7	85.8	91.3	**92.8**
$D \rightarrow W$	96.8	**99.0**	**99.0**	97.9	97.9	98.5
$D \rightarrow C$	80.5	87.8	82.1	85.3	89.8	**91.4**
$W \rightarrow A$	78.8	92.2	83.0	88.4	93.0	**93.2**
$W \rightarrow D$	98.3	100	100	100	100	**100**
$W \rightarrow C$	79.7	88.6	81.3	88.6	89.4	**89.8**

6.4.4 Evaluation on the Office-Caltech Dataset

In this section, we conducted 12 experiments in this dataset that is composed of four domains: Amazon (A), Caltech (C), Dslr (D) and Webcam (W). The feature encoders E_s and E_t are selected as the Alexnet network [100], so that a fair comparison with the state-of-the-art DANN method [62] can be performed. The domain discriminator network D is designed as three fully connected layers with the output activation number of 512, 512 and 1.

Table 6.4 shows the detailed comparison results of different approaches in 12 adaptation tasks. We observe that our proposed RAAN(+) outperforms all the other compared approaches in 10 out of the 12 domain adaptation tasks, and it achieves the second best scores in the remaining 2 tasks. Thus we can draw a conclusion that RAAN(+) can also be applied effectively to small-scale datasets. The matching of the joint feature distribution does not greatly rely on a large number of training samples. Another interesting observation is that our method RAAN (-) in most cases also outperforms DANN which also uses adversarial training. This illustrates the effectiveness of using the OT based objective function. In addition, it is worth noting that RAAN(+) achieves superior performance to RAAN(-), that demonstrates the effectiveness of matching the label distribution in the domain adaptation tasks.

6.4.5 Analysis

In this section, we analyze the results presented in the previous sections, including the re-weighting scheme in adversarial training and the domain distribution divergence in both quantitative and qualitative ways. The evaluation is in the context of the most challenging scenario, which involves adapting from SVHN to MNIST.

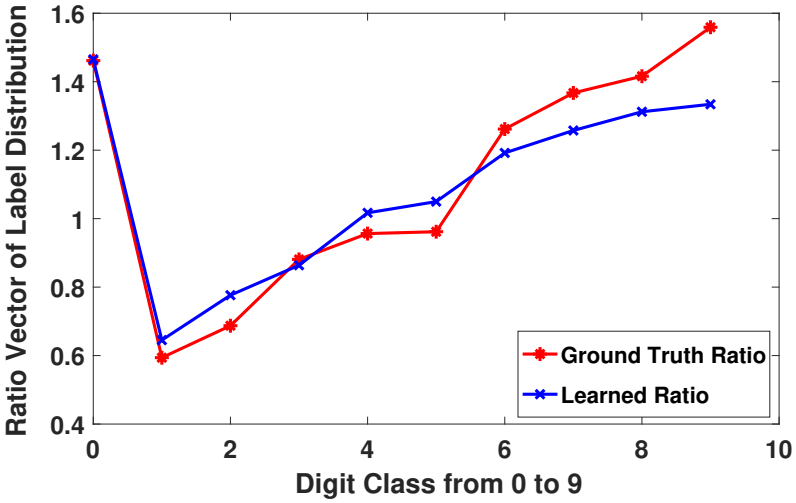


Fig. 6.5 Ratio of label distribution $\boldsymbol{\beta}$ between SVHN and MNIST; red line indicates the ground truth ratio, while blue one indicates the estimated ratio.

As the label distributions between SVHN and MNIST are largely mismatched, it will confuse the domain discriminator and the feature distributions will be matched in a biased manner. In addition, the mismatch of label distribution will directly give rise to the mismatch of classifiers as well. However, as shown in Fig. 6.5, RAAN(+) successfully matches the distribution of labels by simply learning the ratio vector embedded in the adversarial training. Therefore, this can be regarded as the main reason for the 9% improvement achieved by RAAN(+) compared to RAAN(-) shown in Table 6.1. To sum up, matching the label distribution can better adapt the classifiers.

To understand the instance re-weighting scheme intuitively, it is implemented by assigning different significance to the source domain instances. For example, as shown in Fig. 6.5, the learned ratio of digit "0" is around 1.5, which means that in the adversarial training, each sample from digit "0" in SVHN dataset can be regarded as effectively 1.5 samples.

Evaluate Distribution Divergence of Feature Embeddings

To analyze the distribution divergence in a quantitative way, we calculate the $\mathcal{A}$ distance suggested by the UDA community [7] [122], taking the input features extracted by various methods. Using the SVM classifier, we first calculate the generalization error θ of classifying the source and target domain features as a binary classification task. Then the $\mathcal{A}$ distance can be calculated as follows: $d_A = 2(1-2\theta)$. As shown in the Table 6.5, the $\mathcal{A}$ distances of feature embeddings with Source Only (no adaptation), adapted by ADDA, OT based RAAN(-

Table 6.5 $\mathcal{A}$-Distance of Adversarial Training Method

Metric	Source Only	ADDA	RAAN(-)	RAAN(+)
$\mathcal{A}$-Distance	1.673	1.548	1.526	1.506

) and RAAN(+) progressively decrease. In the experiment, since RAAN uses the same CNN architecture as ADDA's, compared with ADDA, the lower $\mathcal{A}$ distance achieved by RAAN(-) infers that feature distribution between domains can be better matched using RAAN(-). This may be due to the fact that the OT based EMD is a better measure for reducing large distribution divergence than the geometry-oblivious JS divergence. In addition, compared to RAAN(-), the smaller $\mathcal{A}$ distance achieved by RAAN(+) indicates that matching the label distribution and the feature distribution are two cooperative tasks and this cooperative training may be the main reason for the lower $\mathcal{A}$ distance.

Evaluation of the Re-weighting Scheme

In Fig. 6.5, we evaluated the re-weighting scheme by comparing the ground truth label ratio vector (red) and the learned one (blue). It can be seen that some ratios are accurate while others are not. However, the relative ratio trend of the learned ratio vector $\boldsymbol{\beta}$ follows that of the ground truth.

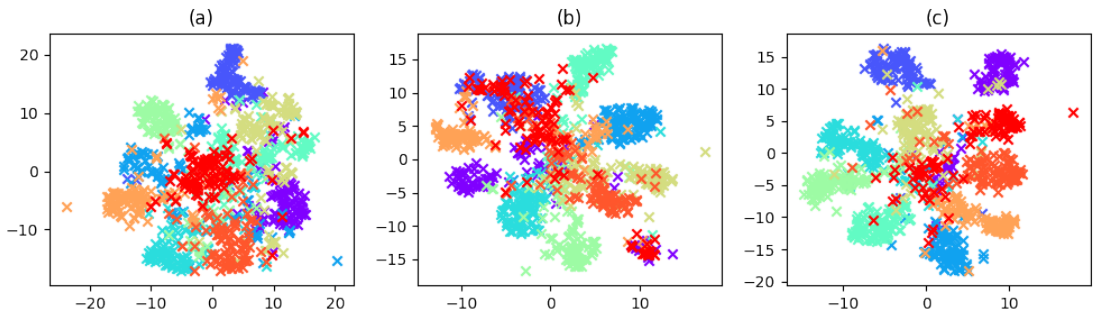


Fig. 6.6 T-SNE plot of features when adapting from SVHN to MNIST; (a) No adaptation (b) Adaptation after ADDA (c) Adaptation after RAAN. We randomly select 1000 features samples from 10 classes, with 100 samples per class.

.

Finally, to measure the feature distribution divergence in a qualitative way, we utilized the T-SNE software package [118] to visualize the 2-D embedding of the extracted features. It can be seen in Fig. 6.6 that the example points from the same class adapted by RAAN in Fig. 6.6(c) are clustered closer than those in Fig. 6.6(b) by ADDA and also those without the adaptation method shown in Fig. 6.6(a).

6.5 Chapter Summary

In this chapter, we explore the significance and use of prior knowledge about common factors among multiple tasks/domains in discriminative feature learning. More specifically, for UDA tasks, we propose a Re-weighted Adversarial Adaptation Network (RAAN) to reduce disparate domain feature distribution and to adapt the classifier. Through an extensive set of experiments using various UDA datasets, RAAN outperforms state-of-the-art methods by a large margin when the domain distribution divergence is large. Therefore we argue that the OT based objective function in the adversarial training exhibits better properties to match distributions when they share less common support. In addition, embedding the estimation of the ratio vector into the adversarial training demonstrates that it is capable of matching the label distribution between domains and further also adapting the classifier. It is also shown that this scheme can help reduce feature distribution divergence.

Chapter 7

Conclusion and Future Work

This dissertation investigates incorporating several types of general-purpose prior knowledge into the feature learning process, in order to learn a discriminative and compact image representation, that is ideal for robust image classification. In this chapter, the work presented in this dissertation is summarized and possible areas for future work are outlined.

7.1 Contribution Summary

This dissertation focuses on exploring discriminative feature representation for robust image classification. Besides leveraging prior information that comes from the labels as most prevalent feature learning methods currently do, we also explore incorporating some other general types of prior knowledge and evaluate their effectiveness in improving the performance of image classification tasks. More specifically, we apply sparsity and hierarchical priors on the explanatory factors that describe data, in order to better discover the data structure. We also explore the priors on the relationship between multiple factors, i.e., multiple factors that can change independently, or common explanatory factors that exist in multiple tasks. As more powerful representation can be learned by implementing such general priors, our novel approaches achieve state-of-the-art results on challenging benchmarks.

Specifically, our contributions include:

1. In Chapter 3, we focus on exploring the role of a sparse prior in discriminative feature learning for the image classification task. The sparse prior encourages information disentangling and allows variable-size feature representation for different input images. The sparse prior can also serve as a regularizer in overfitting prevention. From the view of implementation, we incorporate structured sparsity and propose a support discrimination dictionary learning method, which finds a dictionary under which the

feature representation of images from the same class have a common sparse structure while the size of the overlapped signal support of different classes is minimised. The superior performance of the proposed approach in comparison to the state-of-art is demonstrated using face, object and scene datasets.

2. In Chapter 4, we present the work conducted concerning using both sparse and hierarchical priors on feature representation to achieve more powerful and discriminative features. Using a hierarchical prior promotes the reuse of features and can lead to an abstract representation that is more invariant to local changes of the input. From the view of implementation, by incorporating dictionary learning (i.e., sparse prior) and deep learning (hierarchical prior), we provide a unified framework to automatically select the most useful low-level features and effectively combine them into more powerful and discriminative features for our specific image classification problem. Moreover, we take advantage of the structure of the dictionary and design a new label discriminative regressor to further improve the discriminative capability and avoid overfitting. Our proposed approach is evaluated using various challenging scene datasets and shows superior performance to many state-of-the-art approaches.

3. In Chapter 5, we address the problem where multiple independent factors exist in the image generation process, but only some factors are of interest to us. More specifically, we build a model that disentangles the underlying factors of the image variants associated with particular attributes of interest, consequently benefiting image understanding. A disentangled feature representation is significantly more robust to complexly structured variations of images and enables controllable generation of new data through a generative model. From the view of implementation, we present a novel multi-task adversarial network based on an encoder-discriminator-generator architecture. The encoder and the discriminators are trained cooperatively on the factors of interest, but in an adversarial way on the factors of distraction. The generator provides further regularisation on the learned feature. The experiments conducted on face recognition and font recognition tasks show that our method outperforms the state-of-the-art methods

4. In Chapter 6, we address the problem where multiple learning tasks share common explanatory factors. Leveraging common factors cannot only make the learned feature representation more robust to the variance of uncommon factors but also enable the model to generalise from very few labelled samples. More specifically, we focus on the unsupervised domain adaptation task, i.e., no labelled data in the target domain are available while some labelled data in the source domain are available. By incorporating

common factors between the source and target domain, we propose the re-weighted adversarial adaptation network, that reduces the feature distribution divergence and adapts the classifier between two domains. The overall network can be trained in an end-to-end adversarial manner. Empirical evidence shows that the new approach outperforms state of the art methods on standard domain adaptation benchmarks, particularly when the domain shifts are disparate.

7.2 Future Work

Now we put forth some directions that can be pursued in the future.

7.2.1 Other Potential Priors in Supervised Learning Problems

Besides the general-purpose priors that we have explored in this dissertation, there are some other priors that are potentially useful for improving representation in supervised learning problems, i.e., classification tasks. For example, the prior knowledge about the temporal and spatial coherence is useful for sequence classification problems. Since consecutive (from a temporal sequence) or spatially nearby observations tend to preserve the same or relevant categorical concepts, leveraging the prior knowledge to enforce the slowly changing feature representation might be useful for capturing such categorical concepts. Some related work has been presented in [6] [125] [8] [132].

We expect the future successful application of feature learning will refine and increase the list of useful priors. Another potential direction is to incorporate multiple priors in a unified framework instead of focusing on only one. Discovering more valuable prior knowledge and using them in an appropriate combination can better discover the data structure, thus learning an informative feature representation and consequently improving the performance of many supervised learning problems. Research regarding training criteria that better take these priors into account will also be worthwhile.

7.2.2 Priors for Large-scale Unsupervised Learning Problems

In this dissertation, we explore the effectiveness of several general-purpose priors for discriminative feature learning in order to boost the image classification performance. However, in some applications, the annotated data is difficult and laborious to collect. Since unlabeled images and video are available in practically unlimited quantities, how to use them for large-scale representation learning is an interesting direction to explore. More specifically, self-supervised learning [48] has been proposed in the last few years as a powerful tool to

learn an effective representation without any manual labels. Self-supervision tasks generally involve taking a complex input signal, hiding part of it from the machine learning model, and then asking the model to fill in the missing information, e.g., relative position [47], colourization [195], the "exemplar" task [52], and motion segmentation [134]. It is worth investigating how to incorporate general-purpose priors and evaluate their effectiveness in self-learning problems. This motivates longer-term unanswered questions about the appropriate objectives for learning good representations in large-scale unsupervised problems.

7.2.3 Priors for Model Compression and Training Acceleration

In this dissertation, we combine some general-purpose priors with the deep neural networks. Although deep neural networks have recently achieved great success in many visual recognition tasks, they are computationally expensive and memory intensive, thus limiting their applicability in devices having limited memory and for applications having strict latency requirements. During the past few years, tremendous progress has been made in this area. Some prevailing approaches have taken the sparse priors into consideration for reducing redundant parameters that are not sensitive to the performance [81] [170]. Some low-rank priors are adapted to decompose the parameter matrix/tensor for estimating the informative parameters [158]. We recommend a comprehensive survey conducted concerning model compression [32].

However, how to select and incorporate effective prior knowledge into the model compression tasks is still somewhat under-investigated. Some other general-purpose priors can also be attempted using the weight sharing mechanism [164], quantification [34] or binarization [38] strategies, in order to perform model compression and acceleration in deep networks without significantly decreasing the model performance.

References

[1] Aharon, M. and Elad, M. (2008). Sparse and redundant modeling of image content using an image-signature-dictionary. *SIAM Journal on Imaging Sciences*, 1(3):228–247.

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

[3] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916.

[4] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223.

[5] Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202.

[6] Becker, S. and Hinton, G. E. (1992). Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161.

[7] Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. (2010). A theory of learning from different domains. *Machine learning*, 79(1):151–175.

[8] Bengio, Y. and Bergstra, J. S. (2009). Slow, decorrelated features for pretraining complex cell-like networks. In *Advances in neural information processing systems*, pages 99–107.

[9] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

[10] Bengio, Y., Delalleau, O., and Roux, N. L. (2006). The curse of highly variable functions for local kernel machines. In *Advances in neural information processing systems*, pages 107–114.

[11] Bengio, Y., Delalleau, O., and Simard, C. (2010). Decision trees do not generalize to new variations. *Computational Intelligence*, 26(4):449–467.

[12] Bengio, Y. et al. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

[13] Bengio, Y., LeCun, Y., et al. (2007). Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41.

[14] Berg, A., Deng, J., and Fei-Fei, L. (2010). Large scale visual recognition challenge (ilsvrc), 2010. *URL http://www. image-net. org/challenges/LSVRC*, 3.

[15] Berthelot, D., Schumm, T., and Metz, L. (2017). BEGAN: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*.

[16] Bioucas-Dias, J. M. and Figueiredo, M. A. (2007). A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration. *IEEE Transactions on Image processing*, 16(12):2992–3004.

[17] Bishop, C., Bishop, C. M., et al. (1995). *Neural networks for pattern recognition*. Oxford university press.

[18] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

[19] Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., and Krishnan, D. (2016a). Unsupervised pixel-level domain adaptation with generative adversarial networks. *arXiv preprint arXiv:1612.05424*.

[20] Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., and Erhan, D. (2016b). Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351.

[21] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.

[22] Cai, S., Zuo, W., Zhang, L., Feng, X., and Wang, P. (2014). Support vector guided dictionary learning. In *European Conference on Computer Vision*, pages 624–639. Springer.

[23] Candes, E. and Romberg, J. (2005). l1-magic: Recovery of sparse signals via convex programming. *URL: www. acm. caltech. edu/l1magic/downloads/l1magic. pdf*, 4:14.

[24] Candes, E. J. (2008). The restricted isometry property and its implications for compressed sensing. *Comptes rendus mathematique*, 346(9-10):589–592.

[25] Candès, E. J. et al. (2006). Compressive sampling. In *Proceedings of the international congress of mathematicians*, volume 3, pages 1433–1452. Madrid, Spain.

[26] Candes, E. J., Romberg, J. K., and Tao, T. (2006). Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223.

[27] Candes, E. J., Wakin, M. B., and Boyd, S. P. (2008). Enhancing sparsity by reweighted 1 minimization. *Journal of Fourier analysis and applications*, 14(5-6):877–905.

[28] Caruana, R. (1997). Multitask learning. *Machine Learning*, 28.

[29] Celeux, G. and Govaert, G. (1992). A classification em algorithm for clustering and two stochastic versions. *Computational statistics & Data analysis*, 14(3):315–332.

[30] Chartrand, R. and Yin, W. (2008). Iteratively reweighted algorithms for compressive sensing. In *Acoustics, speech and signal processing, 2008. ICASSP 2008. IEEE international conference on*, pages 3869–3872. IEEE.

[31] Chen, S. S., Donoho, D. L., and Saunders, M. A. (2001). Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159.

[32] Cheng, Y., Wang, D., Zhou, P., and Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*.

[33] Cheung, B., Livezey, J. A., Bansal, A. K., and Olshausen, B. A. (2014). Discovering hidden factors of variation in deep networks. *arXiv preprint arXiv:1412.6583*.

[34] Choi, Y., El-Khamy, M., and Lee, J. (2016). Towards the limit of network quantization. *arXiv preprint arXiv:1612.01543*.

[35] Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

[36] Cortes, C., Mansour, Y., and Mohri, M. (2010). Learning bounds for importance weighting. In *Advances in neural information processing systems*, pages 442–450.

[37] Cotter, S. F., Rao, B. D., Engan, K., and Kreutz-Delgado, K. (2005). Sparse solutions to linear inverse problems with multiple measurement vectors. *IEEE Transactions on Signal Processing*, 53(7):2477–2488.

[38] Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*.

[39] Courty, N., Flamary, R., Habrard, A., and Rakotomamonjy, A. (2017a). Joint distribution optimal transportation for domain adaptation. *arXiv preprint arXiv:1705.08848*.

[40] Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. (2017b). Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865.

[41] Damelin, S. B. and Miller Jr, W. (2012). *The mathematics of signal processing*, volume 48. Cambridge University Press.

[42] Daubechies, I., Defrise, M., and De Mol, C. (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on pure and applied mathematics*, 57(11):1413–1457.

[43] Deng, W., Yin, W., and Zhang, Y. (2013). Group sparse optimization by alternating direction method. In *SPIE Optical Engineering+Applications*, pages 88580R–88580R. International Society for Optics and Photonics.

[44] Denker, J. S., Gardner, W., Graf, H. P., Henderson, D., Howard, R., Hubbard, W., Jackel, L. D., Baird, H. S., and Guyon, I. (1989). Neural network recognizer for hand-written zip code digits. In *Advances in neural information processing systems*, pages 323–331.

[45] Denton, E. and Birodkar, V. (2017). Unsupervised learning of disentangled representations from video. *arXiv preprint arXiv:1705.10915*.

[46] Dixit, M., Chen, S., Gao, D., Rasiwasia, N., and Vasconcelos, N. (2015). Scene classification with semantic fisher vectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2974–2983.

[47] Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430.

[48] Doersch, C. and Zisserman, A. (2017). Multi-task self-supervised visual learning. In *The IEEE International Conference on Computer Vision (ICCV)*.

[49] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655.

[50] Donoho, D. L., Maleki, A., and Montanari, A. (2009). Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences*, 106(45):18914–18919.

[51] Donoho, D. L., Tsaig, Y., Drori, I., and Starck, J.-l. (2006). Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit, submitted to. In *IEEE Trans. on Information theory*. Citeseer.

[52] Dosovitskiy, A., Springenberg, J. T., Riedmiller, M., and Brox, T. (2014). Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 766–774.

[53] Dosovitskiy, A., Tobias Springenberg, J., and Brox, T. (2015). Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546.

[54] Elad, M. and Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745.

[55] Eldar, Y. C. (2009). Generalized sure for exponential families: Applications to regularization. *IEEE Transactions on Signal Processing*, 57(2):471–481.

[56] Eldar, Y. C. and Mishali, M. (2009). Robust recovery of signals from a structured union of subspaces. *IEEE Transactions on Information Theory*, 55(11):5302–5316.

[57] Elhamifar, E. and Vidal, R. (2011). Robust classification using structured sparse representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1873–1879.

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

[59] Fei-Fei, L., Fergus, R., and Perona, P. (2007). Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70.

[60] Figueiredo, M. A., Nowak, R. D., and Wright, S. J. (2007). Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of selected topics in signal processing*, 1(4):586–597.

[61] Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189.

[62] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

[63] Gao, B.-B., Wei, X.-S., Wu, J., and Lin, W. (2015). Deep spatial pyramid: The devil is once again in the details. *arXiv preprint arXiv:1504.05277*.

[64] Georghiades, A. S., Belhumeur, P. N., and Kriegman, D. J. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660.

[65] Ghifary, M., Kleijn, W. B., Zhang, M., Balduzzi, D., and Li, W. (2016). Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*, pages 597–613. Springer.

[66] Girshick, R. (2015). Fast r-cnn. *arXiv preprint arXiv:1504.08083*.

[67] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

[68] Glowinski, R. and Le Tallec, P. (1989). *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*, volume 9. SIAM.

[69] Glowinski, R. and Oden, J. (1985). Numerical methods for nonlinear variational problems. *Journal of Applied Mechanics*, 52:739.

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

[71] Gong, Y., Wang, L., Guo, R., and Lazebnik, S. (2014). Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision*, pages 392–407. Springer.

[72] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

[73] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

[74] Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. *arXiv preprint arXiv:1302.4389*.

[75] Gorodnitsky, I. F. and Rao, B. D. (1997). Sparse signal reconstruction from limited data using focuss: A re-weighted minimum norm algorithm. *IEEE Transactions on signal processing*, 45(3):600–616.

[76] Gregor, K. and LeCun, Y. (2010). Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 399–406.

[77] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773.

[78] Gross, R., Matthews, I., Cohn, J., Kanade, T., and Baker, S. (2010). Multi-pie. *Image and Vision Computing*, 28(5):807–813.

[79] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of wasserstein GANs. *arXiv preprint arXiv:1704.00028*.

[80] Gupta, S., Girshick, R., Arbeláez, P., and Malik, J. (2014). Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer.

[81] Han, S., Mao, H., and Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

[82] Hastad, J. (1986). Almost optimal lower bounds for small depth circuits. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 6–20. ACM.

[83] Håstad, J. and Goldmann, M. (1991). On the power of small-depth threshold circuits. *Computational Complexity*, 1(2):113–129.

[84] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

[85] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[86] Herranz, L., Jiang, S., and Li, X. (2016). Scene recognition with cnns: objects, scales and dataset bias. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 571–579.

[87] Hinton, G. E., Krizhevsky, A., and Wang, S. D. (2011). Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer.

[88] Huang, F. J., Boureau, Y.-L., LeCun, Y., et al. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

[89] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

[90] Jenatton, R., Audibert, J.-Y., and Bach, F. (2011). Structured variable selection with sparsity-inducing norms. *The Journal of Machine Learning Research*, 12:2777–2824.

[91] Ji, S., Xue, Y., and Carin, L. (2008). Bayesian compressive sensing. *IEEE Transactions on Signal Processing*, 56(6):2346–2356.

[92] Jiang, Z., Lin, Z., and Davis, L. S. (2013). Label consistent K-SVD: Learning a discriminative dictionary for recognition. volume 35, pages 2651–2664.

[93] Kan, M., Shan, S., and Chen, X. (2016). Multi-view deep network for cross-view classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4847–4855.

[94] Kantorovitch, L. (1958). On the translocation of masses. *Management Science*, 5(1):1–4.

[95] Kim, S.-J., Koh, K., Lustig, M., Boyd, S., and Gorinevsky, D. (2007). An interior-point method for large-scale ℓ_1-regularized least squares. *IEEE journal of selected topics in signal processing*, 1(4):606–617.

[96] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[97] Kline, D. M. and Berardi, V. L. (2005). Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing & Applications*, 14(4):310–318.

[98] Kong, S. and Wang, D. (2012). A dictionary learning approach for classification: separating the particularity and the commonality. In *European Conference on Computer Vision*, pages 186–199. Springer.

[99] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012a). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[100] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012b). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[101] Kukreja, S. L., Löfberg, J., and Brenner, M. J. (2006). A least absolute shrinkage and selection operator (lasso) for nonlinear system identification. *IFAC proceedings volumes*, 39(1):814–819.

[102] Kulkarni, T. D., Whitney, W. F., Kohli, P., and Tenenbaum, J. (2015). Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547.

[103] Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 2169–2178. IEEE.

[104] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

[105] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[106] Li, J., Zhang, T., Luo, W., Yang, J., Yuan, X.-T., and Zhang, J. (2017a). Sparseness analysis in the pretraining of deep neural networks. *IEEE transactions on neural networks and learning systems*, 28(6):1425–1438.

[107] Li, Y., Dixit, M., and Vasconcelos, N. (2017b). Deep scene image classification with the mfafvnet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5746–5754.

[108] Liu, B., Liu, J., Wang, J., and Lu, H. (2014). Learning a representative and discriminative part model with deep convolutional features for scene recognition. In *Asian Conference on Computer Vision*, pages 643–658. Springer.

[109] Liu, M.-Y. and Tuzel, O. (2016). Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477.

[110] Long, M., Cao, Y., Wang, J., and Jordan, M. (2015). Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*, pages 97–105.

[111] Long, M., Wang, J., Ding, G., Sun, J., and Yu, P. S. (2013). Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 2200–2207.

[112] Long, M., Wang, J., Ding, G., Sun, J., and Yu, P. S. (2014). Transfer joint matching for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1410–1417.

[113] Long, M., Wang, J., and Jordan, M. I. (2016a). Deep transfer learning with joint adaptation networks. *arXiv preprint arXiv:1605.06636*.

[114] Long, M., Zhu, H., Wang, J., and Jordan, M. I. (2016b). Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems*, pages 136–144.

[115] Louizos, C., Swersky, K., Li, Y., Welling, M., and Zemel, R. (2015). The variational fair autoencoder. *arXiv preprint arXiv:1511.00830*.

[116] Lounici, K., Pontil, M., Tsybakov, A. B., and Van De Geer, S. (2009). Taking advantage of sparsity in multi-task learning. *arXiv preprint arXiv:0903.1468*.

[117] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

[118] Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

[119] Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2009). Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696. ACM.

[120] Mairal, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A. (2008). Discriminative learned dictionaries for local image analysis. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[121] Manly, B. F. (1994). *Multivariate statistical methods: a primer*. CRC Press.

[122] Mansour, Y., Mohri, M., and Rostamizadeh, A. (2009). Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*.

[123] Martinez, A. M. (1998). The AR face database. *CVC Technical Report*, 24.

[124] Mathieu, M. F., Zhao, J. J., Zhao, J., Ramesh, A., Sprechmann, P., and LeCun, Y. (2016). Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems*, pages 5040–5048.

[125] Mobahi, H., Collobert, R., and Weston, J. (2009). Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 737–744. ACM.

[126] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

[127] Nathan Silberman, Derek Hoiem, P. K. and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *ECCV*.

[128] Needell, D. and Tropp, J. A. (2009). Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and computational harmonic analysis*, 26(3):301–321.

[129] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5.

[130] Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279.

[131] Odena, A. (2016). Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*.

[132] Oh, J., Guo, X., Lee, H., Lewis, R. L., and Singh, S. (2015). Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pages 2863–2871.

[133] Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[134] Pathak, D., Girshick, R., Dollár, P., Darrell, T., and Hariharan, B. (2017). Learning features by watching objects move. In *Proc. CVPR*, volume 2.

[135] Peng, X., Yu, X., Sohn, K., Metaxas, D. N., and Chandraker, M. (2017). Reconstruction-based disentanglement for pose-invariant face recognition. *intervals*, 20:12.

[136] Qi, G.-J. (2017). Loss-sensitive generative adversarial networks on lipschitz densities. *arXiv preprint arXiv:1701.06264*.

[137] Qian, G., Sural, S., Gu, Y., and Pramanik, S. (2004). Similarity between euclidean and cosine angle distance for nearest neighbor queries. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1232–1237. ACM.

[138] Quattoni, A. and Torralba, A. (2009). Recognizing indoor scenes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 413–420. IEEE.

[139] Rabaud, V. and Belongie, S. (2006). Counting crowded moving objects. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 705–711. IEEE.

[140] Ramirez, I., Sprechmann, P., and Sapiro, G. (2010). Classification and clustering via dictionary learning with structured incoherence and shared features. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3501–3508. IEEE.

[141] Rao, B. D., Engan, K., Cotter, S. F., Palmer, J., and Kreutz-Delgado, K. (2003). Subset selection in noise based on diversity measure minimization. *IEEE transactions on Signal processing*, 51(3):760–770.

[142] Rao, B. D. and Kreutz-Delgado, K. (1999). An affine scaling methodology for best basis selection. *IEEE Transactions on signal processing*, 47(1):187–200.

[143] Reed, S., Sohn, K., Zhang, Y., and Lee, H. (2014). Learning to disentangle factors of variation with manifold interaction. In *International Conference on Machine Learning*, pages 1431–1439.

[144] Rodriguez, F. and Sapiro, G. (2008). Sparse representations for image classification: Learning discriminative and reconstructive non-parametric dictionaries. Technical report, DTIC Document.

[145] Rubinstein, R., Zibulevsky, M., and Elad, M. (2008). Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. *CS Technion*, 40(8):1–15.

[146] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

[147] Saunders, M. A., Kim, B., Maes, C., Akle, S., and Zahr, M. (2002). Pdco: Primal-dual interior method for convex objectives. *Software available at http://www. stanford. edu/group/SOL/software/pdco. html*.

[148] Schmidhuber, J. (1992). Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879.

[149] Shen, J., Qu, Y., Zhang, W., and Yu, Y. (2017). Adversarial representation learning for domain adaptation. *arXiv preprint arXiv:1707.01217*.

[150] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

[151] Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

[152] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[153] Sun, B. and Saenko, K. (2016). Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops*, pages 443–450. Springer.

[154] Sun, Y., Wang, X., and Tang, X. (2015). Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2892–2900.

[155] Suo, Y., Dao, M., Tran, T., Mousavi, H., Srinivas, U., and Monga, V. (2014). Group structured dirty dictionary learning for classification. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 150–154. IEEE.

[156] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147.

[157] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., et al. (2015). Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*.

[158] Tai, C., Xiao, T., Zhang, Y., Wang, X., et al. (2015). Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*.

[159] Tenenbaum, J. B. and Freeman, W. T. (1997). Separating style and content. In *Advances in neural information processing systems*, pages 662–668.

[160] Tran, L., Yin, X., and Liu, X. (2017). Disentangled representation learning GAN for pose-invariant face recognition. In *CVPR*, volume 4, page 7.

[161] Tropp, J. A. and Gilbert, A. C. (2007). Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on information theory*, 53(12):4655–4666.

[162] Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial discriminative domain adaptation. *arXiv preprint arXiv:1702.05464*.

[163] Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., and Darrell, T. (2014). Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*.

[164] Ullrich, K., Meeds, E., and Welling, M. (2017). Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*.

[165] Venkateswara, H., Eusebio, J., Chakraborty, S., and Panchanathan, S. (2017). Deep hashing network for unsupervised domain adaptation. In *(IEEE) Conference on Computer Vision and Pattern Recognition (CVPR)*.

[166] Villani, C. (2008). *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

[167] Wang, H., Yuan, C., Hu, W., and Sun, C. (2012). Supervised class-specific dictionary learning for sparse modeling in action recognition. *Pattern Recognition*, 45(11):3902–3911.

[168] Wang, K., Lin, L., Zuo, W., Gu, S., and Zhang, L. (2016). Dictionary pair classifier driven convolutional neural networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2138–2146.

[169] Wang, Z., Liu, D., Yang, J., Han, W., and Huang, T. (2015). Deep networks for image super-resolution with sparse prior. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 370–378.

[170] Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. (2016). Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082.

[171] Wipf, D. and Nagarajan, S. (2010a). Iterative reweighted and methods for finding sparse solutions. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):317–329.

[172] Wipf, D. and Nagarajan, S. (2010b). Iterative reweighted ℓ_1 and ℓ_2 methods for finding sparse solutions. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):317–329.

[173] Wipf, D. P., Rao, B. D., and Nagarajan, S. (2011). Latent variable bayesian models for promoting sparsity. *IEEE Transactions on Information Theory*, 57(9):6236–6255.

[174] Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

[175] Wright, J., Ma, Y., Mairal, J., Sapiro, G., Huang, T. S., and Yan, S. (2010). Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044.

[176] Wright, S. J., Nowak, R. D., and Figueiredo, M. A. (2009). Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493.

[177] Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. (2010). Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE.

[178] Xie, G., Zhang, X., Yan, S., and Liu, C. (2017). Hybrid cnn and dictionary-based models for scene recognition and domain adaptation. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(6):1263–1274.

[179] Yan, H., Ding, Y., Li, P., Wang, Q., Xu, Y., and Zuo, W. (2017). Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. *arXiv preprint arXiv:1705.00609*.

[180] Yang, J., Yu, K., Gong, Y., and Huang, T. (2009). Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. IEEE.

[181] Yang, J., Yu, K., and Huang, T. (2010). Supervised translation-invariant sparse coding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3517–3524.

[182] Yang, J. and Zhang, Y. (2011). Alternating direction algorithms for l1-problems in compressive sensing. *SIAM journal on scientific computing*, 33(1):250–278.

[183] Yang, M., Dai, D., Shen, L., and Van Gool, L. (2014a). Latent dictionary learning for sparse representation based classification. In *Proceedings CVPR 2014*, pages 4138–4145.

[184] Yang, M., Zhang, L., Feng, X., and Zhang, D. (2011). Fisher discrimination dictionary learning for sparse representation. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 543–550. IEEE.

[185] Yang, M., Zhang, L., Feng, X., and Zhang, D. (2014b). Sparse representation based Fisher discrimination dictionary learning for image classification. *International Journal of Computer Vision*, 109(3):209–232.

[186] Yang, S. and Ramanan, D. (2015). Multi-scale recognition with dag-cnns. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1215–1223.

[187] Yi, D., Lei, Z., Liao, S., and Li, S. Z. (2014). Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*.

[188] Yim, J., Jung, H., Yoo, B., Choi, C., Park, D., and Kim, J. (2015). Rotating your face using multi-task deep neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 676–684.

[189] Yu, Y. and Szepesvári, C. (2012). Analysis of kernel mean matching under covariate shift. *arXiv preprint arXiv:1206.4650*.

[190] Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

[191] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

[192] Zeng, J., Lin, S., and Xu, Z. (2014). Sparse solution of underdetermined linear equations via adaptively iterative thresholding. *Signal Processing*, 97:152–161.

[193] Zhang, K., Schölkopf, B., Muandet, K., and Wang, Z. (2013). Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*, pages 819–827.

[194] Zhang, Q. and Li, B. (2010). Discriminative k-svd for dictionary learning in face recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2691–2698. IEEE.

[195] Zhang, R., Isola, P., and Efros, A. A. (2016). Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer.

[196] Zhao, N., Li, C., Shi, H., and Lin, C. (2011). Multi-frame image super-resolution based on regularization scheme. In *Control, Automation and Systems Engineering (CASE), 2011 International Conference on*, pages 1–4. IEEE.

[197] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495.

[198] Zhou, N., Shen, Y., Peng, J., and Fan, J. (2012). Learning inter-related visual dictionary for object recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3490–3497. IEEE.

[199] Zhu, Z., Luo, P., Wang, X., and Tang, X. (2013). Deep learning identity-preserving face space. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 113–120.

[200] Zhu, Z., Luo, P., Wang, X., and Tang, X. (2014). Multi-view perceptron: a deep model for learning face identity and view representations. In *Advances in Neural Information Processing Systems*, pages 217–225.